Safety Support Program for Automotive

Application Note for POST and Runtime Test

Introduction

Vehicle and passenger safety are inherently at risk with the increased usage of electrical and/or electronic (E/E) components and systems. This risk is due to the occurrence of faults in the E/E systems that can lead to failures in the vehicle operation and result in harm (injury) to the passengers. ISO26262 ([1] INTERNATIONAL STANDARD ISO 26262 Road vehicles — Functional safety Second edition 2018-12) dictates that E/E related faults are mitigated to prevent failures from occurring.

Semiconductors (MCUs, System-on-Chip, memories, etc.) are examples of E/E components that must be analyzed to mitigate these theoretical faults. One method for fault mitigation in E/E components is to implement a safety mechanism to check for existing faults in the elements (e.g. transistors, memory cells) within a specified time interval (FTTI: fault-tolerant time interval) and notify the system that a fault has been detected.

This application note provides an additional guide of the Power on Self Test (POST) and Runtime Test (RTT) safety mechanism used to detect faults in the digital elements of the R-Car (SoC) Gen3 and Gen4 devices. As the name implies, the POST is executed at power on time and the RTT is executed during device runtime, and must complete at least once every FTTI. The RTT subtests are executed periodically and designed to achieve medium diagnostic coverage. Renesas SW Solutions for executing RTT have been developed for smooth integration into customer systems; it is intended for fault detection in the R-Car Application CPU cores and HW Accelerators. RTT dependencies on the OS and impact to System Operations are also explained to hasten product understanding.

This document must be used along with the documents mentioned in the Reference section.

Target Devices

R-Car V3H, V3M, V4H, V4M, S4

Note: POST and RTT technology and principals are implemented as well in other R-Car Gen3 / Gen3e series. For better readability above target devices have been selected for this application note.

Abbreviations

Terms	Definitions
POST	Power On Self Test
RTT	Runtime Test
FTTI	Fault Tolerant Time Interval
CPU	Central Processing Unit
A5x	ARM Cortex-A5x core
HWA	Hardware Accelerator
L2 Cache	Level 2 Cache
SRAM	System RAM
ROM	Read-only Memory
DRAM/DDR RAM	Double Data Rate Random Access Memory
ECM	Error Control Module
ELx	Exception Level x (x=0-3)
BIST	Built-in Self-test
FBIST	Field BIST
LBIST	Logic BIST
MBIST	Memory BIST
FBA	Field BIST Activator
FBC	Field BIST Controller
FBJ	Field BIST JTAG-sequencer
UDF	User Defined Function
SGI	Software-generated Interrupt
WFI	Wait for Interrupt
DSP	Digital Signal Processor
PSCI	Power State Coordination Interface
SMC	Secure Monitor Call. SMC is an Arm assembler instruction that causes an exception that is taken synchronously into EL3
FIQ	An interrupt that is set to group 1 by INTC-AP
PMR	CPU Private Mask Register
APMU	Advanced Power Management Unit
IMP	Image Processing Unit
IMR	Image Renderer
ISP	Image Signal Processor
RFSO	Renesas Failure-safe Detection Output

SoC	System On Chip
HW	Hardware
SW	Software
OS	Operating System
OSAL	Operating System Abstraction Layer
HWA	Hardware Accelerator (mostly referring to ISP, IMP, IMR, Vision IP)
SAN	Safety Application Note
SRS	Safety Requirement Specification
SDK	Software Development Kit
AoU	Assumptions of Use
UM	User Manual
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
FuSa	Functional Safety
FMEDA	Failure Modes Effects and Diagnostic Analysis
SPFM	Single Point Fault Metric
LFM	Latent Fault Metric

Table of Contents

1. Introduction

1.1 Scope of the Document

The two main tests to be discussed in this document are Power-on Self-Test (POST) and Runtime Test (RTT). They are both safety mechanisms required for the Renesas R-Car SoCs.

1.2 Why are POST and RTT Needed

There are various safety mechanisms on the device. Self-test is one of the safety mechanisms to detect faults of the elements on the SoC. POST and RTT are both types of self-test to detect faults of the elements on the SoC.

Faults shall be detected before and during the SoC is running. The difference between POST and RTT is that:

- POST is performed at power on, before the SoC is running.
- RTT is performed periodically during the SoC runtime.

1.3 What's the Test Target of POST and RTT

The SoC contains various HW elements: CPUs, hardware accelerators, memories, etc. The test target of POST and RTT are the selected elements on the SoC. In **Chapter 1.4**, the target elements of each test are given. POST and RTT aims to test the faults of these selected elements on the SoC.

The term "hierarchy" refers to a combination of HW elements on the SoC. For example, the 530 hierarchy on V3H refers to the Cortex-A53 CPU core 0 and associated memories. The reason to group HW elements into hierarchies is to run the tests in segments – test for one hierarchy, back to normal operation, then test the next hierarchy. It can take several milliseconds to test everything. Therefore, if the tests do not run in segments, the HW elements (CPU, HWA, etc.) will be in stop mode for too long. The hierarchies of POST and RTT can be different. The hierarchies of different devices can also be different. For the hierarchies of POST and RTT of each different device please refer to [10] Runtime Test Pattern List R-Car X3x and [11] Runtime Test Pattern List R-Car X4x.

1.4 Terms that Describe How the Tests Are Categorized

There are several terms that the Renesas user manuals and safety application notes use when referring to different categories of tests, such as POST, RTT, A1/A2 test, LBIST, MBIST, etc. In this chapter, the terms are explained based on how the tests are categorized. The relationships between the terms are also explained in this chapter.

1.4.1 Categorized Based on When the Test Is Performed

The tests can be categorized based on the timing when the self-test is performed:

- Power-on Self-Test (POST), performed before the SoC is running
- Runtime Test (RTT), perform periodically when the SoC is running

1.4.1.1 Power-on Self-Test (POST)

The POST is a safety mechanism to detect faults before the SoC is used. This safety mechanism is executed only once when power is supplied to the SoC. Some examples of the target hierarchies are: TOP (TOP Hierarchy), VIO (Video IO Hierarchy), ISP0 (Video IO Hierarchy (ISP0)), etc.

1.4.1.2 Runtime Test (RTT)

The Runtime Test is a safety mechanism to detect faults during the SoC runtime and should be periodically executed by software (at least once per FTTI). The target elements will be explained in the next chapters.

1.4.2 Runtime Test - Categorized Based on the Target Element (i.e., Which Hierarchy to Be Tested)

For Runtime Test (RTT), the tests can be categorized based on the elements to be tested:

- CPU Runtime Test (CPU RTT), to test CPU sub-system
    - A1 Test, to test the CPU cores within the CPU sub-system
    - A2 Test, to test L2 (Gen 4)/L3 (Gen 4) Cache within the CPU sub-system
- Hardware Accelerator Runtime Test (HWA RTT), to test the hardware accelerators

In the sub-chapters below the details of each test are explained.

1.4.2.1 CPU Runtime Test (CPU RTT)

The CPU Runtime Test aims to detect faults of the CPU subsystem on the SoC. One example of the test target is a CPU sub-system without dual core lock step implementation.

(a) A1 Test

The CPU RTT A1 Test aims to detect faults of the CPU cores within the CPU subsystem. The target elements are the CPU cores within the CPU sub-system.

(b) A2 Test

The CPU RTT A2 Test aims to detect faults of the L2 (Gen 4)/L3 (Gen 4) Cache within the CPU subsystem. The target element is the L2 (Gen 4)/L3 (Gen 4) Cache within the CPU sub-system.

1.4.2.2 Hardware Accelerator Runtime Test (HWA RTT)

The HWA Runtime Test aims to detect faults of the Hardware Accelerator subsystems on the SoC. Examples of the target elements are the hardware accelerators such as IMP-X7, IMR-Lx6, Vision IP, ISP etc.

1.4.3 Categorized Based on Test Types

The self-tests can be categorized based on the test types:

- Built-in Self-Test (BIST)
    - Logic Built-in Self-Test (LBIST)
    - Memory Built-in Self-Test (MBIST)
- SCAN test

The table below summarizes the major differences between the different test types:

Table 1-1 Difference between test types

Test Type	Contents
Logic BIST	- Tests the logic section - Uses a random pattern generator, so there is no need to have defined test patterns for each test target - Utilized in the **POST**
Memory BIST	- Tests memory (RAM) section - Uses an algorithm test pattern generator to achieve fault coverage - Utilized in both **POST** and **RTT**
SCAN Test	- Tests the logic section - Can achieve medium fault coverage (90% or more), requiring a defined test pattern for each test target - Utilized in the **RTT**

1.4.3.1 Built-in Self-Test (BIST)

A Built-in Self-test is a term inclusive of any type of integrated testing (i.e., not requiring external stimulus or I/O) that can be used for detecting faults in circuitry. There are two sub types of BIST depending on the test target: Logic Built-in Self-Test (LBIST) and Memory Built-in Self-Test (MBIST).

(a) Logic Built-in Self-Test (LBIST)

The Logic Built-in Self-Test (LBIST) aims to test logics. LBIST Uses a random pattern generator. There is no need for the system integrator to input test patterns for each test target.

(b) Memory Built-in Self-Test (MBIST)

The Memory Built-in Self-Test (MBIST) aims to test memories. MBIST uses an algorithm test pattern generator to achieve fault coverage. There is no need for the system integrator to input test patterns for each test target.

1.4.3.2 SCAN test

SCAN testing is a common method for basic fault detection in semiconductor circuitry. The test targets for SCAN testing are logics. For any logic that is the target of SCAN testing, test control signals and test pattern(s) are input; the circuitry is chained together without regard to the target's function, for the purpose of toggling a percentage of the gates so that an expected output pattern(s) can be checked. The output pattern is compared against the expected sequence and a fault is flagged if the comparison fails. See **Figure 1-1** for a basic image of SCAN test implementation.

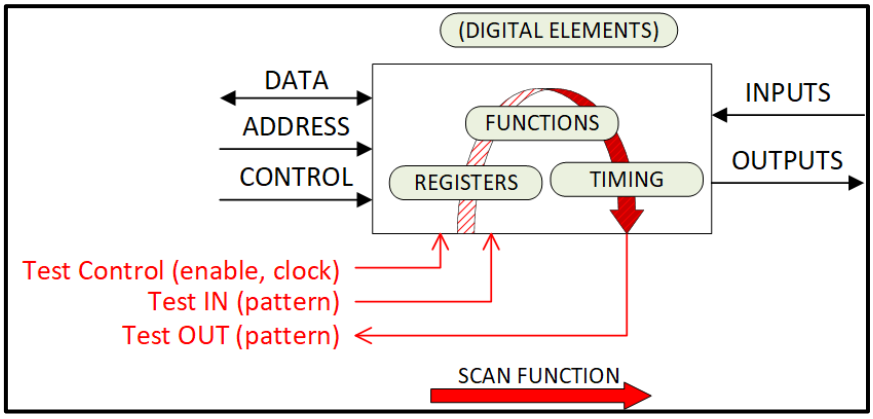

Figure 1-1: SCAN Test Function

With SCAN patterns and chains, a high percentage of faults can be detected. On the other hand, SCAN testing is a destructive test, meaning that register and memory values and other latched signals are destroyed during the test and some efforts are required to restore the target logic to its pre-SCAN state.

1.5 Overview of POST and RTT Implementation and Integration

POST and RTT **shall** be enabled as described in HW SAN and HW SRS:

[4] R-Car Series, 3rd Generation Safety Requirement Specification

[5] R-Car Series, 3rd Generation Safety Application Note

[8] R-Car Series, 4th Generation Safety Requirement Specification

[9] R-Car Series, 4th Generation Safety Application Note

The procedures to enable and execute POST and RTT are described in the POST and RTT chapters in the HW SAN. The system integrators shall follow the "Operations" chapter in the above documents to configure, execute, check the test results and perform fault control.

The POST and RTT are supported by Renesas hardware on the SoC. **Chapter 3** gives detailed information about the hardware modules that support POST and RTT.

POST is not supported by Renesas software. The reason is that POST is executed automatically after PRESET release. RTT is supported by a few Renesas software. Details of these software products can be found in **Chapter 4**.

2. Renesas Hardware Support for POST and RTT

Both POST and RTT are performed using the Field BIST (FBIST) hardware modules.

The Field BIST modules are constituted by one of field BIST controller (FBC) and each hierarchy of the field BIST activators (FBAs), JTAG-sequencer (FBJs).

The field BIST modules work together to provide the POST and RTT functions such as test configuration, test activation, test execution, notify test finish, report errors to ECM (error control module), etc.

Figure 2-1 shows the block diagram of FBIST modules.

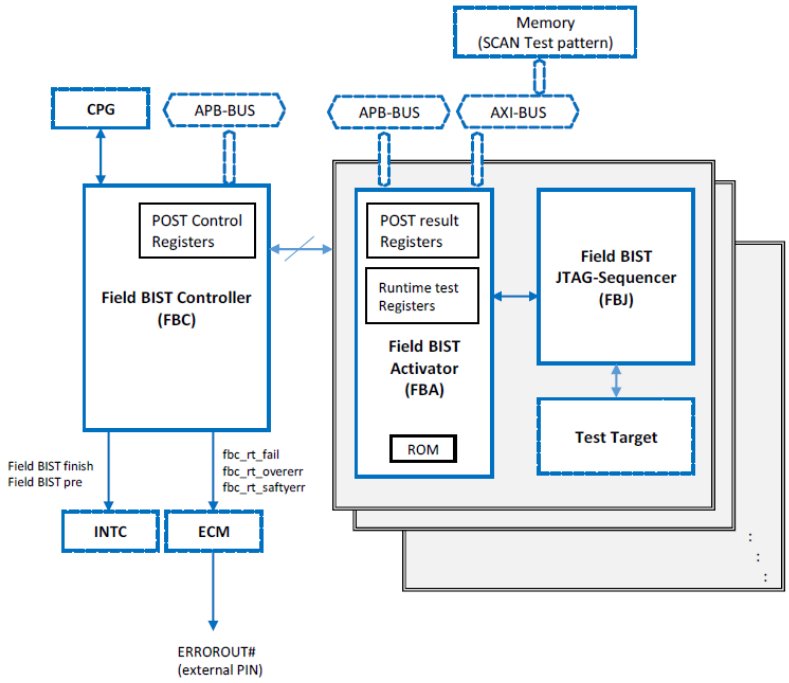

Figure 2-1 FBIST modules Block Diagram

For the detailed function, register configuration, register description please refer to the following document:

[6] R-Car Series, 3rd Generation Hardware Description for Functional Safety HD-FUSA

[7] R-Car Series, 4th Generation Hardware User's Manual

3. Renesas Software Support for RTT

Renesas Software Solution to support Runtime Test includes the following SW components:

- CPU RTT Driver
- HWA RTT Library
- Field BIST Driver
- RFSO Driver
- Secure Monitor

As mentioned before POST does not require special Renesas software. The reason is that POST is executed automatically after PRESET release. The results are reflected in a register and shall be checked by user software.

CPU RTT driver and HWA RTT library provide APIs for executing CPU Runtime Test and HWA Runtime Test to detect permanent faults of the AP system core and HW accelerators during runtime.

The Field BIST driver supports the CPU RTT driver and HWA RTT library to access the Field BIST hardware modules. The Field BIST driver needs support from secure monitor to access the registers which have security privilege (i.e., that can only be accessed in EL3 (exception level 3)).

The RFSO driver accesses RFSO hardware and provides interval timer API and time-out detection API. When user starts the RFSO interval timer, upon the timer expiry an SPI interrupt is triggered. In the Interrupt handler, Callback function provided by RFSO driver is invoked. This callback function shall be used to invoke the Runtime Test periodically and the time-out detection API can be used to monitor time-out of RTT.

Secure Monitor provides the interface to execute Runtime Test by managing the switches between the different ELs (Exception levels). It also supports PSCI function and provides the interfaces to Smoni (Secure monitor) APIs. When an SMC is generated, the Exception Handler of Secure Monitor determines whether it is necessary to switch the EL from the SMC function identifier. When it executes the runtime test, Secure monitor saves and restores data for returning to normal operation.

Figure 3-1 and **Figure 3-2** provide block diagram and layer representation of RTT SW.

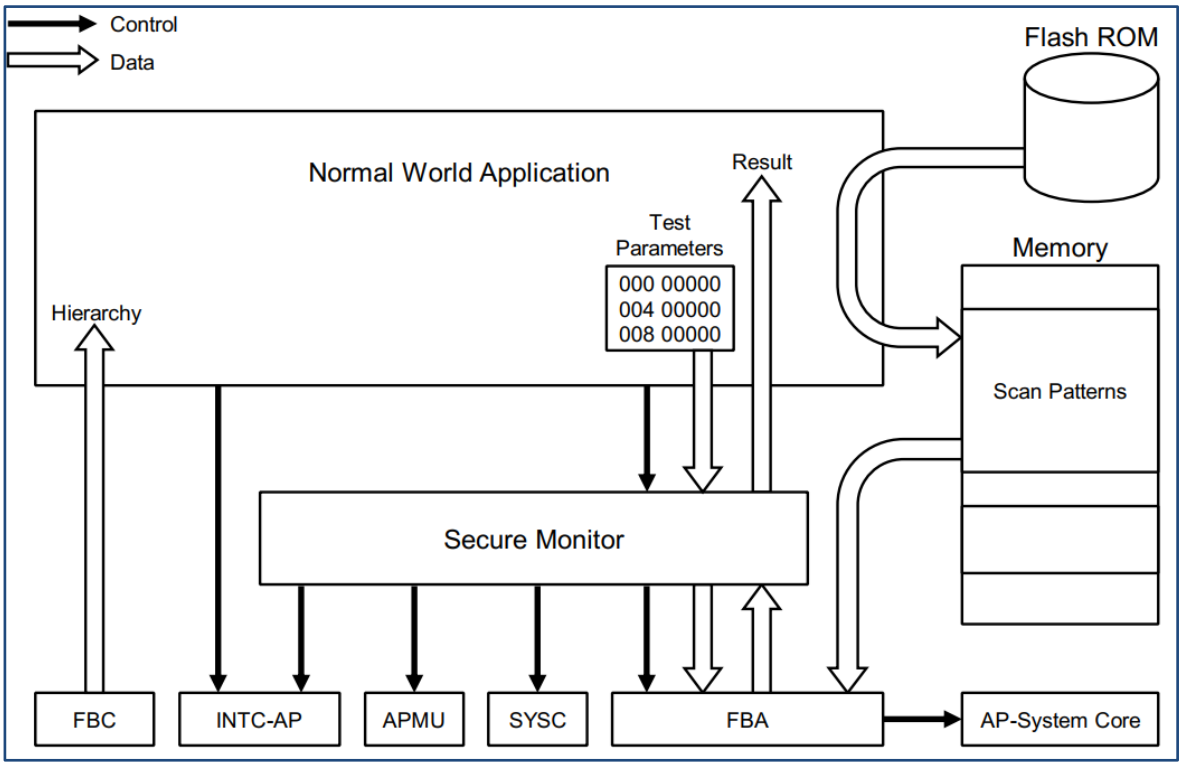

Figure 3-1: Block diagram representation of Control & Data flow of Runtime Test

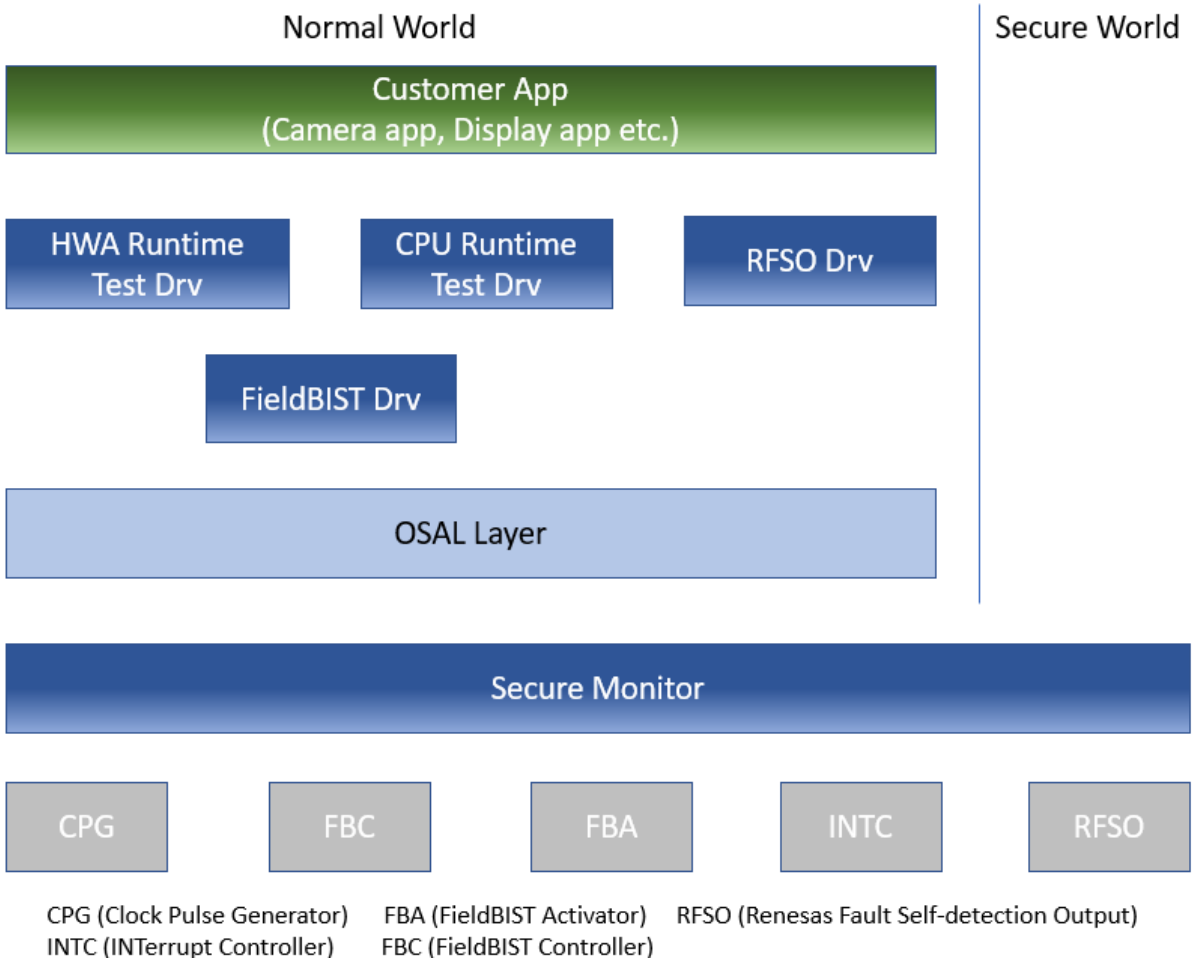

Figure 3-2: Layer representation of Renesas software supporting RTT

Note: Field BIST driver also accesses some HW directly, without using OSAL.

For details of how the software shall be integrated, please refer to the following software documents:

[12] R-Car V3M/V3H CPU Runtime Test Driver Software Users Manual

[13] R-Car V3M/V3H CPU Runtime Test Driver Software Safety Application Note

[14] R-Car V3M/V3H Field BIST driver Software Users Manual

[15] R-Car V3M/V3H Field BIST driver Software Safety Application Note

[16] R-Car V3M/V3H RFSO driver Software Users Manual

[17] R-Car V3M/V3H RFSO driver Software Safety Application Note

[18] R-Car V3M/V3H HWA Runtime test driver User's Manual: Software

[19] R-Car V3M/V3H/V3H2 HWARTT Driver Software Safety Application Note

[20] R-Car V3M/V3H Secure Monitor Software Users Manual

[21] R-Car V3M/V3H Secure Monitor Software Safety Application Note

[22] R-Car V4H/V4M CPU Runtime Test Driver Software Users Manual

[23] R-Car V4H/V4M CPU Runtime Test Driver Software Safety Application Note

[24] R-Car V4H/V4M Field BIST driver Software Users Manual

[25] R-Car V4H/V4M Field BIST driver Software Safety Application Note

[26] R-Car V4H/V4M RFSO driver Software Users Manual

[27] R-Car V4H/V4M RFSO driver Software Safety Application Note

[28] R-Car V4H/V4M HWA Runtime test driver User's Manual: Software

[29] R-Car V4H/V4M HWARTT Driver Software Safety Application Note

[30] R-Car V4H/V4M Secure Monitor Software Users Manual

[31] R-Car V4H/V4M Secure Monitor Software Safety Application Note

[32] R-Car S4 CPU Runtime Test Driver Software Users Manual

[33] R-Car S4 CPU Runtime Test Driver Software Safety Application Note

[34] R-Car S4 Field BIST driver Software Users Manual

[35] R-Car S4 Field BIST driver Software Safety Application Note

[36] R-Car S4 RFSO driver Software Users Manual

[37] R-Car S4 RFSO driver Software Safety Application Note

[38] R-Car S4 Secure Monitor Software Users Manual

[39] R-Car S4 Secure Monitor Software Safety Application Note

The assumption of uses (AoU) defined in the software SAN shall be fulfilled by system integrator.

3.1 Renesas RTT Software Availability

The table below indicates availability of RTT SW for R-Car devices.

Table 3-1: RTT SW Availability per R-Car Device

SW Release	Device	QNX	Linux with ARM compiler
HW RTT driver	V3M	✓	✓
	V3H 1.1	✓	✓
	V3H 2.1	✓	✓
	V4H/V4M	✓	✓
	S4	✓	×
CPU RTT driver	V3M	✓	✓
	V3H 1.1	✓	✓
	V3H 2.1	✓	✓
	V4H/ V4M	✓	✓
	S4	✓	×
Field BIST driver	V3M	✓	✓
	V3H 1.1	✓	✓
	V3H 2.1	✓	✓
	V4H/ V4M	✓	✓
	S4	✓	×
RFSO driver	V3M	✓	✓
	V3H 1.1	✓	✓
	V3H 2.1	✓	✓
	V4H/ V4M	✓	✓
	S4	✓	×
Secure Monitor	V3M	✓	✓
	V3H 1.1	✓	✓
	V3H 2.1	✓	✓
	V4H/ V4M	✓	✓
	S4	✓	×

Note: Renesas R-Car SDK package support includes above devices. If the RTT is needed for other devices, please contact Renesas Sales.

4. Runtime Test Software Integration Guide

This chapter will provide extra guidance in addition to the user documents given in **Chapter 3**. The additional information incudes precautions, constraints, system impacts, integration guidance etc. Renesas recommends to start planning of integrating RTT as early as possible in the system; to avoid conflict on the performance goals.

4.1 Precautions and constraints for CPU RTT

R-Car Gen3 and Gen4 devices contain some combination of AP (application platform) cores (CPU) that must be supported by the RTT safety mechanism**. Figure 4-1** and **Figure 4-2** show RTT execution sequence for Gen 3 platform (V3H, V3M, etc.) and Gen 4 platform (V4H, V4M, S4, etc). Some adjustments are needed for each device. For example, for V3H the Cortex-A57 components shall be ignored; for V3M, the Cortex-A57 and CPU2/CPU3 of Cortex-A53 shall be ignored. RFSO is a special function in R-Car, consisting of interval and timeout timers, that is used for trigger and timeout detection of RTT. The RTT test operation for each core is indicated by the white segments; for example, A57 CPU0 RTT is tested in time slice marked with 570. Time slice marked with 57D corresponds to A2 test.

Constraints:

- The target CPU stops during A1 test
- Simultaneous operation of the CPU and RTT within a cluster is not possible during A2 test. All CPU's in a cluster stop during A2 test
- Parallel execution of A1 test and A2 test within a cluster is not possible.

  Parallel execution of A1 test is possible if target CPU's belong to different clusters (Software implementation is not available yet. Software implementation for V4H is planned.)

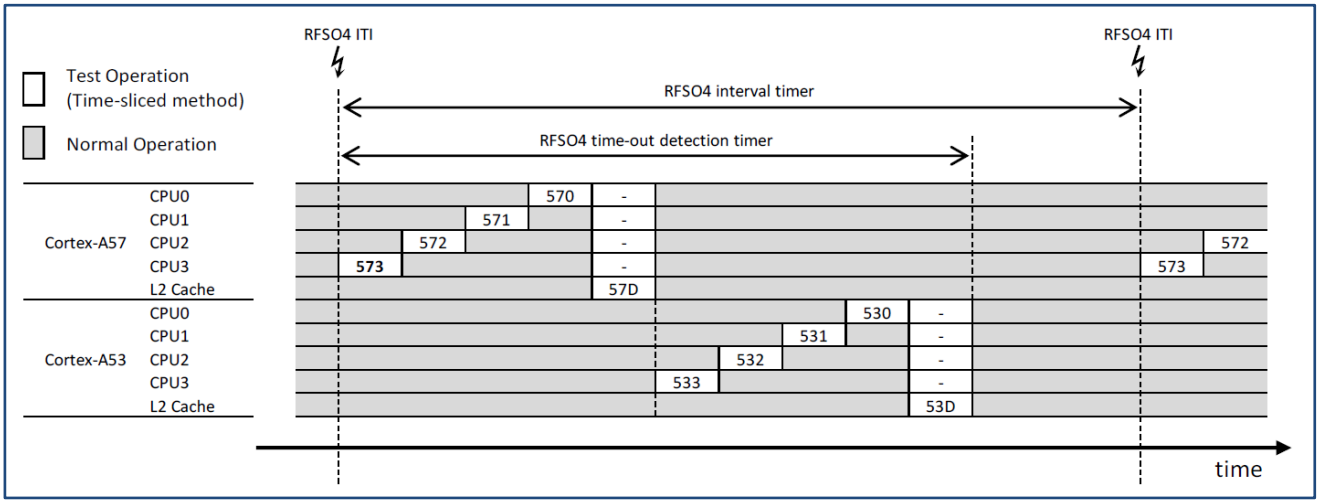

Figure 4-1: Timing Chart of RTT for Gen 3 AP-System Core

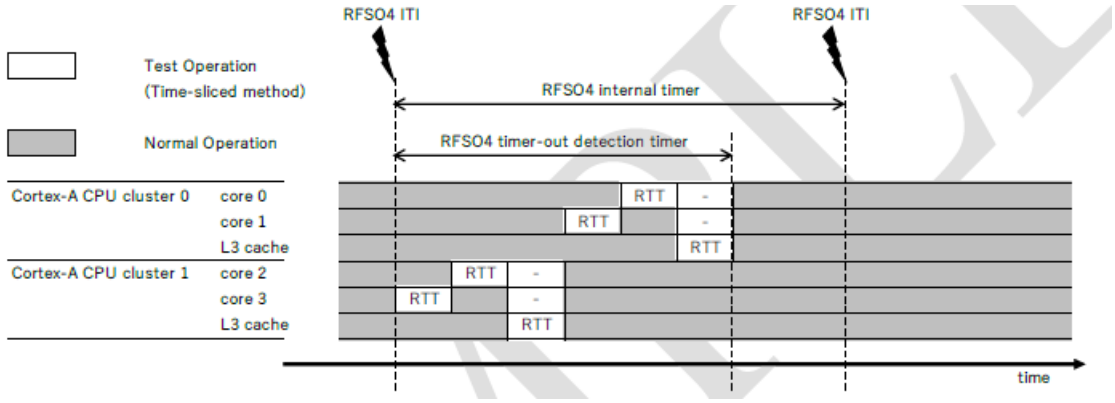

Figure 4-2 Timing Chart of RTT for Gen 4 AP-System Core

4.2 Interrupt Involvement in CPU RTT

When the RTT execution has completed, Field BIST finish interrupt signal is asserted. The Field BIST driver uses Field BIST finish interrupt to handle the completion of Runtime Test.

- In A1 test, the target CPU core must continue sleeping during RTT. Therefore, interrupts of the CPU core are masked during execution of RTT, so as not to wake up the target CPU core. A CPU Core which is not the A1 test target receives Field BIST finish interrupt and wakes up the target CPU Core.
- In A2 test, all CPU cores in the target cluster need to be masked from all interrupts except the Field BIST finish interrupt which wakes up CPU0 in the target cluster. CPU Core0 receives Field BIST finish interrupt and wakes up the other CPU Cores in the cluster.

The method to wake up the CPU Core is different between R-Car Gen3 and R-Car Gen4. In both cases, the masked interrupts which has occurred during CPU RTT are processed after CPU RTT.

[For R-Car Gen3]

For A1 test, a core not under test will receive the field BIST finish interrupt when the RTT is finished. Field BIST driver on this core uses an SGI to wake up the CPU Core under test. To fulfill this function, the following shall be done:

- Before entering A1 test, on the core receiving field BIST interrupt, the field BIST finish interrupt shall be temporarily raised to the highest priority, to ensure the field BIST interrupt is processed immediately.
- Before entering A1 test, on the CPU Core under test, the SGI is raised to have the highest priority and all the other interrupts are masked except for the SGI. This can ensure the core not to be woken up by any interrupt except for the SGI.
- After the A1 test is finished and all cores are awake, the interrupts are restored to the priorities before RTT.

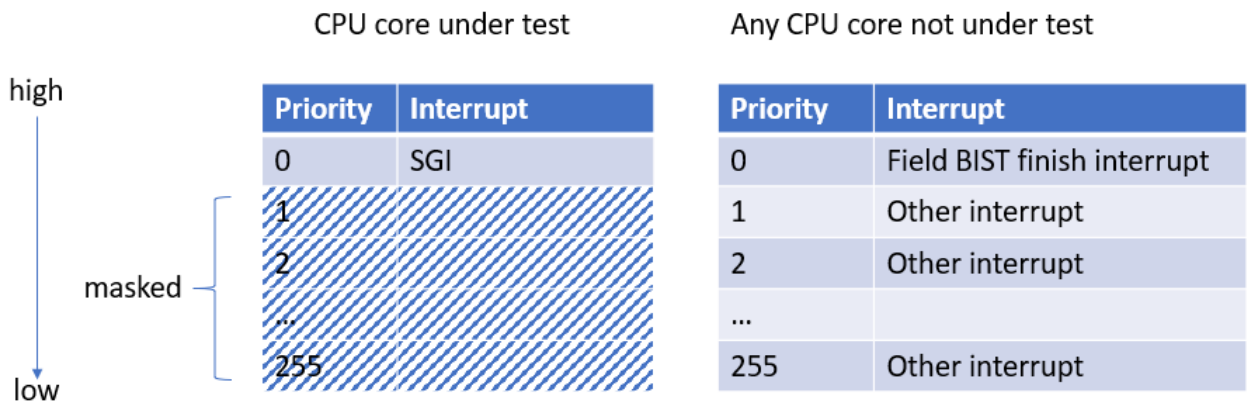

Figure 4-3 Interrupt priority setting during CPU RTT A1 test

For A2 test, CPU core 0 will receive the field BIST finish interrupt when the RTT is finished. The Field BIST driver on core 0 uses an SGI to wake up the other CPU Cores. To fulfill this function, the following shall be done:

- Before entering A2 test, on core 0, the field BIST finish interrupt shall be temporarily raised to the highest priority to ensure the field BIST interrupt is processed immediately.
- Before entering A2 test, on the other CPU Cores, the SGI shall be raised to have the highest priority and all the other interrupts are masked except for the SGI. This can ensure the core not to be woken up by any interrupt except for the SGI.
- After the A2 test is finished and core is awake, the interrupts shall be restored to the priorities before RTT.

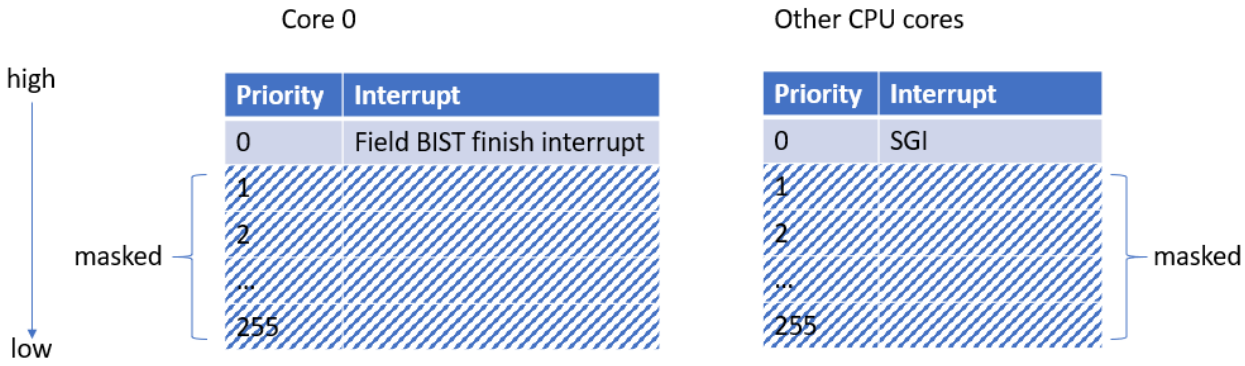

Figure 4-4 Interrupt priority setting during CPU RTT A2 test

[For R-Car Gen4 V4H/V4M, S4]

For Gen 4 devices, the Field BIST driver uses the APMU, instead of the SGI, to wake up the CPU Core. The Field BIST driver configures the CPU so that it can only be woken up by the APMU during the CPU RTT.

The procedure becomes:

For A1 test, a core not under test will receive the field BIST finish interrupt when the RTT is finished. Field BIST driver on this core triggers APMU to wake up the CPU Core under test. To fulfill this function, the following shall be done:

- Before entering A1 test, on the core receiving field BIST interrupt, the field BIST finish interrupt shall be temporarily raised to the highest priority, to ensure the field BIST interrupt is processed immediately.
- Before entering A1 test, on the CPU Core under test, all the interrupts shall be disabled
- After the A1 test is finished and all cores are awake, the interrupts are enabled again

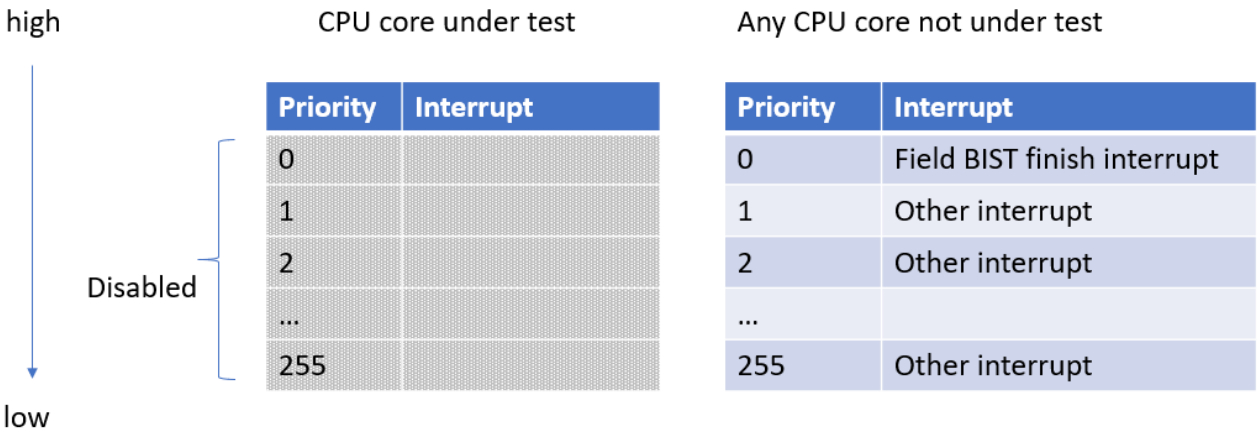

Figure 4-5 Interrupt priority setting during CPU RTT A1 test

For A2 test, CPU core 0 will receive the field BIST finish interrupt when the RTT is finished. The Field BIST driver on core 0 triggers APMU to wake up the other CPU Cores. To fulfill this function, the following shall be done:

- Before entering A2 test, on core 0, the field BIST finish interrupt shall be temporarily raised to the highest priority to ensure the field BIST interrupt is processed immediately.
- Before entering A2 test, on the other CPU Cores, the interrupts shall be disabled

- After the A2 test is finished and core is awake, the interrupts on CPU core 0 shall be restored to the priorities before RTT. The interrupts on other CPU cores shall be enabled again.

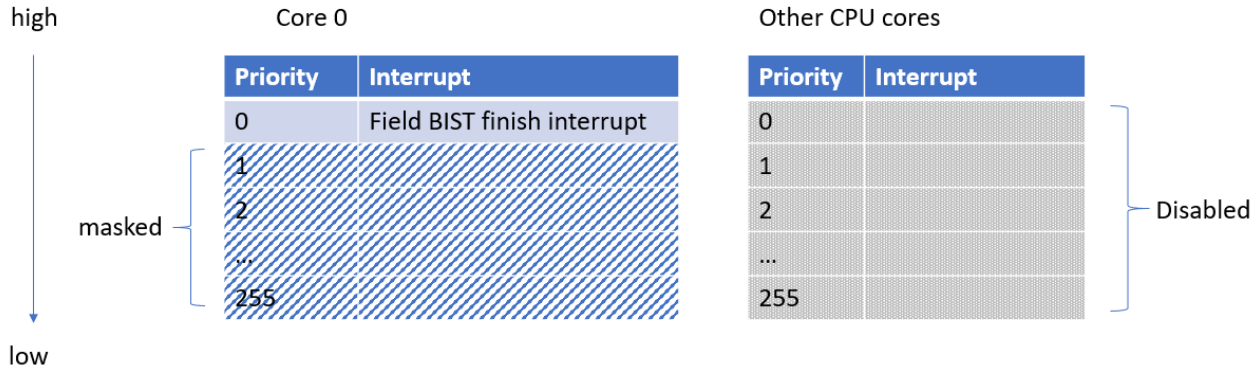

Figure 4-6 Interrupt priority setting during CPU RTT A2 test

4.3 Impact of CPU RTT on Generic Timer

Some registers in the Generic Timer may lose their value during CPU RTT. However, these do not impact the behavior of the timer inside the Generic timer. Only the interrupt handling is delayed if the timer expires during the CPU RTT execution.

Registers which must be backed up/restored by Secure Monitor during CPU RTT are described in the R-Car HW SAN ([5] R-Car Series, 3rd Generation Safety Application Note and [9] R-Car Series, 4th Generation Safety Application Note). These registers lose their value during A1 and A2 RTT. In A1 test, only the target CPU core is backed up and its registers restored. In A2 test, all CPU cores in the cluster are backed up and their registers restored.

Table 4-1 shows the target registers for back-up in the Generic Timer for R-Car Gen3.

Table 4-1: The target registers for back-up in the Generic Timer

Register	Description
CNTKCTL_EL1	Timer Control register (EL1)
CNTFRQ_EL0	Timer Counter Frequency register
CNTP_CTL_EL0	Physical Timer Control register (EL0)
CNTP_CVAL_EL0	Physical Timer CompareValue register (EL0)
CNTV_CTL_EL0	Virtual Timer Control register
CNTV_CVAL_EL0	Virtual Timer CompareValue register
CNTVOFF_EL2	Virtual Timer Offset register
CNTHCTL_EL2	Timer Control register (EL2)
CNTHP_CTL_EL2	Physical Timer Control register (EL2)
CNTHP_CVAL_EL2	Physical Timer CompareValue register (EL2)
CNTPS_CTL_EL1	Physical Secure Timer Control register (EL1)
CNTPS_CVAL_EL1	Physical Secure Timer CompareValue register (EL1)
CNTKCTL_EL1	Timer Control register (EL1)
CNTFRQ_EL0	Timer Counter Frequency register

Physical timer/Virtual timer always work as the System Counter is an always-on device, which provides a fixed frequency incrementing the system count.

The operation of this view of a timer is:

TimerConditionMet = (((Counter[63:0] – Offset[63:0])[63:0] – CompareValue[63:0]) >= 0)

Where:

- TimerConditionMet: Is TRUE if the timer condition for this counter is met, and FALSE otherwise.
- Counter: The physical counter value, that can be read from the CNTPCT_EL0 register.
- Offset: For a physical timer, this value is zero. For the EL1 virtual timer, this value is held in the CNTVOFF_EL2 register. For the Non-secure EL2 and Secure EL2 virtual timer, this value is zero.
- CompareValue: The value of the appropriate CompareValue register, CNTP_CVAL_EL0, CNTHP_CVAL_EL2, CNTPS_CVAL_EL1, CNTV_CVAL_EL0, or CNTHV_CVAL_EL2.

Figure 4-7 shows that the timer expires after CPU RTT, because some registers are not available during CPU RTT. In A1 test, though the cores except for the target CPU cores are running, the timer does not expire because the interrupt is masked. In A2 test, all the CPU cores in the cluster are stopped and therefore, no core is available to compare the Generic timer. After A1 test, the registers are restored and the timer expires.

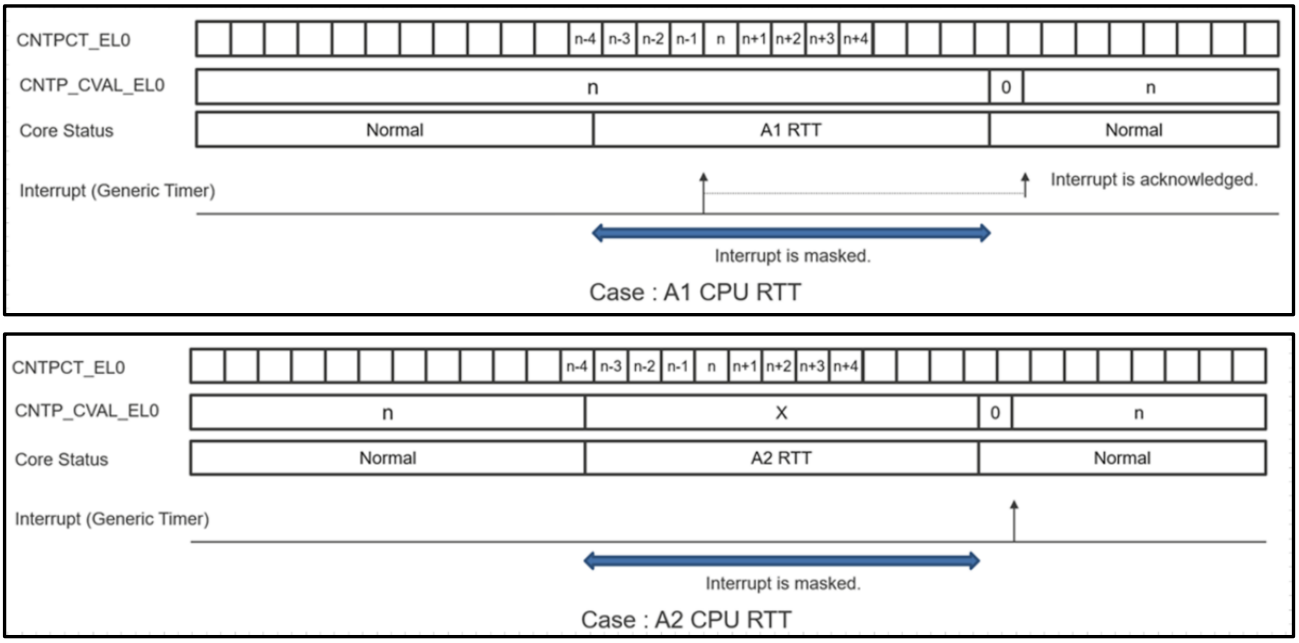

Figure 4-7: Registers of the Generic Timer during CPU RTT (A1 and A2)

4.4 Precautions and constraints for HWA RTT

In order not to immobilize all Hardware Accelerators functionalities during the tests, hierarchies can be tested individually.

Example: Hierarchies of DSP (e.g. DSP0, DSP1, DSP2) can be tested individually in order to prevent immobilization of other DSP functionalities during RTT. When DSP0 is under test, other DSP hierarchies can be used for normal operations. See **Figure 4-8** for an image of this operation.

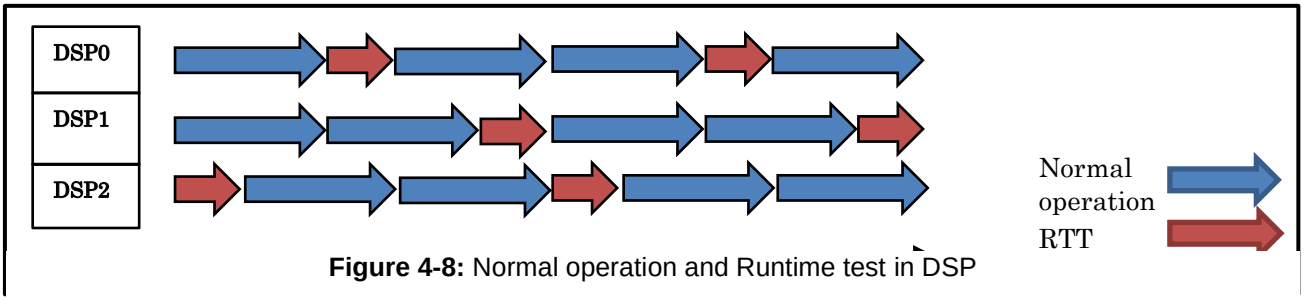

Figure 4-8: Normal operation and Runtime test in DSP

The HWA Runtime Test can realize parallel execution. Groups of hierarchies can be tested together and user can execute RTT for each hierarchy in parallel.

Example: RTT on target hierarchy IMP Ch0 and IMP Ch1 in parallel are possible. See Error! Reference s ource not found.**Figure 4-9** as reference. This is not possible for the CPU.

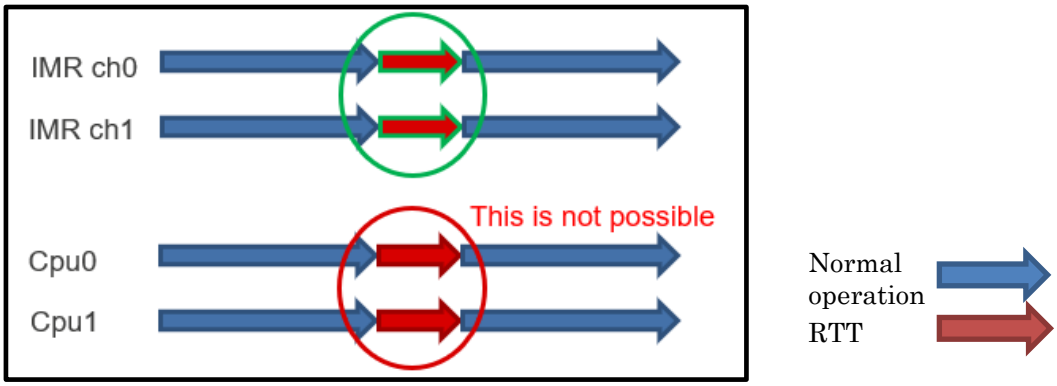

Figure 4-9: HWA RTT vs CPU RTT parallel execution

Normal operation of HWA and its Runtime Test cannot be executed simultaneously. The user cannot execute normal operation and Runtime test on the same hierarchy simultaneously. User application shall stop the main operation to start the Runtime test.

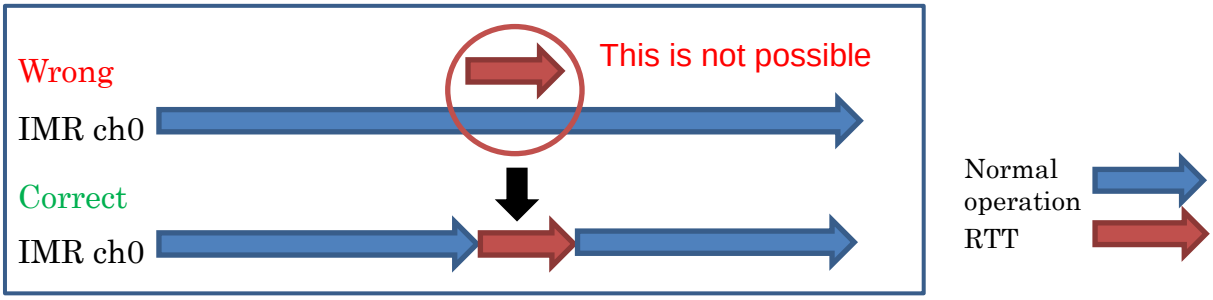

Figure 4-10: HWA RTT and Normal Operation

4.5 Test Patterns

SCAN (Test) Patterns are provided by Renesas Electronics and shall be used during RTT execution. These patterns and register setting values of FBIST modules shall be stored in the external flash ROM. Before executing SCAN test, the patterns and register values shall be transferred from external flash ROM to their respective memory areas: SCAN patterns to DDR memory and register setting values set to relevant registers.

As explained in **Chapter 1.4.3**, the two test types that serve RTT are SCAN Test and MBIST. Most hierarchies have 2 SCAN test, and a single MBIST. However, the number of tests can change. The reasons are explained in below.

For SCAN test:

1) SCAN test uses SCAN patterns and chains to detect a high percentage of faults. The SCAN test is split to multiple tests so that it doesn't occupy the CPU for too long. Each test will have a scan pattern.
2) The number of SCAN test is set to be 2 for most hierarchy, but it can be split to more or less tests depending on implementation decisions during the creation process of the SCAN patterns by Renesas
3)

For MBIST:

1) MBIST uses an algorithm test pattern generator to achieve fault coverage. There is no need for the system integrator to input test patterns for each test target. Therefore, usually the MBIST is run in one test.

For the exact configuration of SCAN test and MBIST for each device please refer to Runtime Test Pattern List R-Car X3x and Runtime Test Pattern List R-Car X4x.

4.5.1 Creating C table-format code from the SCAN Patterns binary

The SCAN patterns and related Register settings are part of the R-Car Hardware Safety Package.
The package includes the SCAN patterns in binary form and the Register settings for each Test mode (Scan test1, Scan test 2 etc.). An example of Register setting details are shown in **Figure 4-11**.

570 Test
(1) SCAN Test 1

Register Name	Abbreviation	Address	Value
Memory information(1)	RTTMI1	H'FF871004	H'000006FB
Memory information(2)	RTTMI2	H'FF871008	(Pattern address)
Expected value of SISR(1)	RTTSISR1	H'FF87100C	H'0008D21A
Up counter of WDT	RTTUC	H'FF871014	H'00028B0A
Low counter of WDT	RTTDC	H'FF871018	H'00000014
Single Step Mode setting register(0)	RTTSET0	H'FF871080	H'FFFFFFFF
Single Step Mode setting register(1)	RTTSET1	H'FF871084	H'00000001
Single Step Mode setting register(44)	Draft	H'FF871130	H'006C0420
Execution option	RTTEX	H'FF871000	H'005A0333

Figure 4-11: Register setting for Runtime test

The binary SCAN Pattern data needs to be inserted into the application source code. It's independent from the Renesas RTT Software release. Especially, the patterns can be updated.

The SCAN pattern shall be stored in the DDR memory as **a unit of 128 Bytes** before starting the RTT. The FBA (Field BIST Activator) loads the SCAN pattern from the address specified within the RTTMI2 register.

For example, the xxd command in Linux translates input binary code into C table-format code.

Following shows the example of C table-format code for SCAN pattern data of V3H.

```
__attribute__((aligned(128))) __attribute__((section(".rtt_sacan_area")))
unsigned char RCV3H_530_D02N01_C20V001[] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
...
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
unsigned int RCV3H_530_D02N01_C20V001_len = 190208;
```

4.6 RTT Execution Time

The Runtime Test has a performance impact on the system. The following time periods must be considered.

- (Software overhead time + Pattern load time (DRAM to FBA) + Hardware execution time) for each FTTI.

- Software overhead time: Pre/Post Processing time of Runtime Test + scheduling process time of operating system.
- Pattern load time (DRAM to FBA) + Hardware execution time: Refer to Reference [Runtime Test Pattern List].

- To support continuous execution during the CPU Runtime Test, applications running on the target CPU can be transferred to another CPU if the OS supports this feature, but it will cause additional software overhead for the scheduling process of the OS.

Please refer to attachment “RunTimeTest_AppNote_RTTExecTiming.xlsx” in this PDF for Runtime Test Timing information for different device and OS.
Note: Some of the SW timings in the table are negative values, because the HW times are theoretical values.

4.7 Usage of RFSO Timers

The RFSO driver provides APIs for the RFSO interval timer and the time-out detection timer. The RFSO driver has the following APIs:

Table 4-2: RFSO Driver APIs

API Name	Purpose
drvRFSO_IntervalTimerStart	This API starts interval timer.
drvRFSO_IntervalTimerStop	This API starts time out detection timer.
drvRFSO_TimeoutTimerStart	This API stops interval timer.
drvRFSO_TimeoutTimerStop	This API stops time out detection timer.
drvRFSO_SetOutputPin	This API controls the RFSO output pins.
drvRFSO_CheckConfigReg	This API detects faults of the target configuration registers.

The interval timer in RFSO is used to trigger the Runtime Test and the time-out detection timer in RFSO is used to detect a time-out of the RTT execution (to detect whether test execution time is out of the tolerant range, i.e., within the Minimum execution time and Maximum execution time). The timers in other channels of RFSO are used for different hierarchies as shown below for V3H device.

Table 4-3: RFSO Channel Usage (V3H)

Third Generation R-Car Series Products	Channel	Usage
R-Car V3H	RFSO 0	Periodical Checks for Arm Realtime Core
	RFSO 1	Runtime Test for Vision IP
	RFSO 2	Periodical Checks for AP-System Core (Except for Runtime Test)
	RFSO 3	Time-sliced method of Runtime Test for IMP-X5
	RFSO 4	Runtime Test for AP-System Core
	RFSO 5	Time-sliced method of Runtime Test for AP-System Core
	RFSO 6	Reserved
	RFSO 7	Reserved
	RFSO 8	Runtime Test and Internal Bus Interface Check for IMP-X5
	RFSO 9	Runtime Test and Internal Bus Interface Check for IMR-LX4
	RFSO 10	Time-sliced method of Runtime Test for IMR-LX4

User shall call API mentioned above (first entry in Error! Reference source not found.) to start the RFSO i nterval timer, and when RFSO interval timer expires, shared peripheral interrupt is triggered. The RFSO

Callback function [*drvRFSO_TimerExpirateCb_t*] in the interrupt handler shall be used to invoke the Runtime Test periodically. Refer to **Figure 4-12** for a visual guide on RFSO timer usage.

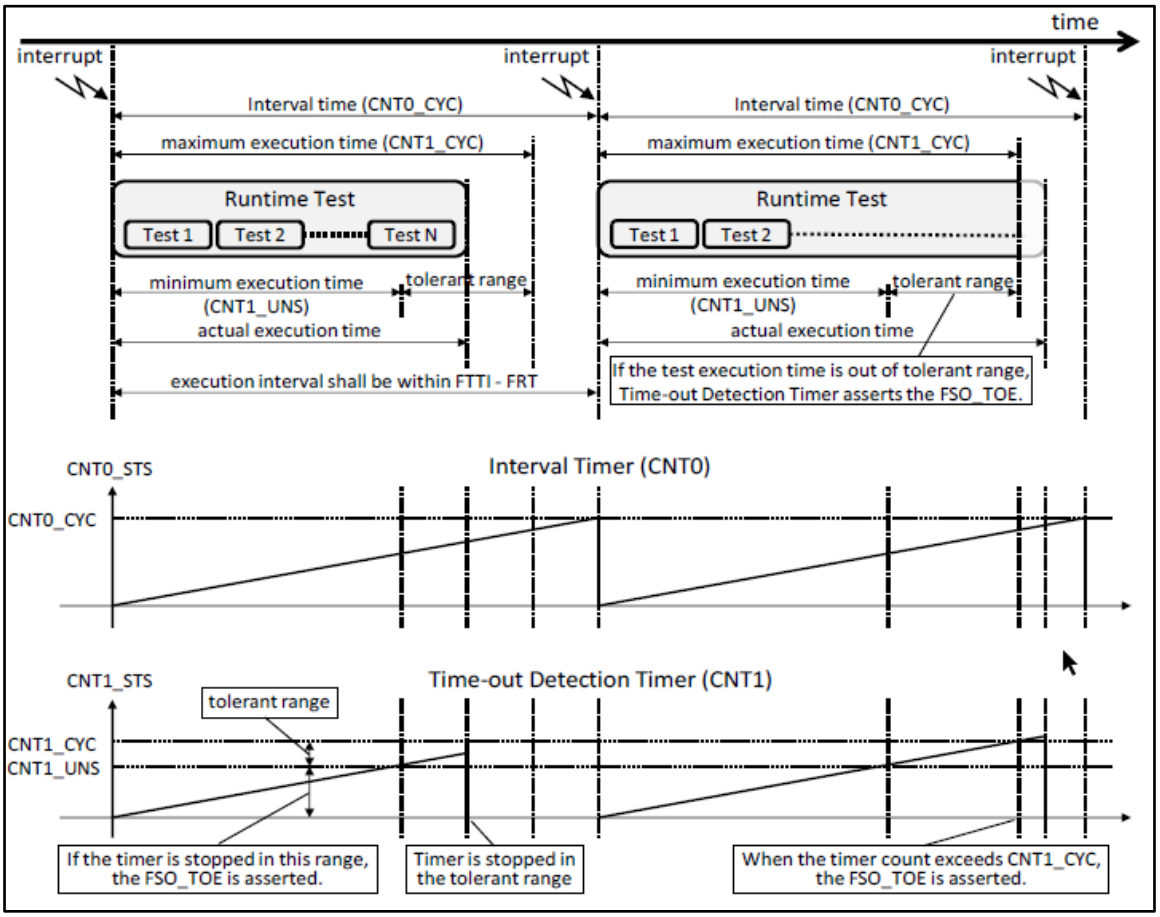

Figure 4-12: Usage of RFSO interval timer and time-out timer

Refer to the R-Car HW SAN ([5] R-Car Series, 3rd Generation Safety Application Note and [9] R-Car Series, 4th Generation Safety Application Note) for the details on the usage of the RFSO Timer Channels for the CPU and HWA Runtime Test.

As the timers of the RFSO are used to periodically execute the RTT, user shall make sure that all the test patterns are executed once within the FTTI. The following **Figure 4-13** shows an example of the timing chart of Runtime Tests. In the timing chart, RFSO4 is used for FTTI and RFSO5 is used for each test pattern.

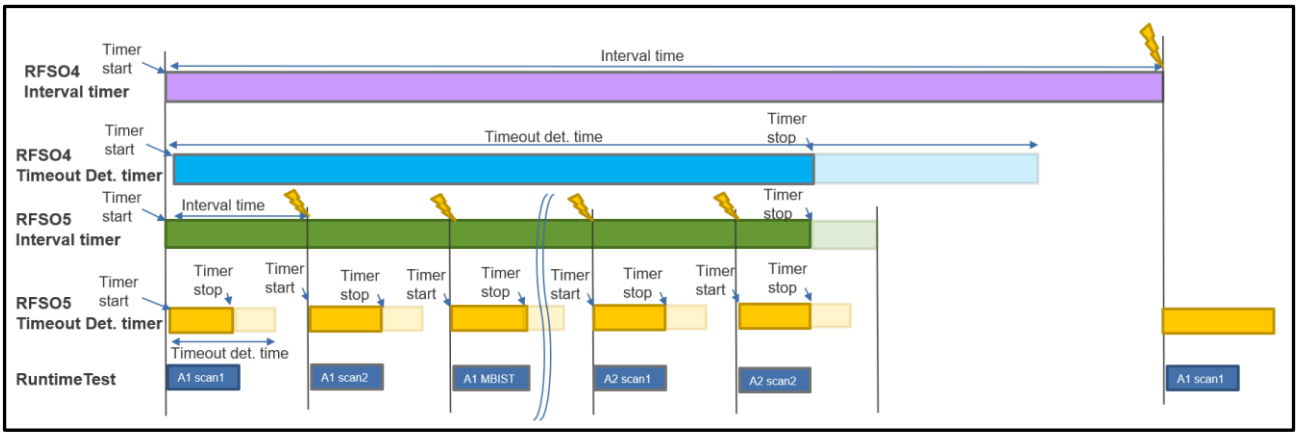

Figure 4-13: Timing chart of Runtime Test

4.8 User Defined Functions (UDFs) in Field BIST Driver

The field BIST driver provides UDF functions for the user to be able to customize the code in UDF. There are two primary reasons to provide UDF functions:

1. To allow user to specify user specific routines in UDF.
2. To allow user to make OS specific calls. The field BIST driver needs to call OS APIs. Some OS APIs are called through OSAL. OSAL is a Renesas abstraction layer software that abstracts OS specific APIs to a common API. However, OSAL abstracts only part of the OS APIs. The field BIST driver also utilizes some OS APIs that are not abstracted by OSAL. Therefore, the field BIST driver has to directly call some OS APIs. The UDF allows the user to call the OS API depending on which OS is used in the system.

[For R-Car Gen 3 and R-Car Gen 4 V4x]

The SW architecture involving the field BIST driver is different between different OSs:

- For Linux, the “field BIST driver” module will call the “UDF for field BIST” module, then the “UDF for field BIST” module will call the “FieldBIST for using kernel API” module which is integrated in the OS. The reason is that EL1 is required to call the FuSa API for Secure monitor on Linux.
- For QNX, the “field BIST driver (User lib)” module will call the “UDF for field BIST” module, then the “UDF for field BIST” module will call the “FieldBIST Driver (process)” module. The “FieldBIST Driver (process)” module is implemented as privileged process separated from the user application, therefore it can call the FuSa API for Secure monitor.

Figure 4-14 and **Figure 4-15** show the system configurations for Linux and QNX.

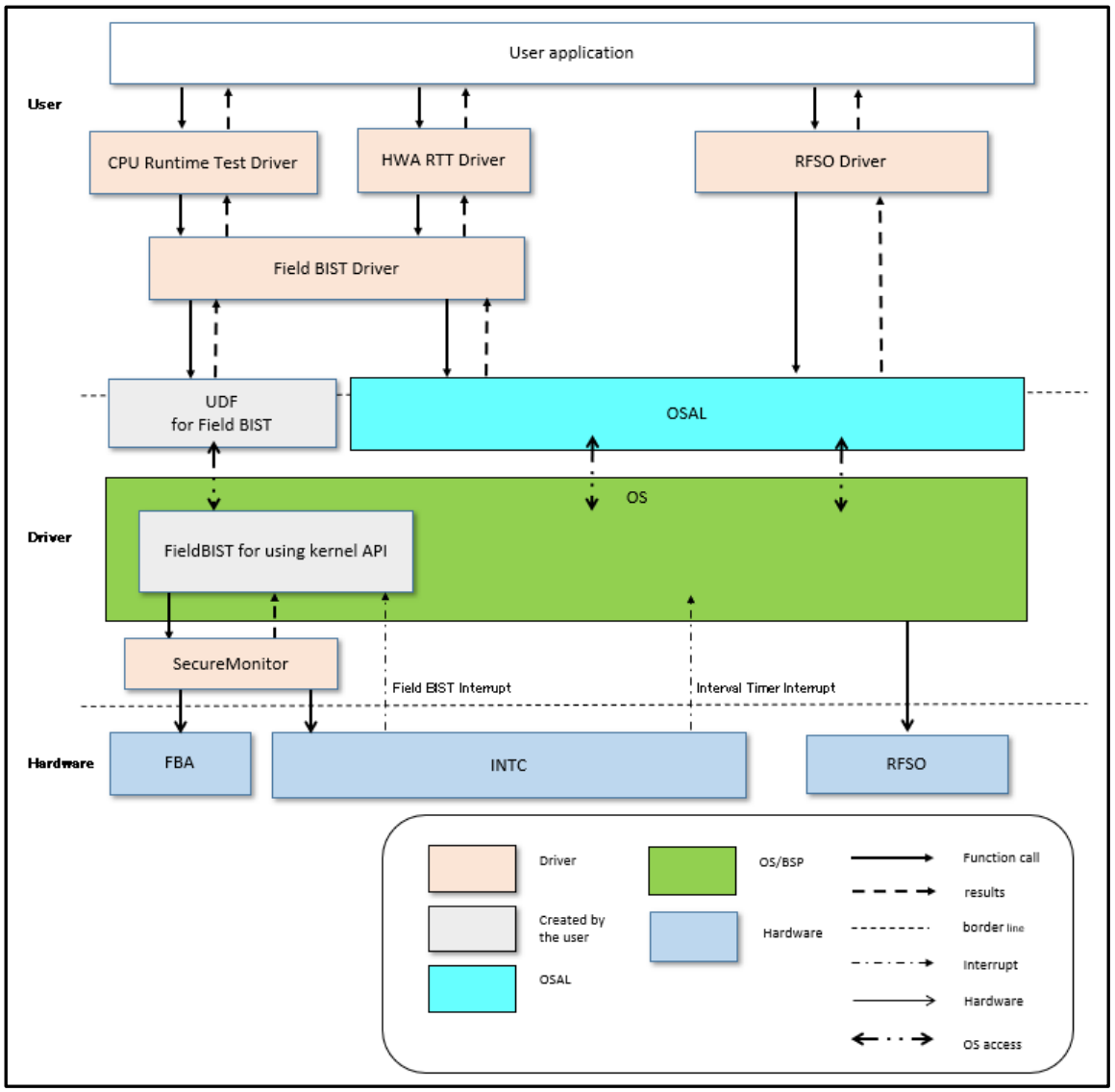

Figure 4-14: System configuration for Gen 3 and Gen 4 V4x Linux

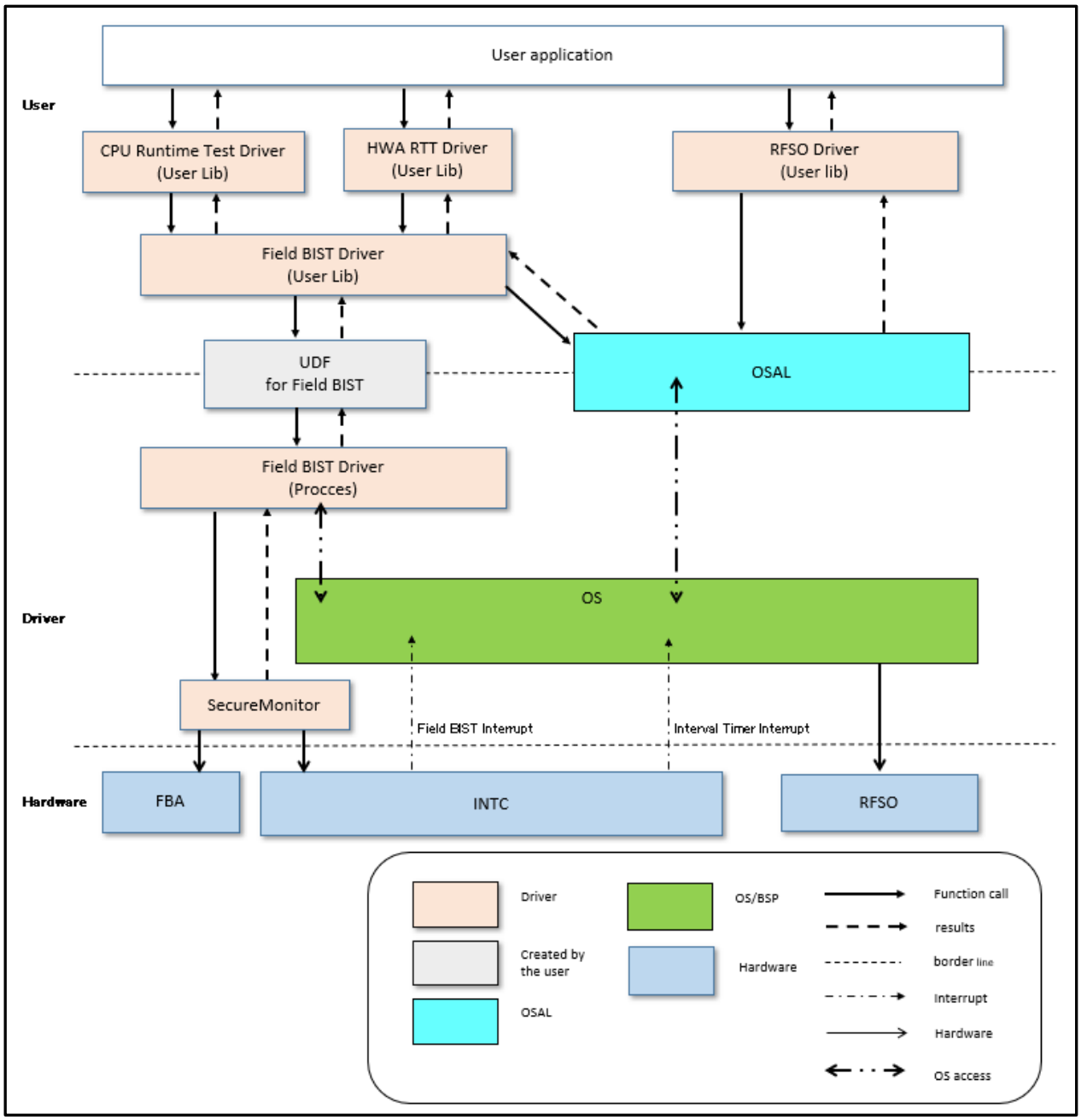

Figure 4-15: System configuration for Gen 3 and Gen 4 V4x QNX

[For R-Car Gen 4 S4]

S4 doesn't not have software to support RTT on Linux.

For QNX, it follows the same architecture as V3x/V4x QNX.

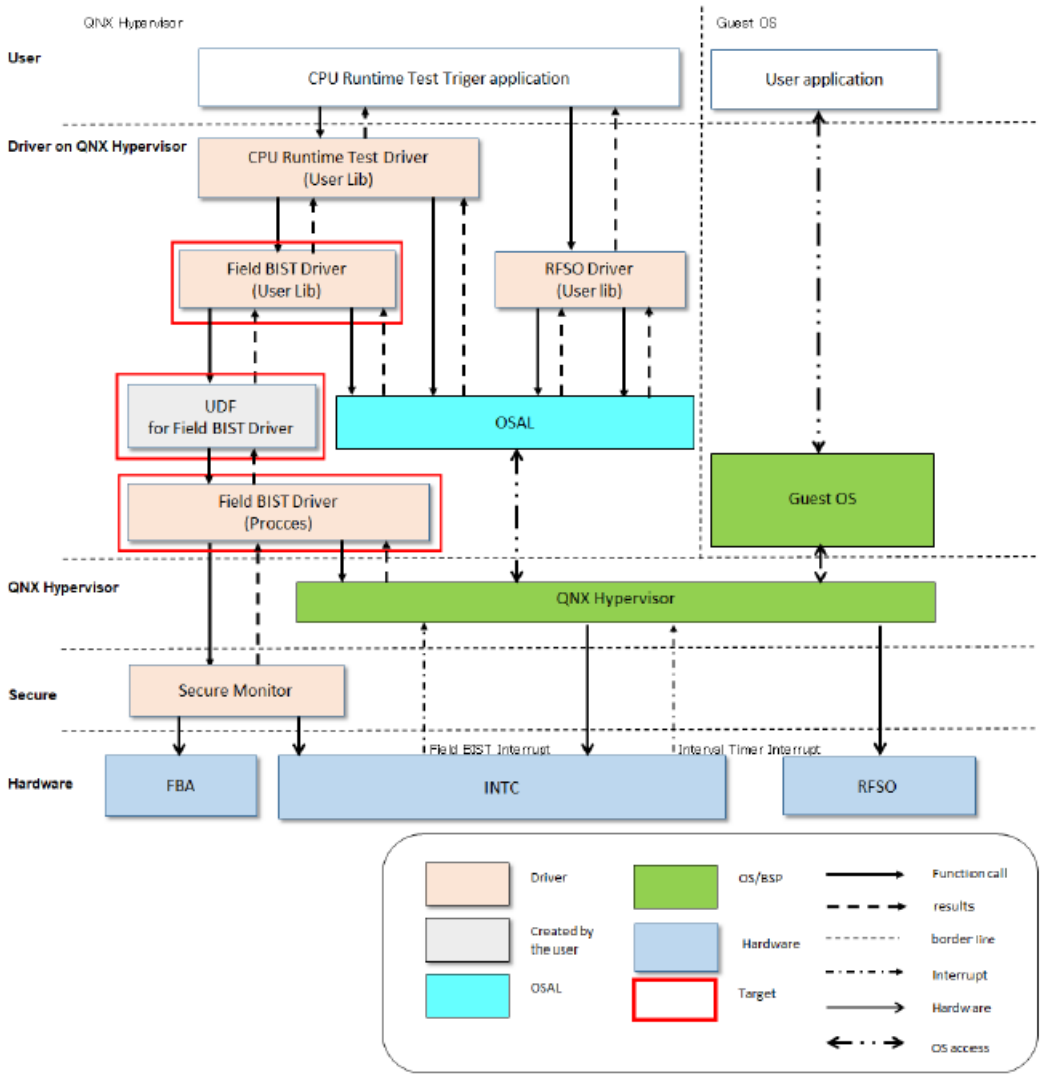

Figure 4-16: System configuration for Gen 4 S4 QNX

Since the UDF (gray blocks in the figures) requires user to modify according to user system, or the used OS, they are not provided in ASIL quality. Renesas do provide them as reference code for the user to reference and modify as needed.

Table 4-4 shows the scope of ASIL for the components of the Field BIST driver.

Table 4-4: The scope of ASIL for the components of the Field BIST driver

Component in Field BIST driver	ASIL or Reference
Field BIST driver	ASIL
UDF for Field BIST driver	Reference
[For Linux] Field BIST for using kernel API	Reference
[For QNX] Field BIST driver (Process)	ASIL

For details of what and how to implement in the UDF please refer to the Field BIST driver UM ([14] R-Car V3M/V3H Field BIST driver Software Users Manual, [24] R-Car V4H Field BIST driver Software Users Manual and [34]R-Car S4 Field BIST driver Software Users Manual).

5. FAQs

5.1 Are the Test Patterns from Renesas part of the Software release package?

- No. They are part of the R-Car Safety Package from the HW delivery. Refer to **Chapter 4.5.**

5.2 How many test patterns exist for Scan test and memory BIST?

- For most R-Car devices, Scan test consists of multiple test patterns, and memory BIST consists of a single test pattern. Please check the test pattern summary sheet in the safety package for the exact number of test patterns for a specific device.

5.3 Can a single test pattern achieve the required diagnostic coverage?

- Accomplishing a fault coverage of at least 90% by the scan tests requires many test patterns in almost every hierarchy. Please use FMEDA tool to determine the SPFM and LFM that can be achieved with your safety mechanism configuration.

5.4 Can user execute the RTT for each Hierarchy in parallel?

- User can execute the RTT for some hierarchies in parallel. Example: RTT on target hierarchy IMP Ch0 and IMP Ch1 can be executed in parallel. However, CPU hierarchies have HW limitations. Please check HW SAN for restrictions of parallel execution. Refer to **Chapter 4.1** and **Figure 5-1** below for more information**.**

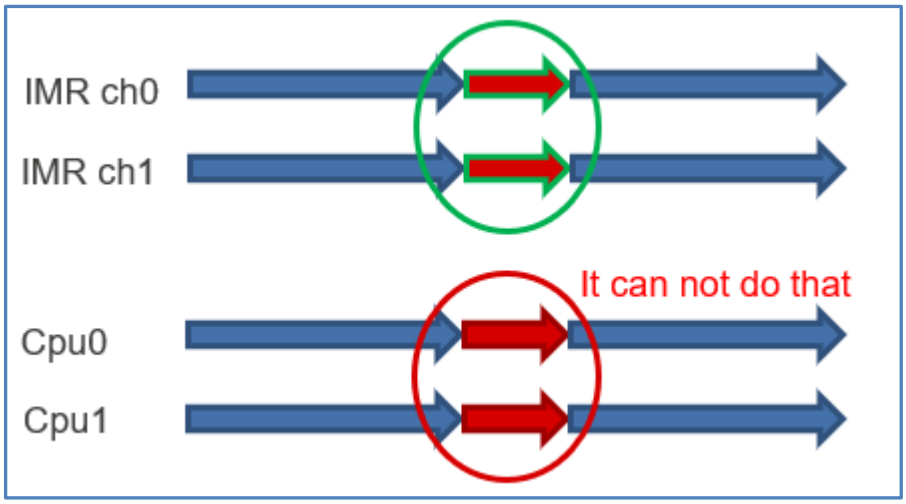

Figure 5-1 Parallel Operation: IMR vs CPU

6. Reference

No.	Document
[1]	INTERNATIONAL STANDARD ISO 26262 Road vehicles — Functional safety Second edition 2018-12
[2]	ARM exception levels
[3]	R-Car Series, 3rd Generation Hardware User's Manual
[4]	R-Car Series, 3rd Generation Safety Requirement Specification
[5]	R-Car Series, 3rd Generation Safety Application Note
[6]	R-Car Series, 3rd Generation Hardware Description for Functional Safety HD-FUSA
[7]	R-Car Series, 4th Generation Hardware User's Manual
[8]	R-Car Series, 4th Generation Safety Requirement Specification
[9]	R-Car Series, 4th Generation Safety Application Note
[10]	Runtime Test Pattern List R-Car X3x
[11]	Runtime Test Pattern List R-Car X4x
[12]	R-Car V3M/V3H CPU Runtime Test Driver Software Users Manual
[13]	R-Car V3M/V3H CPU Runtime Test Driver Software Safety Application Note
[14]	R-Car V3M/V3H Field BIST driver Software Users Manual
[15]	R-Car V3M/V3H Field BIST driver Software Safety Application Note
[16]	R-Car V3M/V3H RFSO driver Software Users Manual
[17]	R-Car V3M/V3H RFSO driver Software Safety Application Note
[18]	R-Car V3M/V3H HWA Runtime test driver User's Manual: Software
[19]	R-Car V3M/V3H/V3H2 HWARTT Driver Software Safety Application Note
[20]	R-Car V3M/V3H Secure Monitor Software Users Manual
[21]	R-Car V3M/V3H Secure Monitor Software Safety Application Note
[22]	R-Car V4H/V4M CPU Runtime Test Driver Software Users Manual
[23]	R-Car V4H/V4M CPU Runtime Test Driver Software Safety Application Note
[24]	R-Car V4H/V4M Field BIST driver Software Users Manual
[25]	R-Car V4H/V4M Field BIST driver Software Safety Application Note
[26]	R-Car V4H/V4M RFSO driver Software Users Manual
[27]	R-Car V4H/V4M RFSO driver Software Safety Application Note
[28]	R-Car V4H/V4M HWA Runtime test driver User's Manual: Software
[29]	R-Car V4H/V4M HWARTT Driver Software Safety Application Note
[30]	R-Car V4H/V4M Secure Monitor Software Users Manual
[31]	R-Car V4H/V4M Secure Monitor Software Safety Application Note
[32]	R-Car S4 CPU Runtime Test Driver Software Users Manual
[33]	R-Car S4 CPU Runtime Test Driver Software Safety Application Note
[34]	R-Car S4 Field BIST driver Software Users Manual
[35]	R-Car S4 Field BIST driver Software Safety Application Note
[36]	R-Car S4 RFSO driver Software Users Manual

[37]	R-Car S4 RFSO driver Software Safety Application Note
[38]	R-Car S4 Secure Monitor Software Users Manual
[39]	R-Car S4 Secure Monitor Software Safety Application Note

7. Revision History

Rev.	Date	Description	
		Page	**Summary**
0.01	11th Nov 2022	-	Initial version
1.00	June 29, 2023	All	Update document structure. Update content details.

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

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

1. Precaution against Electrostatic Discharge (ESD) A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

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

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or

damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters
TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

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