User's Manual

R-Car V4H Series

Security User's Manual

Renesas Electronics
www.renesas.com

Rev.1.10 April 2025

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation.
Arm® and Cortex® are registered trademarks of Arm Limited.
The Arm logo is a registered trademark of Arm Limited.
For the “Cortex” notation, it is used as follows;
- Arm® Cortex®-A55
- Arm® Cortex®-R52

Note that after this page, they may be noted as Cortex-A55 and Cortex-R52 respectively.

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/.

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.

Table of Contents

1. Introduction

1.1 Scope

This document describes the security specifications of R-Car V4H.

The specifications are described from Hardware and Boot ROM program point of view.

1.2 Acronyms and Definitions

AIB	Asynchronous Interface Bus
CSD	CoreSight Debug module
DAP	Data Access Port
DBSC	Bus State Controller for SDRAM
DEVUID	Device Unique ID
IPL	Initial Program Loader
KDR	Key Device Root
LCM	Life Cycle Management
LCS	Life Cycle State
LPD	Low Pin Debug
OTP	One Time Programmable
RST	RESET Control Module
SCEG	Secure boot Engine
TAP	Test Access Port
TDBG	Test Debug Control Module
R-Car V4H HW UM	R-Car V4H Series User's Manual: Hardware

2. Overview

R-Car V4H supports secure boot and hardware access protection function. ICUMX boots from Boot ROM in R-Car V4H with disabling debug function at secure state. And the Initial Program Loader (IPL) is verified by Secure boot Engine controlled by ICUMX which executes the codes on the Boot ROM.

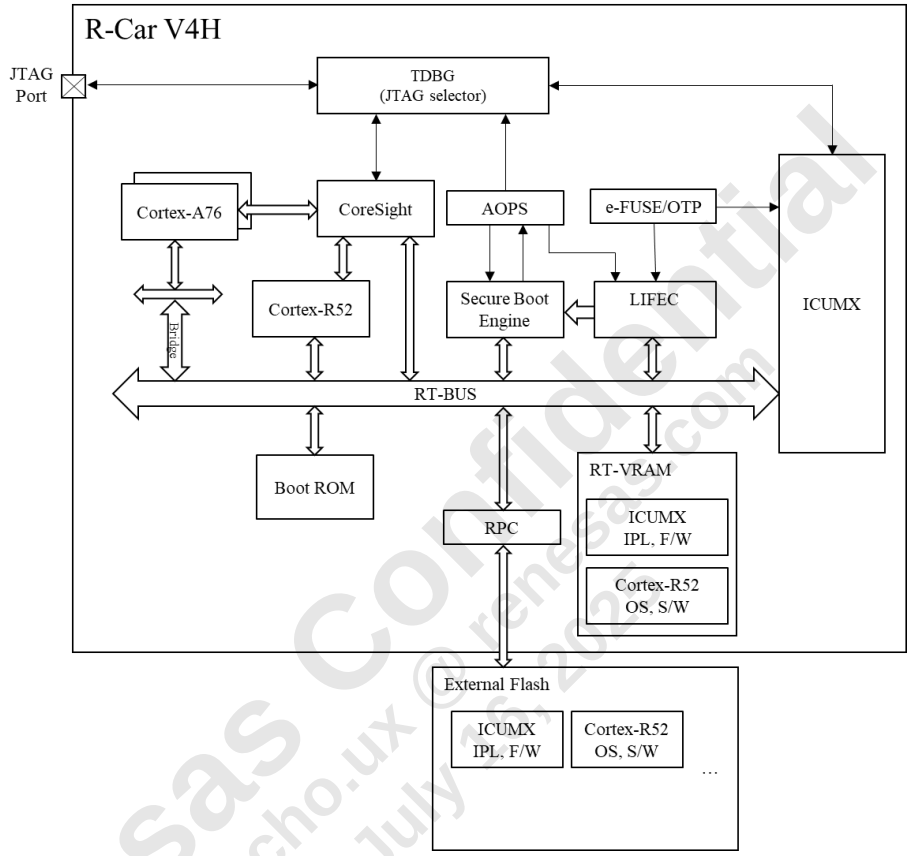


Figure 2-1 Block diagram for security related modules

2.1 Access Protection

R-Car V4H supports access protection function by Region ID protection.

Refer to R-Car V4H HW UM for details.

2.2 HW Acceleration for Cryptographic Services

R-Car V4H implement the Secure Boot Engine and ICUMX.

2.3 LCM state support

R-Car V4H supports LCM (Life Cycle Management) to control security level for each stage of life cycle. LCM states are determined by e-FUSE bits and OTP values.

Table 2-1 shows the descriptions of each LCM states.

Table 2-1 LCM State

LCM State	Description
Chip Manufacture (CM)	This state indicates that all debugging capabilities of the chip are enabled. Device unique key is fixed value.
Device Manufacture (DM)	In this state, debugging capabilities are still enabled. Device unique key is OTP value and not readable other than Secure Boot Engine or ICUMX.
Secure Disable (SD)	Used for marking devices that lack security functionality. This state is NOT supported.
Secure (SE)	This state permits the execution of security functions, but restricts the debugging capabilities.
Failure Analysis (FA)	Used for devices returned to the manufacture for failure analysis. Existing secret keys are lost, but regains full access to all debugging capabilities. (as available in Chip Manufacture state)

2.4 Secure Boot and Secure debug

R-Car V4H supports Secure Boot and Secure Debug functionality which enforce only boot from authorized program and debugger access.

In the Secure (SE) state, R-Car V4H boots from ICUMX while debugging function is disabled and external ROM boot is restricted. Enabling debug function and JTAG pins are controlled by ICUMX. Debug function can be enabled by secure debug procedure.

2.5 Security Vulnerability Countermeasure

R-Car V4H has security vulnerability countermeasure circuit to prevent bypass the secure boot by slip to other than SE state.

1. Stopping Boot sequence if OTP errors occur

   If OTP errors occur during the initial read of the OTP_MEM, R-Car V4H will stop the boot sequence and will not be able to boot.

   OTP errors are as follows:

   - DCLS error
   - Bit redundancy error between Main and Copy bits (*)
   - ECC uncorrectable error

   Note :

   (*) Setting the OTP Main and Copy bits must be completed without PRESET#.

   If PRESET# is asserted with only Main bits or Copy bits set, R-Car V4H will not be able to boot.

3. HW Acceleration for Secure Boot

R-Car V4H implement Secure Boot Engine.

3.1 Supported Algorithms

Table 3-1 Supported Algorithms of Secure Boot Engine Hardware accelerator

Category	Function	Note
HASH	MD-5, SHA-1, SHA-256, SHA-512	
AES		Key size: 128-bit, 192-bit, 256-bit Modes: ECB, CBC, CBC-CTS, OFB, CTR, CBC-MAC, CMAC, XCBC-MAC, XTS, CCM
PKA	Modular arithmetic (addition, subtraction, multiplication and division) Regular arithmetic (addition, subtraction, multiplication and division) Modular inversion Modular exponentiation Logical operations (AND, OR, XOR, SHIFT)	

3.2 Interface of Secure Boot Engine

Figure 3-1 shows the HW interfaces of Secure Boot Engine. Secure Boot Engine has dedicated module that is named as AOPS (Always On Persistent State). Secure Boot Engine can pass LCS(Life Cycle State) indication, Debug control signal via AOPS. The communication mechanism between Host and Secure Boot Engine is based on shared RAM(DRAM, System RAM or RT-VRAM) for message passing. Secure Boot Engine has an AXI master port. AXI master ports are connected to system bus. Secure Boot Engine can read e-FUSE and OTP data (e.g. Key Device Root, Digest of Root Public Key) from LIFEC via AIB I/F. Secure Boot Engine has Secure APB I/F. Only Secure Master can access via Secure APB I/F.

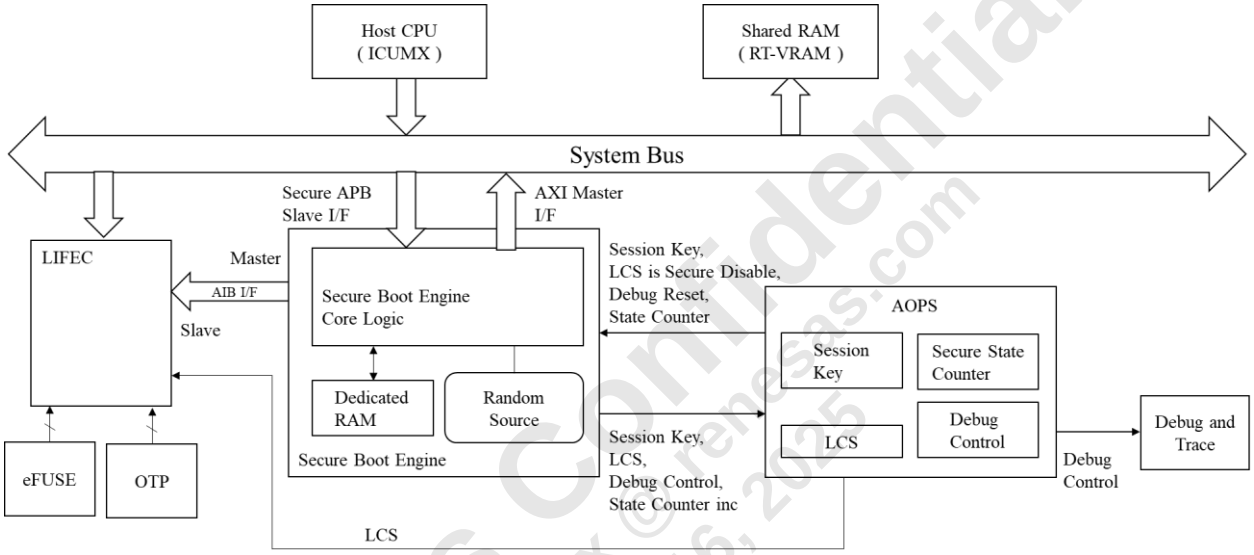


Figure 3-1 HW Interface of Secure Boot Engine, LIFEC, e-FUSE and OTP

4. LCM Support

R-Car V4H supports LCM (life cycle management) to control security level for each stage of life cycle.

4.1 LCM States Features

Table 4-1 shows the key features of each LCM state. LCM state is determined by Secure Boot Engine based on e-FUSE and OTP bits value.

Table 4-1 LCM states Key Features

LCM state	Boot mode			JTAG Debug	KDR value	KDR/KDR2/KCE value
	Master Boot Processor	Boot device restriction	Boot sequence Secure or Normal		Secure Boot Engine	ICUMX
CM	None	None	Depends on MD5 pin setting	Depend on MD pins setting	Fixed value	OTP value
DM	None	None	Same as CM	Same as CM	OTP value	OTP value
SE	ICUMX only	XIP mode is prohibited.	Secure Boot	Can be activated by secure debug function.	OTP value	OTP value
FA	None	None	Same as CM	Depend on MD pins setting	HW will zero the KDR and FW will randomize it each boot.	HW will zero the KDR/KDR2/KCE.

4.1.1 Chip Manufacture State (CM)

During this state, all boot mode and debugging capabilities are enabled. Key Device Root (KDR) in Secure Boot Engine is fixed value.

4.1.2 Device Manufacture State (DM)

In this state, boot mode and debugging capabilities are still enabled. The KDR value of OTP is used by Secure Boot Engine. If bits [0:7] of OPF in OTP are programmed to not zero, LCS change from DM to SE.

4.1.3 Secure Enabled (SE)

This is the state used for devices “in the field” (once they are out of the manufacturing line). This state permits the execution of security functions, but restricts the boot mode and debugging capabilities. The KDR value of OTP is used by Secure Boot Engine. When JTAG debug for secure host (SPIDEN of CoreSight or SPIDEN of Cortex-A76 or DBGEN of Cortex-R52) is enabled by secure debug procedure, Secure Boot Engine(SCEG) is immediately reset (hence, its root-key is cleared), and enters a temporary security-disabled state. It is possible to prevent Secure Boot Engine enters a temporary security-disabled state by OTP setting. In case of temporary security- disabled state, KDR of SCEG is fixed value, PKA and RNG of SCEG are disabled. Exiting security-disabled state requires a power-on reset of the device.

4.1.4 Failure Analysis State (FA)

This is a state for devices that are returned to the manufacturer for analysis of fatal failures. Secure Boot Engine HW will zero the KDR internally and Secure Boot Engine’s FW will randomize it each boot.

5. Boot device selection and Master boot processor

The SoC has several operation modes. The mode setting is done by external mode pins. Settable mode is restricted depending on Secure LCM state and OTP value.

In the Secure(SE) state, ICUMX boots from Boot ROM (on-chip) regardless mode pin setting and external ROM boot is restricted. MD[5] is not affect boot device selection. It is used for boot code flow control on Boot ROM. The boot device selection should be matched MD[4:1] pins and OTP settings.

Refer to 9.2.3 for more detail.

Boot device selection and Master boot processor selection are shown below.

Table 5-1 Boot device selection except for SE(Secure)

MD[4:1]	Description	Boot from
B'0000	No boot	Reserved
B'0001	No boot	Reserved
B'0010	HyperFlash ROM boot at 160MHz using DMA	Boot ROM (On-chip)
B'0011	HyperFlash ROM boot at 80MHz using DMA	Boot ROM (On-chip)
B'0100	Serial Flash ROM boot at single read 40MHz using DMA	Boot ROM (On-chip)
B'0101	No boot	Boot ROM (On-chip)
B'0110	Serial Flash ROM boot at 40/80/133MHz using DMA	Boot ROM (On-chip)
B'0111	Octal SPI Flash 160/80MHz using DMA	Boot ROM (On-chip)
B'1000	No boot	Boot ROM (On-chip)
B'1001	No boot	Boot ROM (On-chip)
B'1010	HyperFlash ROM at 160MHz(320Mbps) using XIP mode	RPC in External address space read mode H'00 0800 0000
B'1011	HyperFlash ROM at 80MHz using XIP mode	RPC in External address space read mode H'00 0800 0000
B'1100	No boot	Boot ROM (On-chip)
B'1101	eMMC boot at 12.5MHz x8 bus widths using DMA	Boot ROM (On-chip)
B'1110	No boot	Boot ROM (On-chip)
B'1111	SCIF download mode	Boot ROM (On-chip)

Table 5-2 Boot device selection at SE(Secure)

MD[4:1]	Description	Boot from
B'0000	No boot	Boot ROM (On-chip)
B'0001	No boot	Boot ROM (On-chip)
B'0010	HyperFlash ROM boot at 160MHz using DMA	Boot ROM (On-chip)
B'0011	HyperFlash ROM boot at 80MHz using DMA	Boot ROM (On-chip)
B'0100	Serial Flash ROM boot at single read 40MHz using DMA	Boot ROM (On-chip)
B'0101	No boot	Boot ROM (On-chip)
B'0110	Serial Flash ROM boot at 40/80/133MHz using DMA	Boot ROM (On-chip)
B'0111	Octal SPI Flash 160/80MHz using DMA	Boot ROM (On-chip)
B'1000	No boot	Boot ROM (On-chip)
B'1001	No boot	Boot ROM (On-chip)
B'1010	No boot	Boot ROM (On-chip)
B'1011	No boot	Boot ROM (On-chip)
B'1100	No boot	Boot ROM (On-chip)
B'1101	eMMC boot at 12.5MHz x8 bus widths using DMA	Boot ROM (On-chip)
B'1110	No boot	Boot ROM (On-chip)
B'1111	No boot	Boot ROM (On-chip)

Table 5-3 Master boot processor selection except for SE

MD7	MD6	Description
0	0	Reserved
0	1	Reserved
1	0	Booted through ICUMX
1	1	Booted through Cortex-R52

Table 5-4 Master boot processor selection at SE

MD7	MD6	Description
0	0	Booted through ICUMX
0	1	Booted through ICUMX
1	0	Booted through ICUMX
1	1	Booted through ICUMX

6. Security related e-FUSE/OTP bits

Table 6-1 shows the Secure Boot Engine Security related e-FUSE/OTP bits of R-Car V4H.

Table 6-1 Security related e-FUSE/OTP bits

Function	Bit number	Description
Key Device Root (KDR)	256	KDR is Per-device unique 256-bit AES key, used as root for deriving all device-specific keys. This key is connected to ICUMX ROM_KEY_1 and is set by Renesas other than CM device.
Key Device Root2 (KDR2)	256	KDR2 is Per-device unique 256-bit AES key, used as key wrapped key. Refer to ICUMX HW UM for more details. This key is connected to ICUMX ROM_KEY_2 and is set by Renesas other than CM device.
Provisioning renewability secret (SCP)	64	SCP share of KCP (Provisioning Master Key). This bit is set by Renesas other than CM device.
Manufacturer-programmed flags (MPF)	32	MPF is for KDR/SCP value error detection. This bit is set by Renesas other than CM device.
OPF	24	OPF is for HBK value error detection. [7:0] Number of "0" bits in HBK (0..255) [15:8] Set to 255 if HBK is set. [23:16] Reserved
Code encryption key for ICUMX (256bitKCE for icumx)	256	KCE for ICUMX is AES 256-bits key, used for encryption of the SW modules protected by Secure Boot. This key is used in ICUMX. This key is connected to ICUMX ROM_KEY_0.
Digest of Root Public key (HBK)	256	HBK is used as a 256-bit SHA256 digest of the Secure Boot public key.
Key revocation code	3	Key revocation code is used to select a root public key of the 4 root keys.
Trusted Firmware minimum version (TFMV)	128	TFMV and TFMV/NTFMV for the minimum version table are Encoded values between 0 and 128. The version is the number of bits set in the OTP.
TFMV for the minimum version table	128	
NTFMV for the minimum version table	128	
On Chip Debug ID (OCDID)	256	OCDID is 256 bits ID, used for authentication of on chip debug when LCM state is Secure. If OCDID is not set (All '0'), OCDID authentication of Secure Debug Function is skipped.
Temporary SD disable	2	Temporary SD disable bit disables transition to SD state if this bit is set to other than B'00. Renesas recommends that this bit should be set only for failure analysis.
Secure debug	2	Secure debug bit is Secure Debug limit check setting. Refer to 9.4 and 9.7.3 for more details.
Recovery	2	Recovery bit is Recovery function setting. Refer to 9.2.6 and 9.7.3 for more details.
TEST mode certificate ID	256	TEST mode certificate ID and TEST mode certificate ID_clone are 256 bits IDs. If these bits mismatch, the TEST port will be locked and unavailable. If the customer returns the device to Renesas for failure analysis, the values of these bits must be matched.
TEST mode certificate ID_clone	256	
Retry count	3	Retry counter for Secure Boot. Refer to 9.2.5 and 9.7.4 for more details.
Boot Device	4	Boot Device selection bit. Refer to 9.2.3 and 9.7.4 for more details.

Management bits (Read/Write protection setting) are supported.
OTP has management bit for each address.

7. JTAG and debug enable control

7.1 JTAG and debug enable control except SE(Secure)

When LCS is not Secure, both Main-JTAG and Sub-JTAG can be selected through MD pins. Internal debug authentication signals for Cortex-A76/Cortex-R52 are asserted automatically.

Refer to R-Car V4H HW UM for details.

7.2 JTAG and Debug enable control at SE state.

When LCS is Secure, all debug enable signals are disabled. Debug enable signals and Sub-JTAG port for debugging are enabled through Secure Debug function.

Refer to 9.4 for more details about Secure Debug function.

Table 7-1 shows the supported combinations of MD pin settings and Debug-Mask values.

Table 7-1 JTAG selection table (LCS is Secure) after the execution of secure debug function

MDT0	MD21,20	MD11	MD10	Main JTAG	Sub JTAG	Secure Debug authentication Phase			Debug-Mask Value		Description
						Certificate method	C&R(Cold) method	C&R(Hot) method	Secure Debug Certificate	Non-secure Debug Certificate	
0	00	1	0	ICUMX JTAG	Normal Function	BootROM	BootROM	IPL or later	H'20100000	-	ICUMX debug is possible.
			1	ICUMX LPD	Normal Function	BootROM	BootROM	IPL or later	H'30100000	-	ICUMX debug is possible.
	01	0	0	ICUMX JTAG	CoreSight	BootROM	BootROM	IPL or later	H'04100000	H'06EC37FF	Cortex-A76, Cortex-R52 and ICUMX debug are possible.
			1	ICUMX LPD	CoreSight	BootROM	BootROM	IPL or later	H'14100000	H'16EC37FF	Cortex-A76, Cortex-R52 and ICUMX debug are possible.
	10	0	0	CoreSight	Normal Function	BootROM	BootROM	IPL or later	-	H'0AEC37FF	Cortex-A76 and Cortex-R52 debug are possible.
1	01	0	0	CoreSight	VDSP0	BootROM	BootROM	IPL or later	-	H'07EFF7FF	Cortex-A76, Cortex-R52 and VDSP0 debug are possible.
			1	CoreSight	VDSP1	BootROM	BootROM	IPL or later	-	H'17EFF7FF	Cortex-A76, Cortex-R52 and VDSP1 debug are possible.
		1	0	CoreSight	VDSP2	BootROM	BootROM	IPL or later	-	H'27EFF7FF	Cortex-A76, Cortex-R52 and VDSP2 debug are possible.
			1	CoreSight	VDSP3	BootROM	BootROM	IPL or later	-	H'37EFF7FF	Cortex-A76, Cortex-R52 and VDSP3 debug are possible.
	11	0	0	CoreSight	PAP	BootROM	BootROM	IPL or later	-	H'4FEC3FFF	Cortex-A76, Cortex-R52 and PAP debug are possible.
		1	0	ICUMX JTAG	CoreSight	BootROM	BootROM	IPL or later	H'2D100000	H'2FEC37FF	Cortex-A76, Cortex-R52 and ICUMX debug are possible.
		1	1	ICUMX LPD	CoreSight	BootROM	BootROM	IPL or later	H'3D100000	H'3FEC37FF	Cortex-A76, Cortex-R52 and ICUMX debug are possible.

* : don't care
\- : Not used

The Debug Mask value in the Secure Debug certificate (Secure Certificate and Non-secure Certificate) should be specified to enable debugging for each CPUs.

In case of Hot plug-in selected for C&R authentication, DCU_EN register is not set and locked in the BootROM code.

Note : In case of selected certificate authentication can select cold plug-in only.

It is recommended that Region ID set to DCU_EN register by IPL to protect access from other than ICUMX, because the debugging will be open if DCU_EN register is set illegally.

It is assumed that C&R authentication is performed by an interrupt from the debugger.

7.3 Debug Control Unit Enable register (DCU_EN) H'E660 0A64

This register controls debug enable / disable for each CPU.

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

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

Bit	Bit Name	Initial Value	R/W	Description
31	LOCK	0	W	Lock bit. Writing 1 to this bit locks this register.
30 to 0	DEBUG_DOMAINS	All 0	W	DCU Enable bits

8. Security related registers

This section shows the R-Car V4H security related registers.

8.1 LIFEC SDRAM data scramble setting register

Table 8-1 SDRAM data scramble setting register list of LIFEC.

Register Abbreviation	LCS	
	CM/DM/SE	FA
DBSCRM	R/W	R

R/W: Readable and Writable

R : Readable only. Write is ignored.

8.2 e-FUSE/OTP related registers

Table 8-2 e-FUSE/OTP related registers List

Address offset	name	CM	DM	SE	FA	initial value
H'E611 0010	KDR3	W/R(*c)	-	-	-	KDR[127:96]
H'E611 0014	KDR2	W/R(*c)	-	-	-	KDR[95:64]
H'E611 0018	KDR1	W/R(*c)	-	-	-	KDR[63:32]
H'E611 001C	KDR0	W/R(*c)	-	-	-	KDR[31:0]
H'E611 0024	MPF	W/R(*c)	-	-	-	MPF[31:0]
H'E611 0028	LCS	R	R	R	R	H'0000 0050 ->LCS indicator value
H'E611 0040	DEVUID0	W/R	R	R	R	DEVUID[31:0]
H'E611 0044	DEVUID1	W/R	R	R	R	DEVUID[63:32]
H'E611 0048	DEVUID2	W/R	R	R	R	DEVUID[95:64]
H'E611 0088	KDR7	W/R(*c)	-	-	-	KDR[255:224]
H'E611 008C	KDR6	W/R(*c)	-	-	-	KDR[223:192]
H'E611 0090	KDR5	W/R(*c)	-	-	-	KDR[191:160]
H'E611 0094	KDR4	W/R(*c)	-	-	-	KDR[159:128]
H'E611 00A0	SCP0	W/R(*c)	-	-	-	SCP[31:0]
H'E611 00A4	SCP1	W/R(*c)	-	-	-	SCP[63:32]
H'E611 00A8	OPF	W/R(*c)	-	-	-	OPF[31:0]
H'E611 0100	HBK0	W/R	R	R	R	HBK[31:0]
H'E611 0104	HBK1	W/R	R	R	R	HBK[63:32]
H'E611 0108	HBK2	W/R	R	R	R	HBK[95:64]
H'E611 010C	HBK3	W/R	R	R	R	HBK[127:96]
H'E611 0110	HBK4	W/R	R	R	R	HBK[159:128]
H'E611 0114	HBK5	W/R	R	R	R	HBK[191:160]
H'E611 0118	HBK6	W/R	R	R	R	HBK[223:192]
H'E611 011C	HBK7	W/R	R	R	R	HBK[255:224]
H'E611 0218	TFMV0	W/R	R	R	R	TFMV[31:0]
H'E611 021C	TFMV1	W/R	R	R	R	TFMV[63:32]
H'E611 0220	TFMV2	W/R	R	R	R	TFMV[95:64]
H'E611 0224	TFMV3	W/R	R	R	R	TFMV[127:96]
H'E61B F180	TFMV_TABLE0	R	R	R	R	TFMV_MIN_VER_TABLE[31:0]
H'E61B F184	TFMV_TABLE1	R	R	R	R	TFMV_MIN_VER_TABLE[63:32]
H'E61B F188	TFMV_TABLE2	R	R	R	R	TFMV_MIN_VER_TABLE[95:64]
H'E61B F18C	TFMV_TABLE3	R	R	R	R	TFMV_MIN_VER_TABLE[127:96]
H'E61B F190	NTFMV_TABLE0	R	R	R	R	NTFMV_MIN_VER_TABLE[31:0]
H'E61B F194	NTFMV_TABLE1	R	R	R	R	NTFMV_MIN_VER_TABLE[63:32]
H'E61B F198	NTFMV_TABLE2	R	R	R	R	NTFMV_MIN_VER_TABLE[95:64]
H'E61B F19C	NTFMV_TABLE3	R	R	R	R	NTFMV_MIN_VER_TABLE[127:96]

(*c) These bits can be accessed in not debug case. Masking condition is (DBGEN | SPIDEN |SPNIDEN | NIDEN)

W/R: Readable and Writable

R : Readable only. Write is ignored.

- : Not Readable and Not Writable. Read value is 0. Write is ignored.

8.2.1 KDR*n* (*n*=0~7)

KDR*n* (*n*=0~7) are registers to read KDR value which is programmed in OTP bits. These registers can be accessed in LCM = CM state only. Table 8-3 shows the bit assignment of each KDR register.

Table 8-3 Bit Assignment of KDR

register name	OTP bit assign
KDR7	KDR[255:224]
KDR6	KDR[223:192]
KDR5	KDR[191:160]
KDR4	KDR[159:128]
KDR3	KDR[127:96]
KDR2	KDR[95:64]
KDR1	KDR[63:32]
KDR0	KDR[31:0]

8.2.2 MPF

MPF is registers to read MPF value which is programmed in e-FUSE/OTP bits. This register can be accessed in LCM = CM state only. Table 8-4 shows the bit assignment of MPF register.

Table 8-4 Bit Assignment of MPF

register name	e-FUSE/OTP bit assign
MPF	MPF[31:0]

8.2.3 LCS

LCS is 32 bits read only register. Low 8 bits of the register shows the LCS determined by Secure Boot Engine.

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

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

Bit	Bit Name	Initial Value	R/W	Description
31 to 8	-	All 0	R	Reserved.
7 to 0	LCS	H'50 -> LCS indicator value	R	Indicator Value of each LCS H'00: CM H'10: DM H'50: SE H'F2: FA

8.2.4 DEVUID*n* (*n*=0~2)

DEVUID*n* (*n*=0~2) are registers to read DEVUID value which is programmed in e-FUSE bits. These registers can be written in LCM = CM state only. These registers are Renesas internal use. Table 8-5 shows the bit assignment of each DEVUID registers.

Table 8-5 Bit Assignment of DEVUID

register name	e-FUSE bit assign
DEVUID0	DEVUID[31:0]
DEVUID1	DEVUID[63:32]
DEVUID2	DEVUID[95:64]

8.2.5 SCP*n* (*n*=0~1)

SCP*n* (*n*=0~1) are registers to read SCP value which is programmed in e-FUSE bits. These registers can be accessed in LCM = CM state only. Table 8-6 shows the bit assignment of each SCP registers.

Table 8-6 Bit Assignment of SCP

register name	e-FUSE bit assign
SCP0	SCP[31:0]
SCP1	SCP[63:32]

8.2.6 OPF

OPF is registers to read OPF value which is programmed in OTP bits. This register can be accessed in LCM = CM state only. Table 8-7 shows the bit assignment of OPF register.

Table 8-7 Bit Assignment of OPF

register name	OTP bit assign
OPF	OPF[31:0]

8.2.7 HBK*n* (*n*=0~7)

HBK*n* (*n*=0~7) are registers to read HBK value which is programmed in OTP bits. These registers can be read in any LCM states. Table 8-8 shows the bit assignment of each HBK registers.

Table 8-8 Bit Assignment of HBK

register name	OTP bit assign
HBK0	HBK[31:0]
HBK1	HBK[63:32]
HBK2	HBK[95:64]
HBK3	HBK[127:96]
HBK4	HBK[159:128]
HBK5	HBK[191:160]
HBK6	HBK[223:192]
HBK7	HBK[255:224]

8.2.8 TFMV n (n=0~3)

TFMV n (n=0~3) is registers to read TFMV value which is programmed in OTP bits. These registers can be read in any states. These registers can be written in LCM = CM state only. Table 8-9 shows the bit assignment of each TFMV register.

Table 8-9 Bit Assignment of TFMV

register name	OTP bit assign
TFMV0	TFMV[31:0]
TFMV1	TFMV[63:32]
TFMV2	TFMV[95:64]
TFMV3	TFMV[127:96]

8.2.9 TFMV_TABLE0~3

TFMV_TABLE0~3 are registers to read TFMV for minimum version table value which is programmed in OTP bits. These registers can be read in any states. Table 8-10 shows the bit assignment of each TFMV_TABLE register.

Table 8-10 Bit Assignment of TFMV for minimum version table

register name	OTP bit assign
TFMV_TABLE0	TFMV_MIN_VER_TABLE[31:0]
TFMV_TABLE1	TFMV_MIN_VER_TABLE[63:32]
TFMV_TABLE2	TFMV_MIN_VER_TABLE[95:64]
TFMV_TABLE3	TFMV_MIN_VER_TABLE[127:96]

8.2.10 NTFMV_TABLE0~3

NTFMV_TABLE0~3 are registers to read NTFMV for minimum version table value which is programmed in OTP bits. These registers can be read in any states. Table 8-11 shows the bit assignment of each NTFMV_TABLE register.

Table 8-11 Bit Assignment of NTFMV for minimum version table

register name	OTP bit assign
NTFMV_TABLE0	NTFMV_MIN_VER_TABLE[31:0]
NTFMV_TABLE1	NTFMV_MIN_VER_TABLE[63:32]
NTFMV_TABLE2	NTFMV_MIN_VER_TABLE[95:64]
NTFMV_TABLE3	NTFMV_MIN_VER_TABLE[127:96]

9. Boot ROM code

9.1 Overview

This chapter describes an overview of Boot ROM program.

9.1.1 Features

Boot ROM program is implemented on Boot ROM, and it has 4 features for booting R-Car V4H.

Table 9-1 shows the feature supported by R-Car V4H.

Table 9-1 Feature supported by R-Car V4H.

Feature	Detail		R-Car V4H
Boot Processer	Cortex-R52		✓
	ICUMX		✓
ROM code size	Cortex-R52	16 Kbyte	✓
	ICUMX	144Kbyte	✓
Boot Mode	Normal Boot		✓
	Secure Boot (only ICUMX)		✓
Boot Address	Cortex-R52	H'EB10 0000	✓
	ICUMX	H'EB10 4000	✓
Boot device	HyperFlash 160/80MHz		✓
	Serial Flash 133/80/40MHz		✓
	Octal SPI Flash 160/80MHz	Micron	✓
		Macronix	✓
	eMMC 12.5MHz		✓
	SCIF Download	SCIF	✓
		HSCIF	✓
Secure Boot Key	RSA Key length	2048 bit	✓
		3072 bit	✓
		4096 bit	✓
	4 Root RSA Public Key		✓
Secure Debug	HyperFlash		✓
	Serial Flash		✓
	Octal SPI Flash		✓
	eMMC		✓
	SCIF Download		✓
C&R	HyperFlash		✓
	Serial Flash		✓
	Octal SPI Flash		✓
	eMMC		✓
	SCIF Download		–

(1) Launching IPL (Initial Program Loader)

In general, software boot sequence of embedded system starts from “boot loader” software. But just after the LSI power-on reset, there is no software on RAM. Therefore, the R-Car V4H executes Boot ROM program first in order to load the boot loader from an external serial flash memory/eMMC to a RT-VRAM0 mirror space (hereinafter referred to as Internal RAM). And then Boot ROM program jumps to an entry point of the boot loader. In this document, the boot loader software loaded by Boot ROM program is called as IPL (Initial Program Loader). Figure 9-1 illustrates an overview of launching the IPL after the LSI power-on reset. Refer to 9.2 about details.

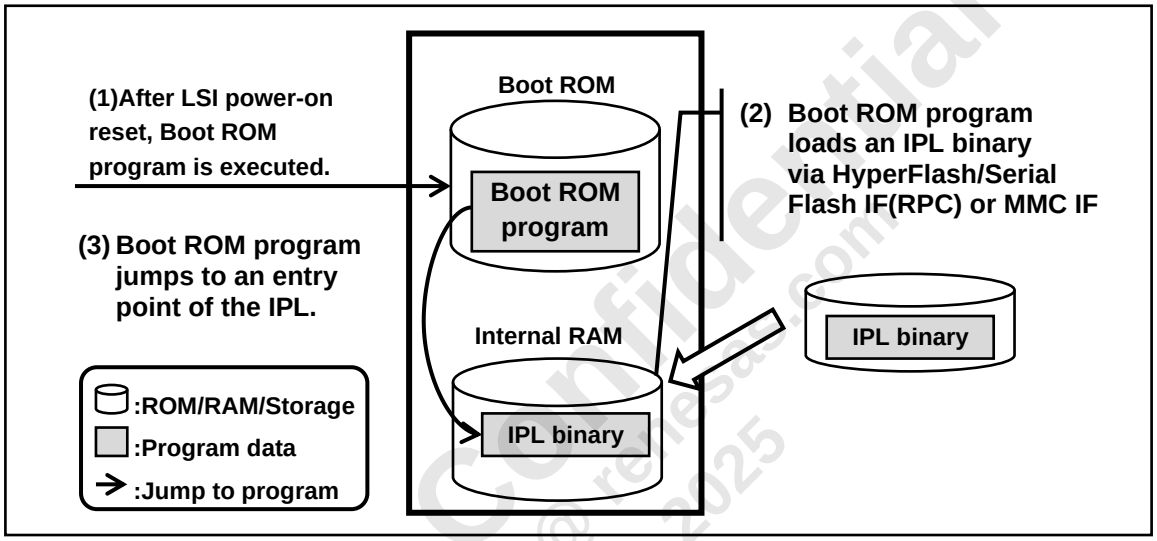


Figure 9-1 Overview of launching IPL

(2) Providing SCIF download mode

SCIF download mode uses the serial port, to transfer the data from Host PC to Internal RAM. Normal SCIF download sequence or Secure SCIF download sequence can be selected through LCM state and MD5.

The SCIF download mode supports SCIF and HSCIF. Refer to 9.3 for details.

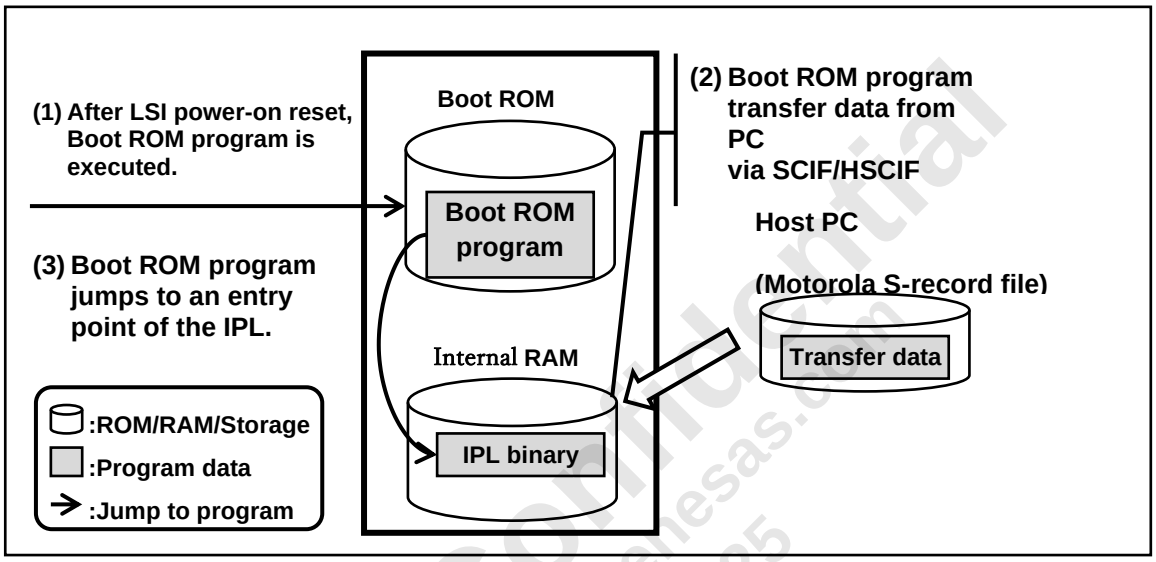


Figure 9-2 Overview of SCIF download mode

(3) Providing secure boot function

The secure boot function verifies the binary images. If the R-Car V4H is configured booting as “secure boot” mode (refer to 9.1.2), Boot ROM program calls the function internally in order to check whether the launching IPL binary is reliable.

After booting, secure boot function can be called from software.

(4) Providing secure debug certificate function

Debugging using JTAG interface is disabled if secure boot engine is used in Secure LCM state, and it can be enabled with secure debugging integrity check. See Section 9.4 about the specification and usage of this function.

(5) Security vulnerability countermeasures

(a) Execution and write limits on the Internal RAM and ICUMX Local RAM

Protect the memory area by granting execute / read / write permission in the MPU of ICUMX.

Refer to 9.2.1 CPU Initialization (3) Set MPU registers.

(b) Access restrictions by region ID

The region ID is set to protect the memory area.

Refer to 9.2.1 CPU Initialization (4) Set Region ID registers and Internal RAM protection.

(c) Implementation of stack canary

The canary value is held in the stack at the start of the function in the Boot ROM program, and if the value of the stack changes at the end of the function, it jumps to NO_BOOT.

9.1.2 Boot Mode

Boot Mode is determined by LCM state and MD[5] pin.

Secure boot verify the IPL image with a certificate. Normal boot does not verify the image.

If LCM state is SE, Boot Mode is forcibly fixed to Secure boot and fixed to boot from ICUMX.

Otherwise, can select Boot Mode by MD[5] pin.

Table 9-2 shows those relationships.

Table 9-2 Boot modes

CPU	LCM State	MD[5]	Verification	Boot Mode
ICUMX	CM	0	Yes	Secure Boot
		1	No	Normal Boot
	DM	0	Yes	Secure Boot
		1	No	Normal Boot
	FA	0	Yes	Secure Boot
		1	No	Normal Boot
	SE	Don’t care	Yes	Secure Boot
Cortex-R52	CM	0	No	No Boot
		1	No	Normal Boot
	DM	0	No	No Boot
		1	No	Normal Boot
	FA	0	No	No Boot
		1	No	Normal Boot
	SE	Don’t care	Yes	ICUMX Secure Boot

Figure 9-3 shows Boot ROM code flow chart.

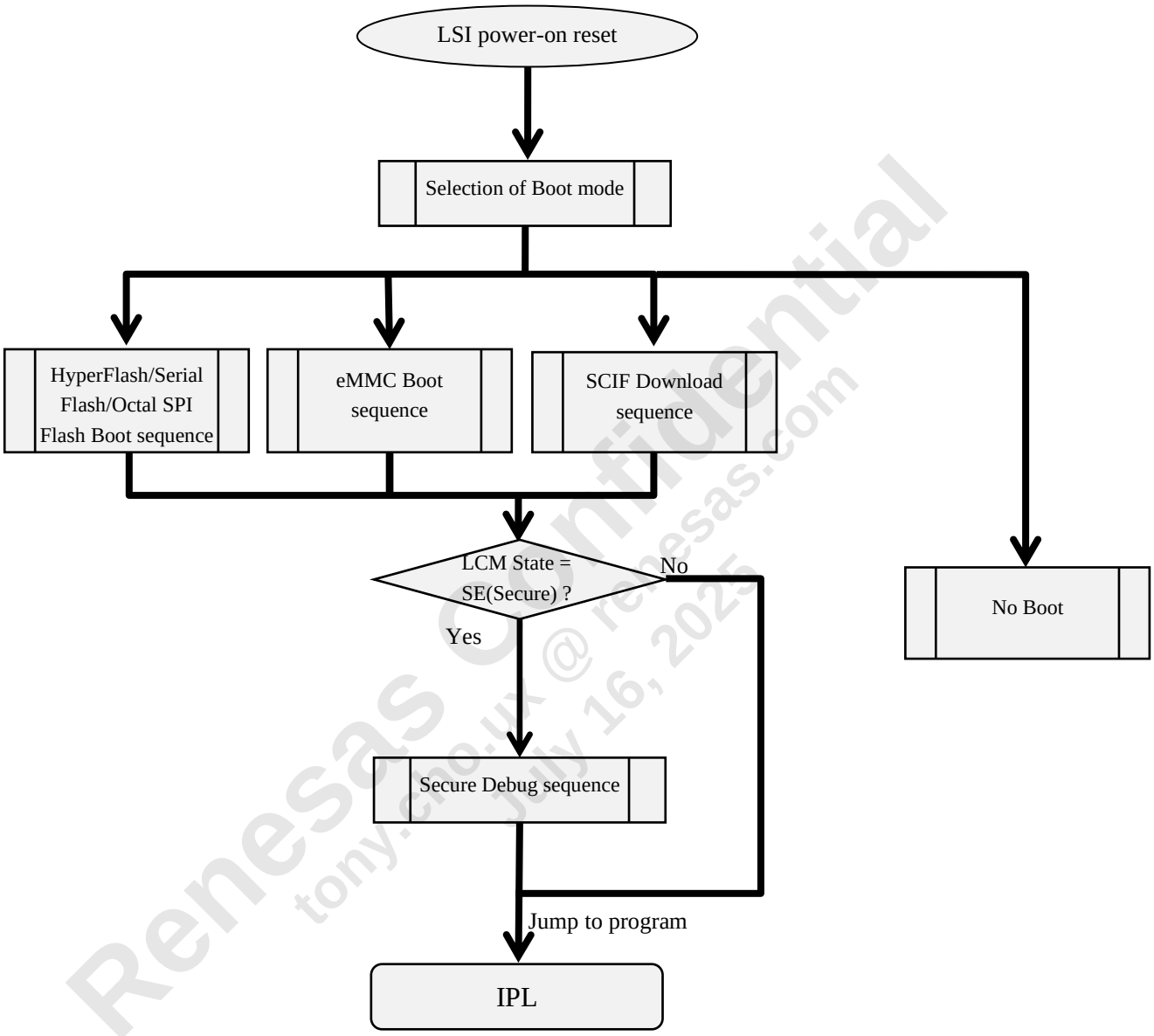


Figure 9-3 Boot ROM flowchart

9.1.3 Block Diagram

Figure 9-4 shows a block diagram of Boot ROM program, which is focused on functions of the program and using hardware modules.

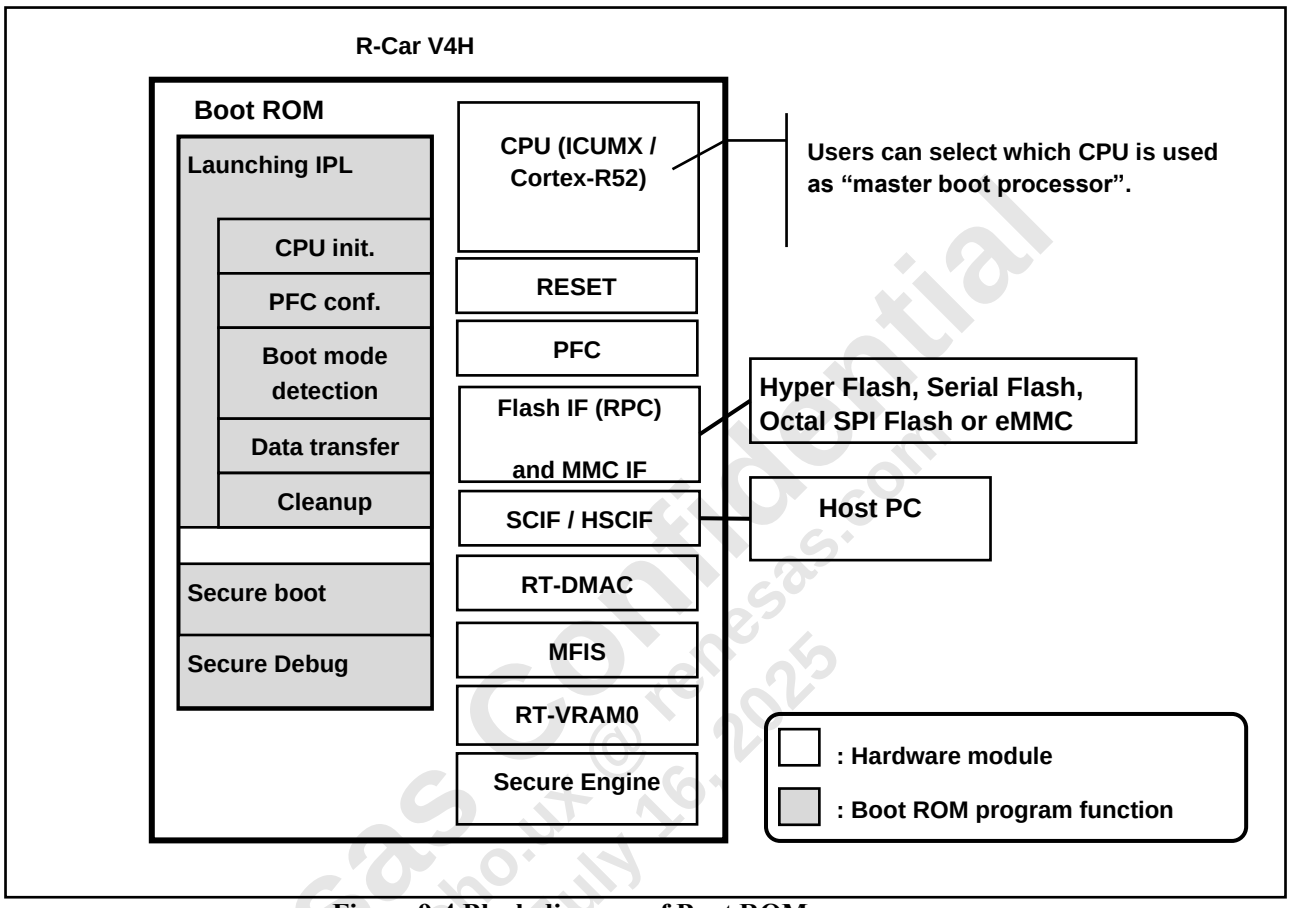


Figure 9-4 Block diagram of Boot ROM program

The “launching IPL” function consists of sub-functions as listed below.

(1) CPU initialization

A CPU used for the boot procedure is called as “master boot processor”. Therefore, at first Boot ROM program initializes the master boot processor. Refer to 9.2.1 about the initialization procedure.

(2) PFC configuration

Boot ROM configures PFC. Refer to 9.2.2 about details of the PFC configuration.

(3) Boot device

This sub-function detects which boot mode is selected. Refer to 9.2.3 Boot device about how to select boot mode.

(4) Data transfer

Boot ROM uses DMA transfer in order to load the IPL. Refer to 9.2.5 about detailed behavior of the data transfer.

(5) Cleanup

Before jumping to the IPL, Boot ROM re-initializes hardware modules used by Boot ROM program. Refer to 9.2.7 about details of the cleanup.

9.1.4 Memory Map

(1) Boot ROM

Boot ROM program is supported ICUMX / Cortex-R52 boot.

ICUMX code and Cortex-R52 code are located on Boot ROM as shown Figure 9-5. Either of the code is used is determined by the settings of MD7, MD6 pin. Refer to section 31 of R-Car V4H HW UM.

ICUMX memory map includes Secure Boot API.

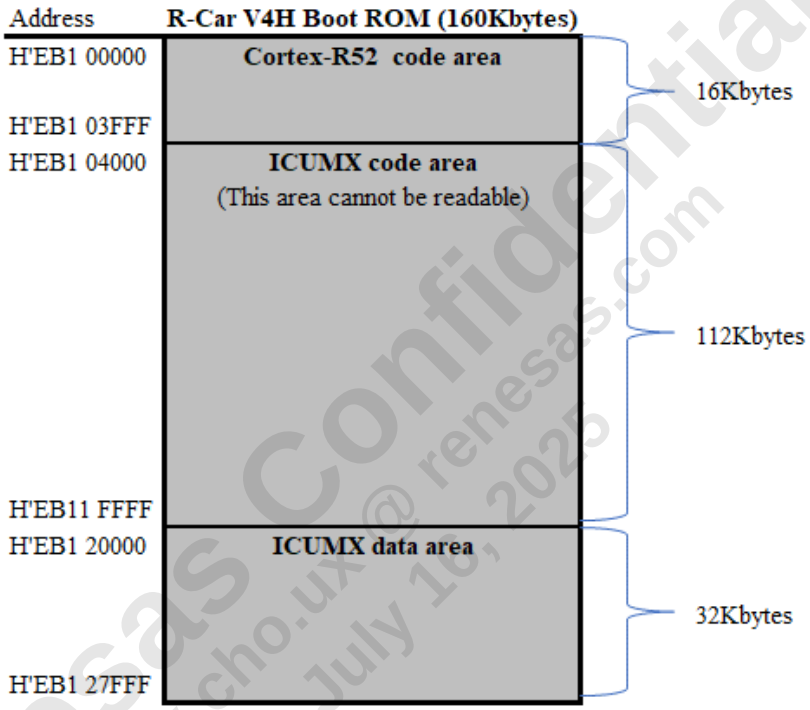


Figure 9-5 Boot ROM memory map

(2) Internal RAM (**RT-VRAM0) memory map (HyperFlash/Serial Flash/Octal SPI Flash)**

Figure 9-6 shows Internal RAM (RT-VRAM0) memory map of HyperFlash / Serial Flash / Octal SPI Flash boot.

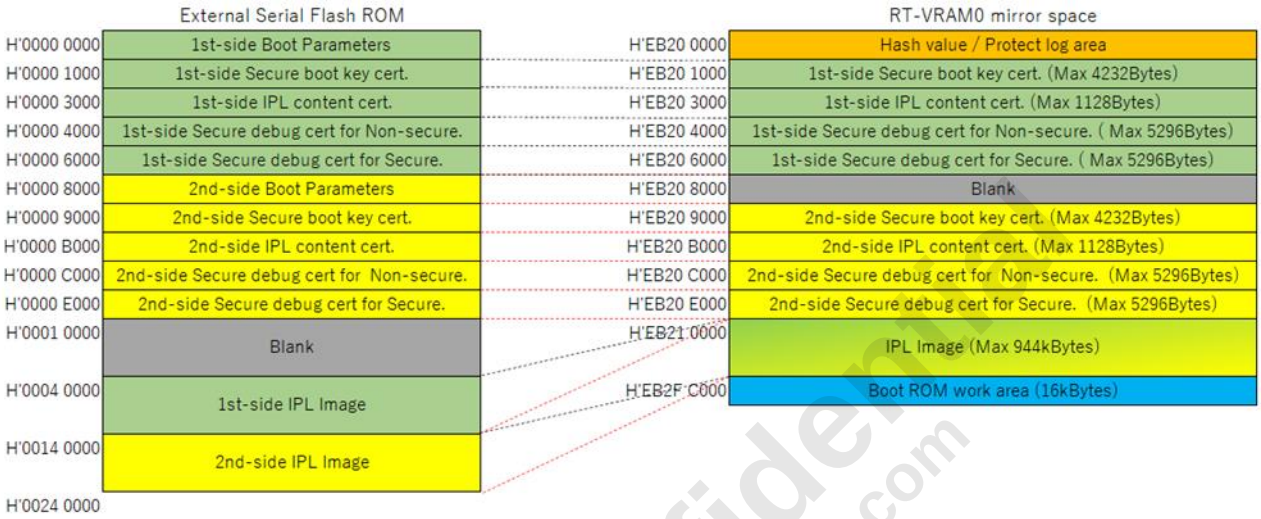


Figure 9-6 RT-VRAM0 memory map (Flash)

(3) Internal RAM (RT-VRAM0) Memory map (eMMC)

Figure 9-7 shows Internal RAM (RT-VRAM0) memory map of eMMC boot.

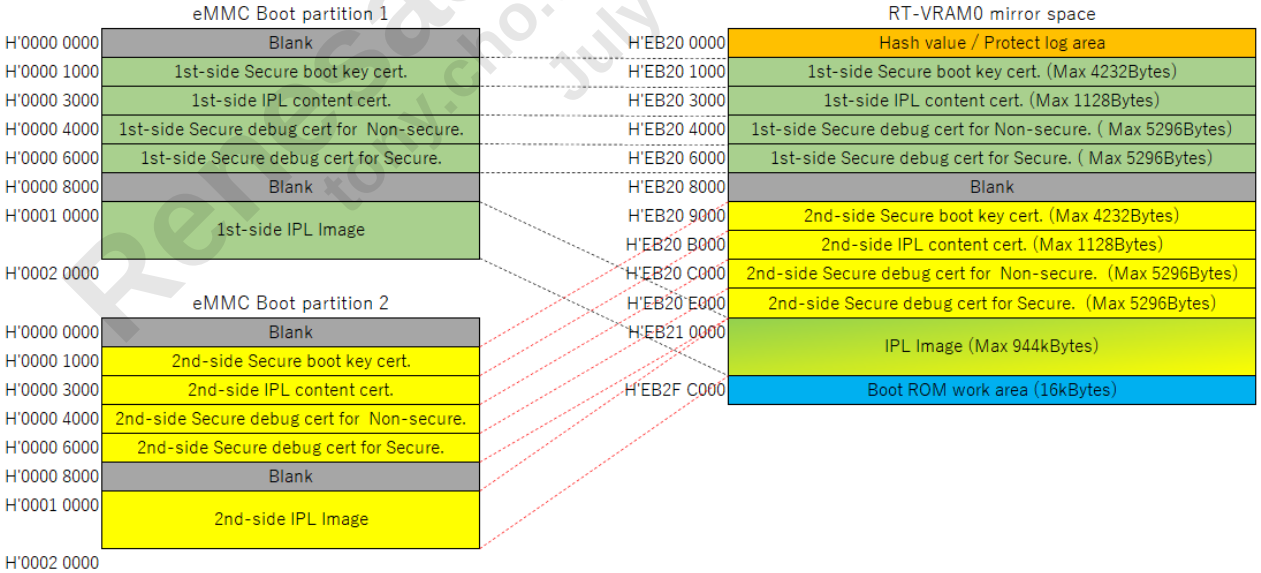


Figure 9-7 RT-VRAM0 memory map (eMMC)

(4) Internal RAM (RT-VRAM0) memory map (SCIF Download)

Regarding Memory map in case of SCIF Download mode, the data is transferred in Motorola S-record format.

The offset value of the data is as same memory map in case of eMMC Boot mode (but there is no 2nd-side cert.).

(5) Address of IPL transfer address and image size

The address that the IPL transfer address and size information is stored are determined according to the key length and the number of keys.

The IPL transfer address has a 64-bit field, the upper 32 bits must be zero, and address must be aligned on 512 byte boundary.

The IPL size has a 32-bit field, and the unit is specified in words (number of bytes divided by 4).

Also, the IPL address and IPL size must be within the IPL image area. Refer to Figure 9-6 and Figure 9-7.

If there is no Content certificate in Normal boot mode, it conforms to 1 key 2048-bit key length.

Table 9-3 Address of IPL address and size

Parameter (RT-VRAM0 mirror space physical address)			
Boot surface of storage	**RSA key length (bit)**	**IPL address**	**IPL size**
1st side	2048	H'EB20 3154	H'EB20 3264
	3072	H'EB20 31D4	H'EB20 3364
	4096	H'EB20 3254	H'EB20 3464
2nd side	2048	H'EB20 B154	H'EB20 B264
	3072	H'EB20 B1D4	H'EB20 B364
	4096	H'EB20 B254	H'EB20 B464

(6) Hash storage

The SHA256 hash value calculated by IPL verification of the Secure Boot sequence and the SHA256 hash value used for comparison are saved in the Internal RAM.

The storage location of the Internal RAM that stores the hash value is shown below.

Table 9-4 Hash storage in Internal RAM

Address (+offset)	Size (byte)	Explanation
H'EB2 00000 (+H'00)	32	A copied hash value from a certificate.
H'EB2 00020 (+H'20)	32	The hash value calculated from the image of IPL.

(7) Protect log

Save the protect log in H'200byte (H'EB20 0040 to H'EB20 023F) from the position of + H'40 from the beginning of Internal RAM.

The save format is shown below.

Table 9-5 Event code type

Event code	Explanation
H'EC00 0000	Boot ROM memory area protect initialize.
H'EC00 0001	The SW image is transferred to RAM.
H'EC00 0002	Decryption of the SW image was completed.
H'EC00 0003	SW image compare fails and retransfer the SW image to RAM.
H'EC00 0004	Boot ROM memory area protect de-initialize.
H'EC00 FFFF	Boot ROM finish (No data is recorded after this code)

Table 9-6 Protect log format

Address (+offset)	Size(byte)	Explanation
H'EB20 0040 (+H'00)	4	Event code
H'EB20 0044 (+H'04)	4	Register address (Region ID)
H'EB20 0048 (+H'08)	4	Register setting value
H'EB20 004C (+H'0C)	4	Register address (Region ID)
H'EB20 0050 (+H'10)	4	Register setting value

(8) Internal RAM (ICUMX **Local RAM) memory map**

Figure 9-8 shows Internal RAM (ICUMX Local RAM) memory map.

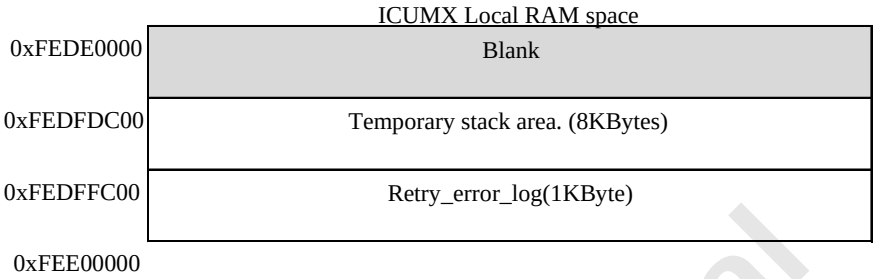


Figure 9-8 ICUMX Local RAM memory map

The purpose and usage period for each area of ICUMX Local RAM is as follows:

- Blank
  Boot ROM is not used.

- Temporary stack area
  During Boot ROM execution, this area is used as a stack area.
  After booting BootROM, this area will be released.

- Retry error log
  This area is recorded the Retry error logs during Boot ROM execution.

(9) Retry error log

The retry error log is in 1KB (H'FEDF FC00 to H'FEDF FFFF) at the end of ICUMX Local RAM.

The save format is shown below.

Table 9-7 Retry error log format

Address(+offset)	Size(byte)	Explanation
H'FEDF FC00 (+0x00)	4	[retry : 0] Error route identifier
H'FEDF FC04 (+0x04)	4	[retry : 0] Error code (Secure Library return code)
H'FEDF FC08 (+0x08)	4	[retry : 0] MFISBTSTSR register value
H'FEDF FC0C (+0x0C)	4	[retry : 0] reserve (all 0)
…	…	…
H'FEDF FC90 (+0x90)	4	[retry : 9] Error route identifier
H'FEDF FC94 (+0x94)	4	[retry : 9] Error code (Secure Library return code)
H'FEDF FC98 (+0x98)	4	[retry : 9] MFISBTSTSR register value
H'FEDF FC9C (+0x9C)	4	[retry : 9] reserve (all 0)
H'FEDF FCA0 (+0xA0)	4	LCS state
H'FEDF FCA4 (+0xA4)	4	MODEMR0 register value
H'FEDF FCA8 (+0xA8)	4	retry count

Table 9-8 Retry error route identifier

Value	Explanation
H'0000 0000	Normal
H'BE10 0000	1st side IPL Certificate verify error
H'BE20 0000	2nd side IPL Certificate verify error
H'BE30 0000	Both 1st and 2nd IPL Certificate verify error
H'BE10 0001	1st side IPL image load error
H'BE20 0001	2nd side IPL image load error
H'BE10 0002	1st side IPL image decrypt error
H'BE20 0002	2nd side IPL image decrypt error
H'BE10 0003	1st side IPL image compare error
H'BE20 0003	2nd side IPL image compare error

9.2 Launching IPL Function

This chapter describes how to load the IPL.

9.2.1 CPU Initialization

This sub-function initializes the master boot processor at beginning of the Boot ROM program. CPU Initialization is different depending on the type of master boot processor.

Listed below are the initialization procedure.

ICUMX

(1) Set H'0 to general-purpose registers (R1-R29, EIPC, CTPC).

(2) Set remap registers. (Refer to Table 9-9 Boot ROM remap settings)

For access to all address areas, the ICUMX has a remapping function for address translation during access to specific address ranges. The destination area for access by the remapping function is called the Remap area and this area is specified via the SICREMAP2Mi registers (i=0-15).

For access to the Remap area, set the higher-order 11 bits of the area to be remapped in the ICUMX_SICREMAP register and then access the area by accessing the range from H'FC00 0000 to H'FDFF FFFF (32 MB). The Remap area takes up a 2 MB range.

The remap settings remain after the boot ROM program.

Table 9-9 Boot ROM remap settings

Register name	Register address	Reset value	Setting value	Mapped IP
SICREMAP2M0	H'FF1FC400	H'FC000000	Not used	-
SICREMAP2M1	H'FF1FC404	H'FC200000	Not used	-
SICREMAP2M2	H'FF1FC408	H'FC400000	Not used	-
SICREMAP2M3	H'FF1FC40C	H'FC600000	Not used	-
SICREMAP2M4	H'FF1FC410	H'FC800000	Not used	-
SICREMAP2M5	H'FF1FC414	H'FCA00000	H'E7600000	Region ID
SICREMAP2M6	H'FF1FC418	H'FCC00000	H'E7200000	SYS-DMAC
SICREMAP2M7	H'FF1FC41C	H'FCE00000	H'E6400000	HSCIF
SICREMAP2M8	H'FF1FC420	H'FD000000	H'EE000000	eMMC
SICREMAP2M9	H'FF1FC424	H'FD200000	H'E6E00000	SCIF
SICREMAP2M10	H'FF1FC428	H'FD400000	H'FFC00000	RT-DMAC
SICREMAP2M11	H'FF1FC42C	H'FD600000	H'EE200000	RPC
SICREMAP2M12	H'FF1FC430	H'FD800000	H'E6200000	MFIS
SICREMAP2M13	H'FF1FC434	H'FDA00000	H'E6000000	GPIO/PFC/CPGA/Reset/SYSC (*1)
SICREMAP2M14	H'FF1FC438	H'FDC00000	H'E6600000	Secure Engine (*1)
SICREMAP2M15	H'FF1FC43C	H'FDE00000	H'EB200000	RT-VRAM0 mirror space

(*1) When using Secure boot API, please do not change prohibition.

(3) Set MPU registers.

MPU is a memory protection function of ICUMX, and memory is protected by granting execute / read / write permission. Protection is set at the start of the Boot ROM program and released at the end of the Boot ROM program's processing.

Execution permission: Boot ROM code execution area

Write permission: IPL download area, registration area, stack area, work area, hash / log protection area

Read permission: Boot ROM code execution area, IPL download area, registration area, stack area, work area, hash / log protection area

No permission: Other areas

(4) Set Region ID registers and Internal RAM protection. (Refer to Table 9-10 Region ID settings)

Table 9-10 Region ID settings

IP (master)	Region ID	Internal RAM protection
ICUMX	0	All permissions
Secure Engine	2	Read permission for Boot ROM and IPL Download area and write permission for Stack / Work area
RT-DMAC (Use channel)	3	Write permission to the IPL Download area
Others	15	All write-protected

(5) Set H'FDF00000 (RT-VRAM) to the stack pointer.

Cortex-R52

(1) Set H'0 to general-purpose registers (R0-R12).

(2) Set H'EB300000 to the stack pointer.

(3) Enable MPU. For details about MPU settings, refer to 6. Realtime Core section of R-Car V4H UM.

(4) Enable I-cache. Enable branch prediction setting.

9.2.2 PFC configuration

PFC configuration for the HyperFlash and Serial Flash are used by initial setting value of HW.

PFC configuration for eMMC boot is set by Boot ROM program.

PFC configuration on Boot ROM program is described in R-Car V4H HW UM. Refer to section 7 of R-Car V4H HW UM.

9.2.3 Boot device

The Boot ROM program refers to the MD pin, BCFG2[3:0] settings and LCM state to select the device to use for booting.

The relationship between the BCFG2[3:0] setting and the boot device selection according to the LCM state is shown below.

Table 9-11 Configurations of boot device selection

Boot CPU	Boot mode (See 9.1.2)	BCFG2[3:0] setting status (*1)	BCFG2[3:0] and MD [4: 1] match /unmatched (*1)	Boot device selection
Cortex-R52	Normal	-	-	Selected device (MD[4:1]) See section 31 of R-Car V4H HW UM
ICUMX	Secure	not B'0000	match	Selected device (MD[4:1]) See Table 5-2 Boot device selection at SE(Secure)
			unmatched	SCIF download or No BOOT See 9.2.5(1) Data Transfer Sequence in Secure Boot Mode Retry function
		B'0000	-	SCIF download or No BOOT See 9.2.5(1) Data Transfer Sequence in Secure Boot Mode Retry function
	Normal	not B'0000	match	Selected device (MD[4:1]) See section 31 of R-Car V4H HW UM
			unmatched	SCIF download or No BOOT See 9.2.5(1) Data Transfer Sequence in Secure Boot Mode Retry function
		B'0000	-	Selected device (MD[4:1]) See section 31 of R-Car V4H HW UM

*1:See 9.7.4.

(a) Boot device detection for normal boot sequence

Normal Boot sequence on Boot ROM program is described in R-Car V4H HW UM. Refer to section 31 of R-Car V4H HW UM.

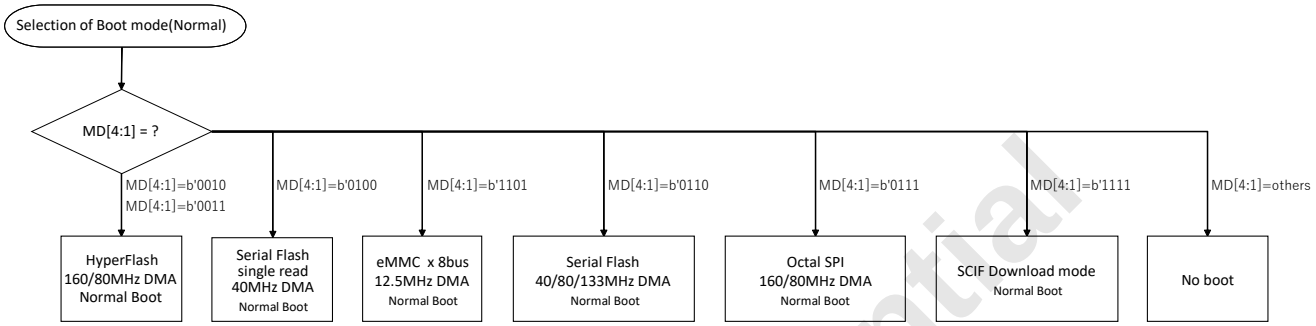


Figure 9-9 Boot mode detection of Normal Boot mode

(b) Boot device detection for secure boot sequence

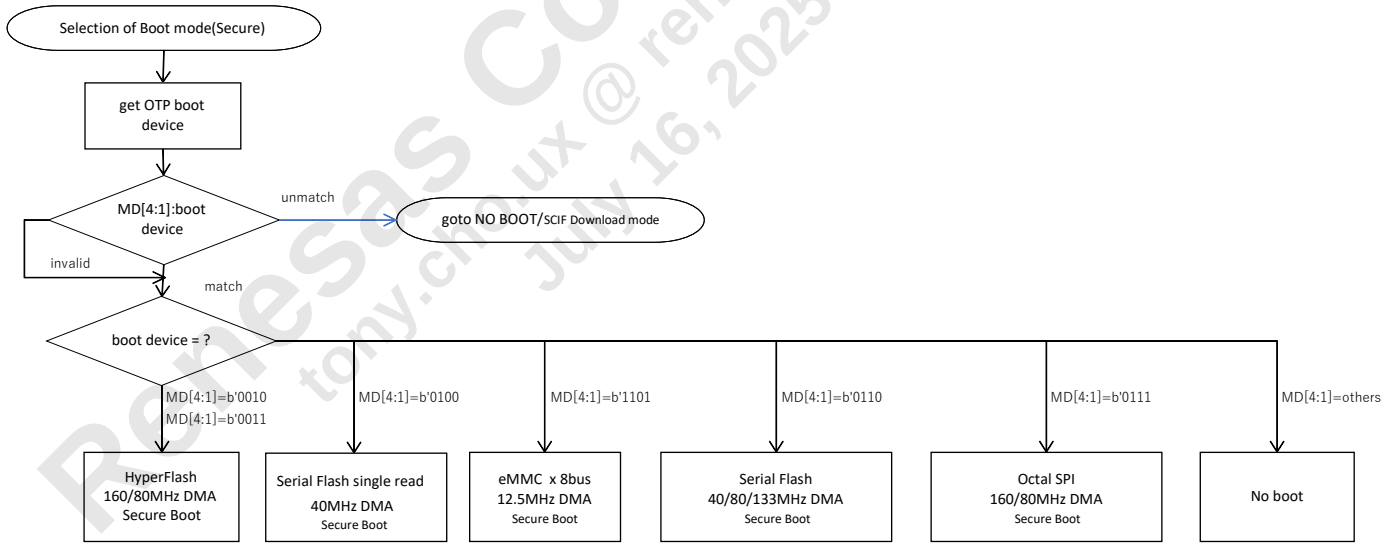


Figure 9-10 Boot device detection of Secure Boot mode

9.2.4 Data Transfer Sequence in Normal Boot Mode

Normal Boot sequence on Boot ROM program is described in R-Car V4H HW UM. Refer to section 31 of R-Car V4H HW UM.

9.2.5 Data Transfer Sequence in Secure Boot Mode

Secure Boot sequence on Boot ROM program is described below.

1. ICUMX executes Boot ROM program, and Boot ROM parameters, Secure debug certificate (C&R key certificate), Secure boot key certificate, IPL content certificate are transferred from external boot device to Internal RAM.
2. Boot ROM program checks the range of load area from address and size information in content certificate.
   2-1. Load the certificate on the 1st /2nd side.
   2-2. Check the MAGIC (*1) of both certificates to see if the certificate is included.
   2-3. When the 1st side certificate is valid.
      Performs 1st side certificate validation
      If the certificate validation is successful, specify 1st side as the boot side.
   2-4. When the 2nd side certificate is valid.
      Performs 2nd side certificate verification
      If the certificate validation is successful, specify 2nd side as the boot side.
   2-5. When both certificates on the 1st /2nd side are valid.
      Performs certificate validation on the 1st side.
   2-6. If the verification of the certificate on the 1st /2nd side is successful, specify the side that succeeded in verification as the boot side.
      Compare both 1st /2nd SW versions to determine the boot side.
      Save the other side as the sub boot side.
3. In case of error or comparison failure. After that, retry processing is performed.
4. If there is no error, the IPL image will be transferred to the address specified by Internal RAM.
   4-1. Loads the IPL image of the specified boot surface.
   4-2. If the image is encrypted, decrypt the image.
   4-3. After transferring the data (IPL image), the Boot ROM program checks the data integrity to make sure that it has not been tampered with.
5. When an error occurs and retry processing is performed.
   Retry loading, decrypting, and integrity of the IPL image on the sub boot surface.
6. Secure Debug function is judged before jump to IPL.
   Refer to 9.4 for details.
7. If there is falsification of the other IPL, Boot ROM program jumps to SCIF Download mode or no boot.

(*1) MAGIC mean reserved value which specified by certificate format.

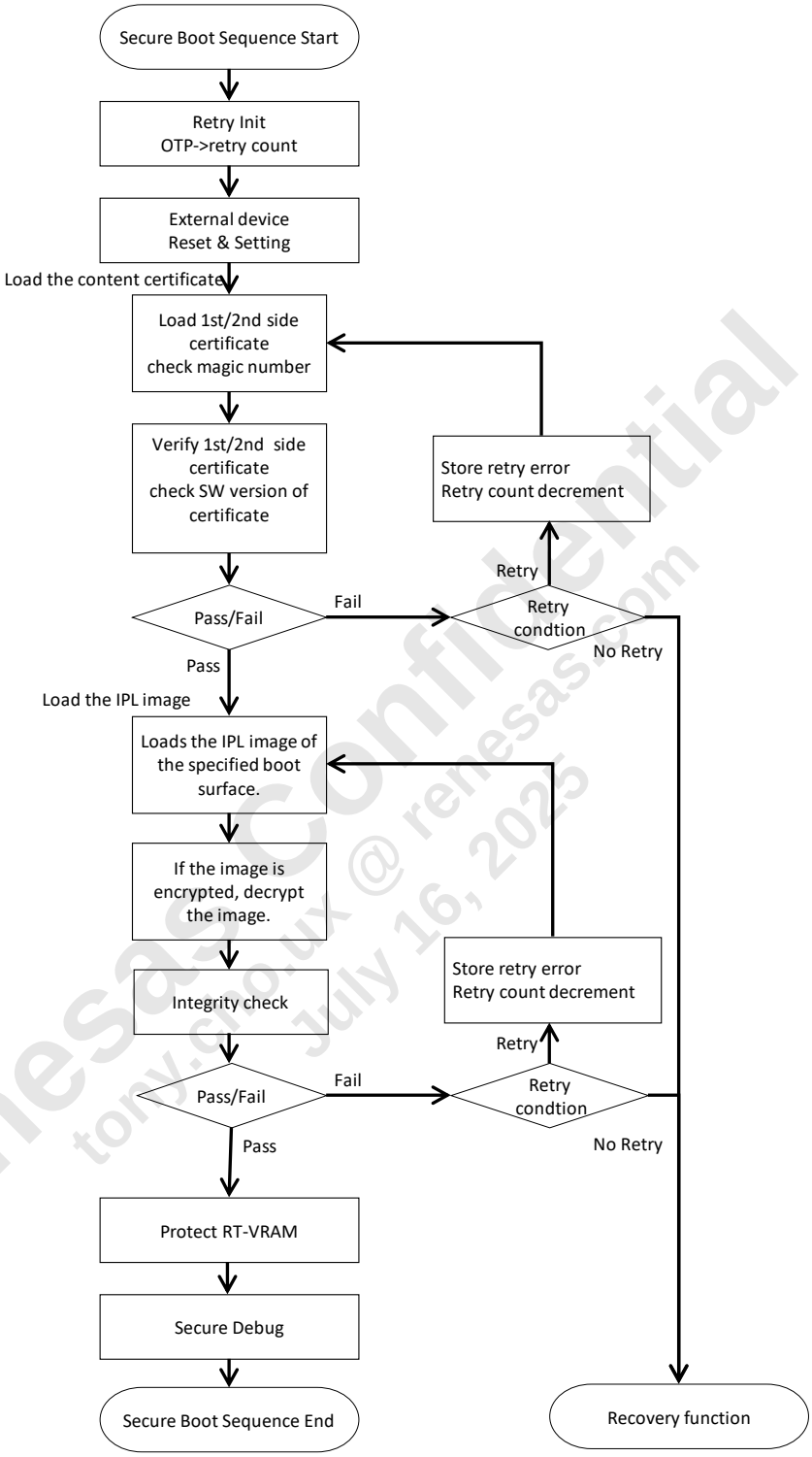


Figure 9-11 Secure Boot sequence

(1) Retry function

The number of retries is determined by referring to the value written in BCFG2[10:8]. (See 9.7.4.)

The 3 bits are treated as a binary number, and the number of retries from 0 to 7 (B'000-B'111) is determined.

The retry process is closely related to the sequence of 1st/2nd side certificate verification and IPL image verification.

After booting Boot ROM and determining LCS, the external memory selects.

When the external memory is HyperFlash, SPI (Single, Quad, Octal), eMMC, 1st / 2nd side switching and retry function are executed.

(2) SW version

In comparing with the SW version of the 1st/2nd certificates, to determine the IPL to be booted.

The SW version includes the SW version type (NV counter ID) and value (NV counter value) in each certificate, and when the certificate is verified, it is compared with the value written in TFMV[127:0] to determine whether the certificate is available.

The SW version value is represented by the total number of bits (0 to 128) in which each bit of the TFMV value is set to 1.

Table 9-12 How to read the SW version

TFMV[127:0]	SW version(dec.)
H'0000 0000 0000 0000 0000 0000 0000 0000	0
H'0000 0000 0000 0000 0000 0000 0000 0001	1
H'0000 0000 0000 0000 0000 0000 0000 0003	2
H'0000 0000 0000 0000 0000 0000 0000 FFFF	16
H'0000 0000 0000 0000 0000 0000 FFFF FFFF	32
H'0000 0000 0000 0000 FFFF FFFF FFFF FFFF	64
H'7FFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF	127

The reference method of the SW version value written in TFMV and TFMV for the table is shown below.

See Table 8-2 e-FUSE/OTP related registers List.

Table 9-13 Configurations of SW version

SW version	type	Register	bit	Size (bit)	Explanation
TFMV	1	TFMV0	[31:0]	128	Used when authenticating the IPL. If it is executed from IPL, an error will occur.
		TFMV1	[63:32]		
		TFMV2	[95:64]		
		TFMV3	[127:96]		
TFMV for the table	3	TFMV_TABLE0	[31:0]	128	Used when calling the secure boot API with an IPL certificate
		TFMV_TABLE1	[63:32]		
		TFMV_TABLE2	[95:64]		
		TFMV_TABLE3	[127:96]		
NTFMV for the table	4	NTFMV_TABLE0	[31:0]	128	Used when calling the secure boot API with an IPL certificate
		NTFMV_TABLE1	[63:32]		
		NTFMV_TABLE2	[95:64]		
		NTFMV_TABLE3	[127:96]		

SW version check judgment condition

Condition 1

If the SW version type in the certificate is different from TFMV, it is judged that the corresponding certificate is not used (verification failure).

Condition 2

Table 9-14 OTP's and certificate TFMV condition

TFMV condition	result
OTP: TFMV <= certificate :TFMV	Pass
OTP: TFMV > certificate :TFMV	Failure
127 < certificate :TFMV	Failure

If both 1st / 2nd side certificates are successfully checked with OTP's TFMV, the IPL to be activated is as follows.

IPL on the face of the certificate with a large TFMV value

If TFMV is equivalent, IPL on 1st side

Condition 3

Both the Secure boot key certificate and the Secure boot content certificate include the SW version.

Both SW versions are considered to be authenticated successfully if the following conditions are met.

Secure boot key certificate : TFMV <= Secure boot content certificate : TFMV

9.2.6 Recovery Function

If the data transfer fails and does not retry, the Boot ROM program jumps to SCIF download mode or does not boot. If recovery (BCFG1[9:8]) is 2'b00, it jumps to SCIF download mode, otherwise it jumps to No BOOT. (See 9.7.3)

After jumping, an error message is output. (See 9.5 for error messages)

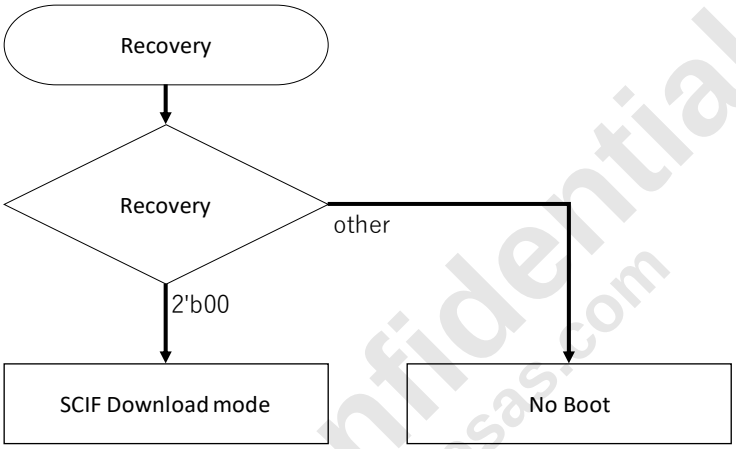


Figure 9-12 Recovery Function

9.2.7 Cleanup

ICUMX

(1) RT-DMAC registers are set to hardware initial value.
(2) Region ID registers are set to hardware initial value.
(3) MPU settings are kept at the CPU Initialization value.
(4) Clear the stack area.
(5) Set H'0 to general-purpose registers (R1-R9, R11-R29, EIPC, CTPC).
Note

- The settings of OS timer module and REMAP module are not cleaned in Boot ROM.
- The settings of RPC module for the transfer is automatically set by MD pin settings. Therefore, registers of RPC module are not cleaned in Boot ROM.
- In the case of eMMC boot, initialize the SDHI interface registers.
- FE level maskable interrupt (FEINT) and EI level maskable interrupt (EIINT) are always disabled.

Cortex-R52

(1) RT-DMAC registers are set to hardware initial value.
(2) Disable I-cache. Disable branch prediction setting.
Note

- The PFC setting is not cleared after jumping to the IPL.
- The settings of RPC module for the transfer is automatically set by MD pin settings. Therefore, registers of -RPC module are not cleaned in Boot ROM.
- In the case of eMMC boot, initialize the SDHI interface registers.
- IRQ and FIQ are disable.
- MPU settings are kept at the CPU Initialization value.

9.3 SCIF download mode

This chapter describes SCIF download mode for R-Car V4H.

9.3.1 Functions

SCIF download mode uses the serial port, to transfer the data from Host PC to Internal RAM. And SCIF download mode can be selected Normal SCIF download sequence and Secure SCIF download sequences depend on LCM state. If selected Secure SCIF download mode, the software transferred from Host PC is verified in Boot ROM program.

The SCIF download mode supports SCIF and HSCIF.

In the SCIF download mode, HSCIF baud rate can be selected by mode pin.

Table 9-15 Transfer specification

Item	Description
Baud rate	See Table 9-16
Data length	8bit
Parity	none
Stop bit	1bit
Flow control	none
Data format	Motorola S-record

Table 9-16 Mode Pin Configuration

MD32	MD31	Mode	Baud rate (bps)	Input Clock	Pin voltage
0	0	SCIF Boot	115200	Internal	TX0/RX0 3.3Volt
0	1	HSCIF boot 1	921600	Internal	HTX0/HRX0 3.3Volt
1	0	HSCIF boot 2	1843200	Internal	HTX0/HRX0 1.8Volt

- **Normal SCIF download sequence**

In this sequence, since secure boot is disabled, the software transferred from Host PC is not verified in Boot ROM.

- **Secure SCIF download sequence**

In this sequence, since secure boot is enabled, the software transferred from Host PC is verified in Boot ROM. After transferring the data, the program does integrity check of the data, and confirms there is no falsification. If there is no falsification of the data, the program executes the address of the transferred data. If there is falsification of the data, the program jumps to Secure SCIF download sequence again. If the failed 3 times to integrity check, the program jumps to No boot.

9.3.2 Download sequence

Figure 9-13 shows a flowchart of SCIF download mode.

When LCM state is SE (Secure), Secure SCIF download sequence is forcibly executed.

When LCM state is CM, DM, and FA, whether Normal or Secure SCIF download sequence is selected by MD5 pin.

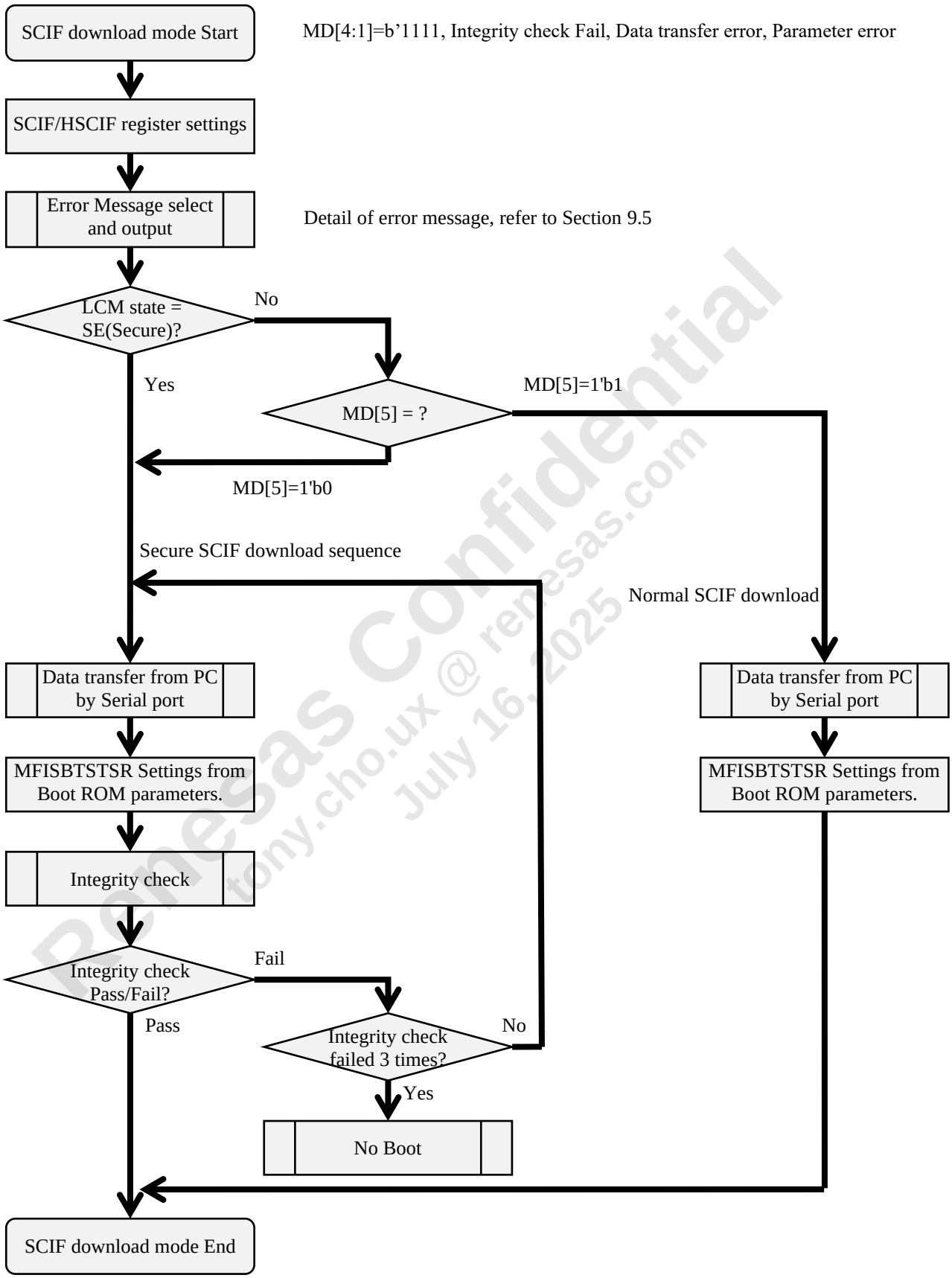


Figure 9-13 SCIF download mode flowchart

9.3.3 Transfer Data Format

This section describes the data format of software transferred from Host PC.

The data format is Motorola S-record.

Refer to Figure 9-14 for memory allocation.

1. Header Record

Regard to Header Record of top, it can be set comment or file name. It is necessary to create by using S0 type. And regard to Header Record of end, it can be set the address program is executed. It is necessary to create using S7 type.

2. Secure boot key cert.

It is necessary to create from H'EB20 1000 by using S3 type.

This field is used for integrity check at secure boot.

3. IPL content cert.

It is necessary to create from H'EB20 3000 by using S3 type.

If using Normal SCIF download mode, transfer address and transfer size must be set, and Magic Number and Key Flags field must be set to 0.

4. Secure debug cert for Non-secure.

It is necessary to create from H'EB20 4000 by using S3 type.

This field is used for secure debug authentication and C&R authentication.

5. Secure debug cert for Secure.

It is necessary to create from H'EB20 6000 by using S3 type.

This field is used for secure debug authentication and C&R authentication.

6. IPL image

This is the program executed. It is necessary to create by using S3 type.

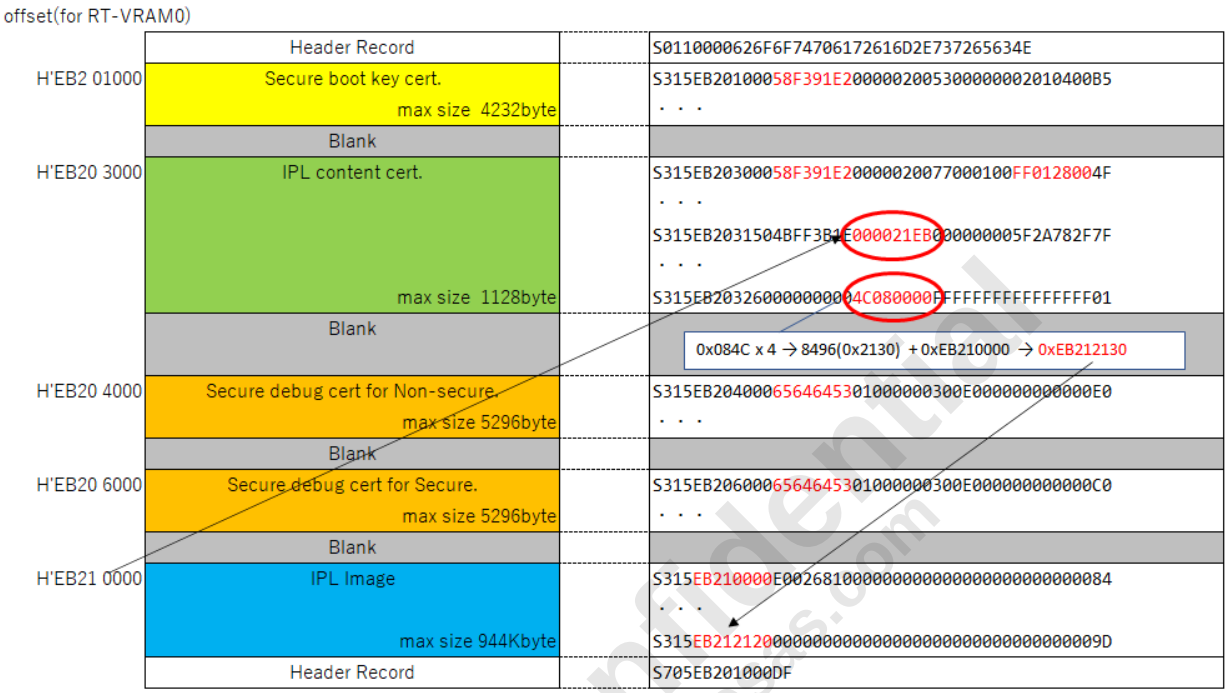


Figure 9-14 Data format for Transfer data

9.4 Secure Debug Function

The Secure Debug function determines whether or not the debugger can be connected through three-step sequence.

9.4.1 Features

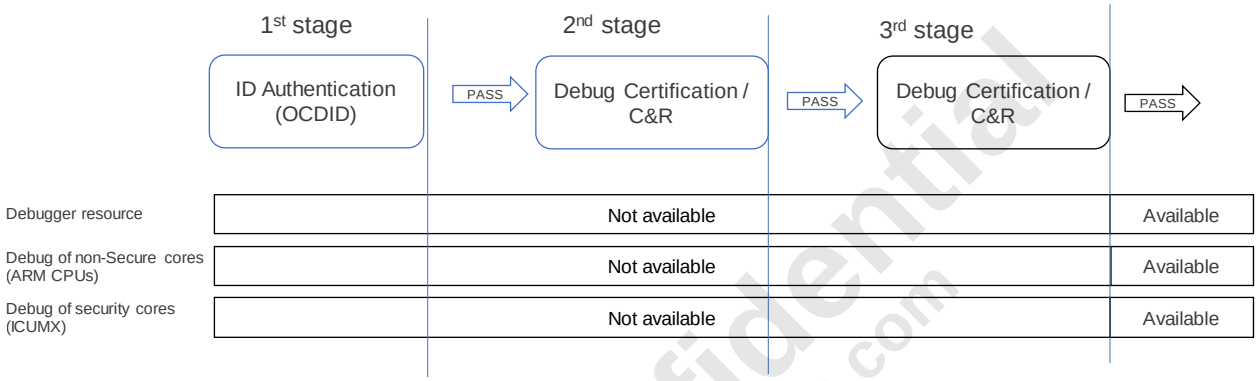


Figure 9-15 Secure debug 3-step authentication

1st stage

Boot ROM checks the result of the comparison between the 256-bit OCDID programmed by OTP memory and the ID entered by the Debugger. If the IDs match, three-step sequence will transition to the next stage.

2nd stage

Determines whether the debugger can be connected to the non-Secure CPU core.

Whether the connection is possible or not can be performed only by certificate verification using the Secure Debug certificate, or the pattern of executing certificate verification and C&R (Challenge and Response) authentication in combination can be switched by specifying e-FUSE.

Non-Secure CPU targets the following.

- Arm core (Cortex-A76, Cortex-R52)

3rd stage

Determines whether the debugger can be connected to the Secure CPU core.

Whether the connection is possible or not can be performed only by certificate verification using the Secure Debug certificate, or the pattern of executing certificate verification and C&R (Challenge and Response) authentication in combination can be switched by specifying e-FUSE.

Secure CPU targets the following.

- ICUMX

9.4.2 Secure debug certificate

When Secure Debug certificate authentication is selected by e-FUSE, the Boot ROM program sets whether the debugger can be connected according to the contents of the Secure Debug certificate stored in the external Flash.

RSA digital signatures are used to validate secure debug certificates.

The Debug certificate is divided into a CPU in Normal mode and a CPU in Secure mode. (See 9.1.4 Memory Map)

Table 9-17 Secure debug certificate

Kinds	Contents
Non-secure debug certificate	Certificate for Non-Secure CPU
Secure debug certificate	Certificate for Secure CPU

9.4.3 Secure Debug Certificate

Figure 9-16 shows flowchart of secure debug authentication by Secure Debug certificate in Boot ROM program.

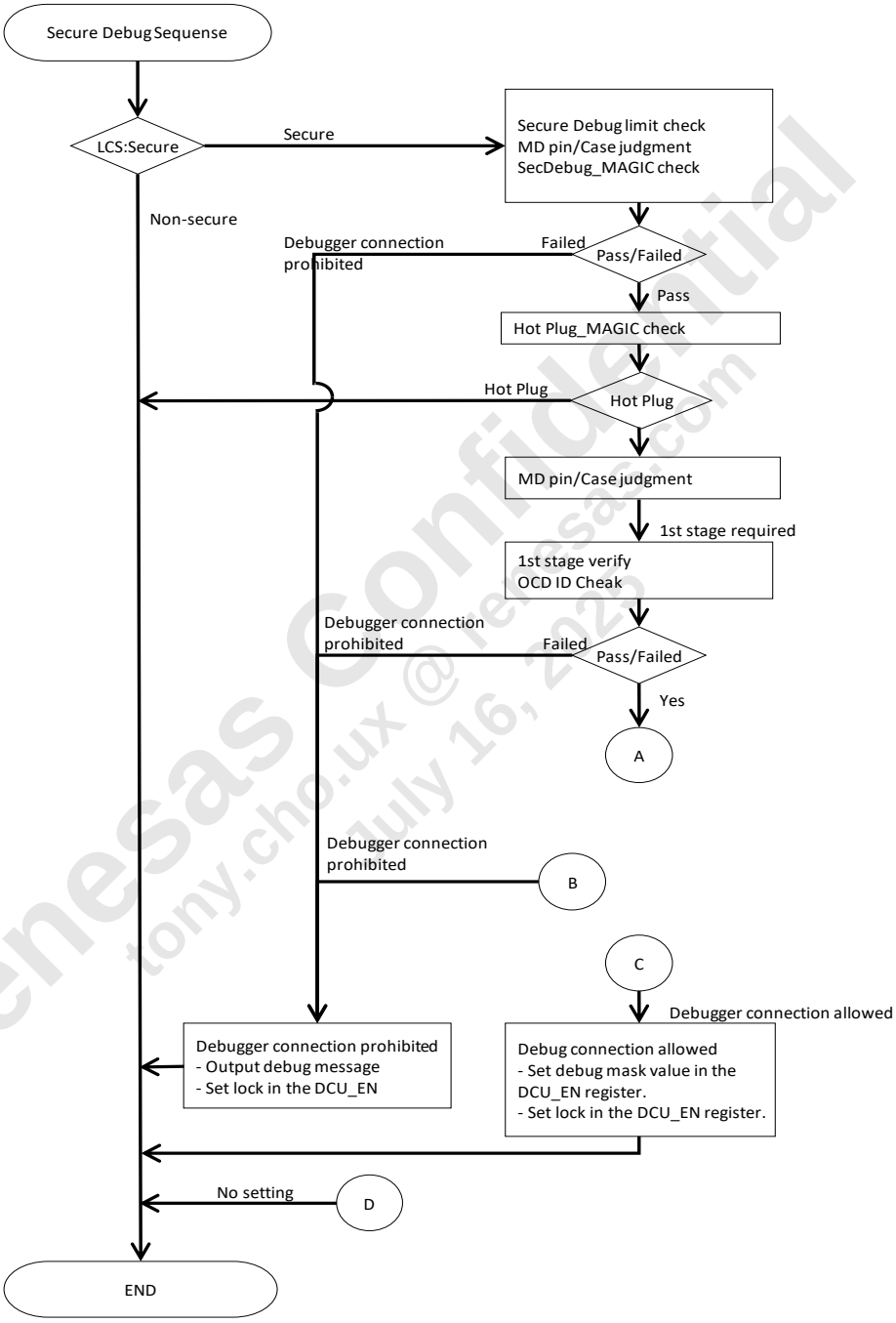


Figure 9-16 Secure Debug Certificate flowchart

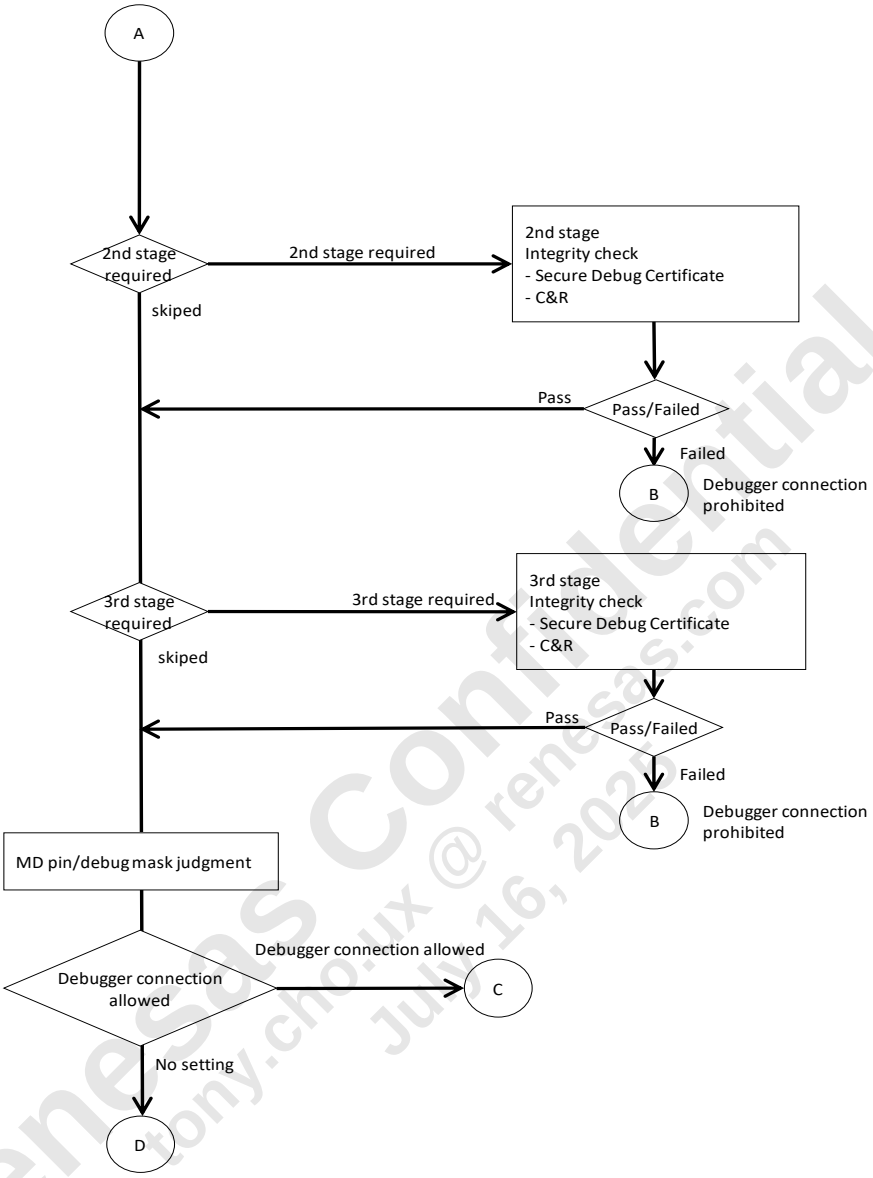


Figure 9-17 Secure Debug Certificate flowchart(continuation)

9.4.4 Challenge & Response

In the Boot ROM program, Challenge & Response(C&R) integrity check is executed after passing authentication with Secure Debug Certificate. When both C&R and certificate authentication integrity check succeeds, debugger access via JTAG connection is enabled.

The following indicates the feature of the C&R.

- One-time authentication by using the C&R integrity check.
- Integrity check is executed in C&R integrity check program that is executed in the Boot ROM.
- Challenge data is generated as a random value in 128bits and send to host PC via JTAG port.
- Algorithm for generating Response data form challenge data is RSA algorithm.
- The Developer public key used to authenticate response data is stored to Secure debug certificate in External Flash. The Key length for RSA is selected from 2048bits, 3072bits, and 4096bits, which should be the same as the key length of the secure boot certificate.

Figure 9-18 shows C&R sequence between Host PC and Boot ROM.

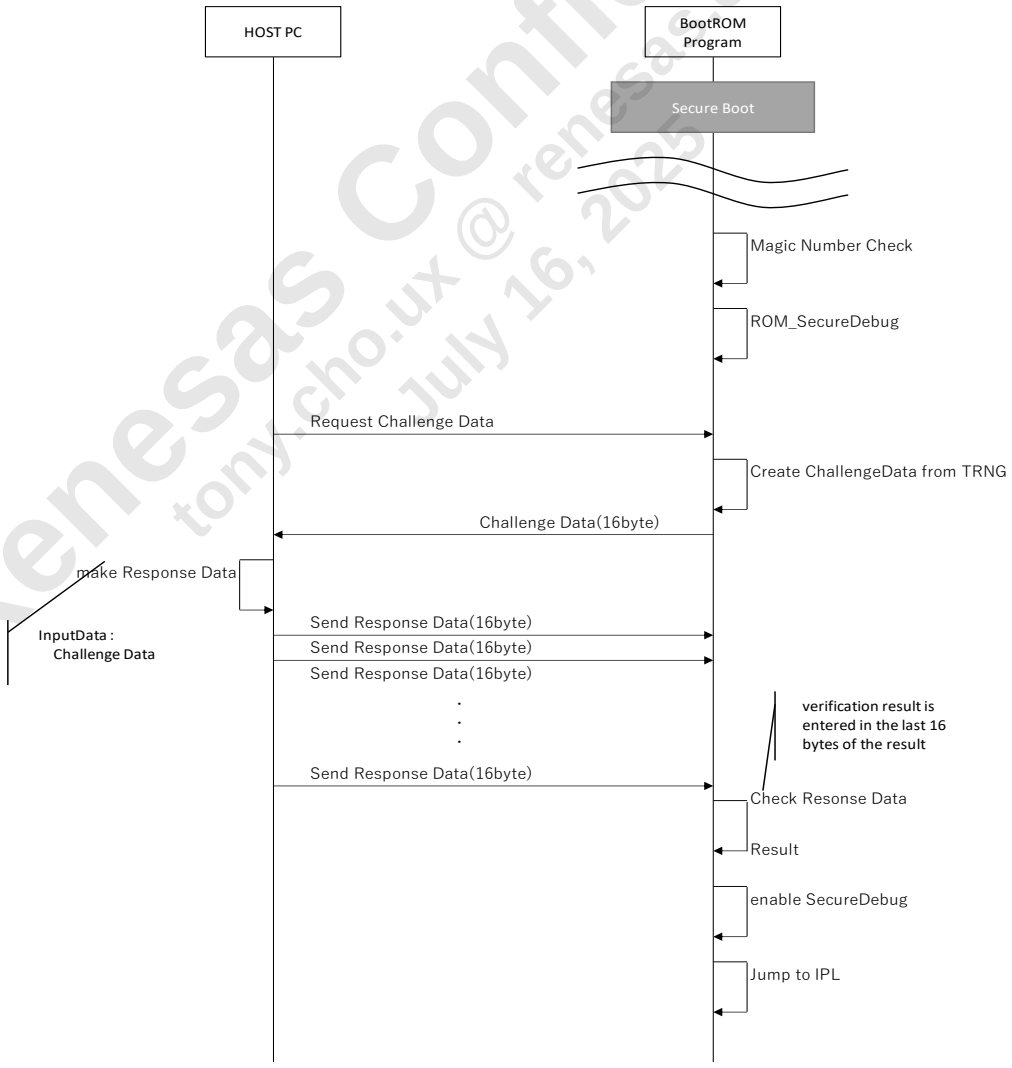


Figure 9-18 C&R sequence between Host PC and Boot ROM

9.4.5 Secure debug message

The Boot ROM program outputs debug messages to detect the cause of secure debug errors.

Table 9-18 Secure debug message shows the relationship between error message and error causes.

Table 9-18 Secure debug message

No.	Error message	Cause
1	Secure debug mode Not Support	Secure Debug function is disabled by BCFG1[11:10] and ICUMX_CFG4.
2	No applicable case	Incorrect JTAG selection by MD pin setting.
3	IDLOCK is Locked	The ID entered by the debugger did not match the OCDID.
4	NonSecure debug certificate verification Failed	The Secure Debug certificate for Non-secure is incorrect.
5	NonSecure C&R verification Failed	Challenge & Response integrity check for Non-secure is failed.
6	NonSecure debugmask mismatch	The Debug-mask of the Non-secure debug certificate and the MD pin setting did not match.
7	NonSecure debugmask undefined bit setting	An undefined bit is set in the Debug-mask in the Secure Debug certificate for Non-secure.
8	NonSecure debugmask necessary bit not setting	The necessary bits are not set in the Debug-mask in the Secure Debug certificate for Non-secure.
9	Secure debug certificate verification Failed	The Secure Debug certificate for Secure is incorrect.
10	Secure C&R verification Failed	Challenge & Response integrity check for Secure is failed.
11	Secure debugmask mismatch	The Debug-mask of the Secure debug certificate and the MD pin setting did not match.
12	Secure debugmask undefined bit setting	An undefined bit is set in the Debug-mask in the Secure Debug certificate for Secure.
13	Secure debugmask necessary bit not setting	The necessary bits are not set in the Debug-mask in the Secure Debug certificate for Secure.

9.4.6 Hot Plug Function

The Boot ROM program checks the IPL's magic number to determine if the Hot Plug feature is requested.

The IPL's magic number is stored the last 4 bytes of the IPL in little endian.

If the magic number matches the value (0x853F912E), it is determined that the Hot Plug feature is available.

If the Hot Plug feature is enabled:

The Boot ROM program does not change the debug settings.

The debug settings must be performed at IPL or later.

Secure debug setting status of the Boot ROM program:

No error in Secure debug settings.

The debug mask value setting is not set in DCU_EN register.

The DCU_EN register is not locked.

Required processing on the IPL side:

IPL requires Secure debug integrity check.

Set the debug mask value in the DCU_EN register.

Set the lock on the DCU_EN register.

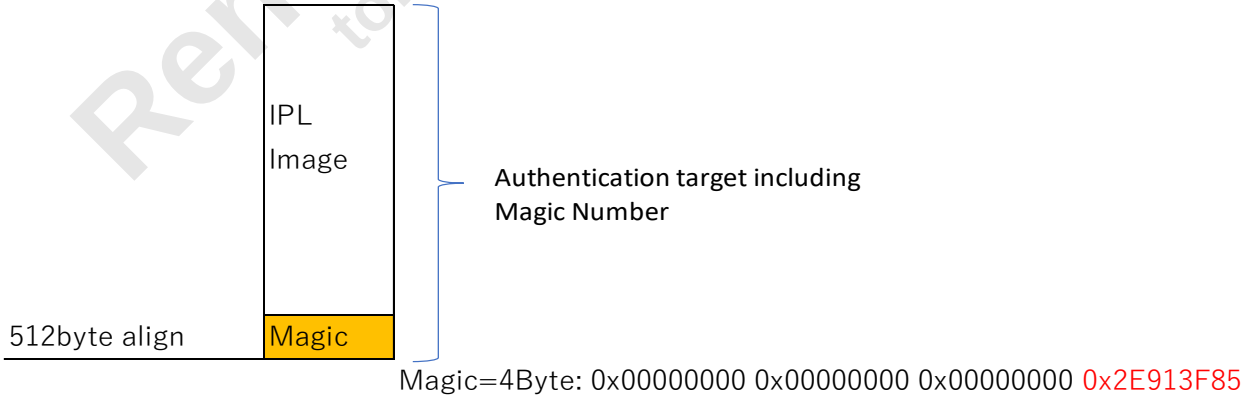


Figure 9-19 Hot Plug

9.5 Error Handling

If the Boot ROM program detects an error occasion in the loading IPL function, stop booting. And the Boot ROM program is output the error message from the serial port.

Figure 9-20 shows the format of error message.

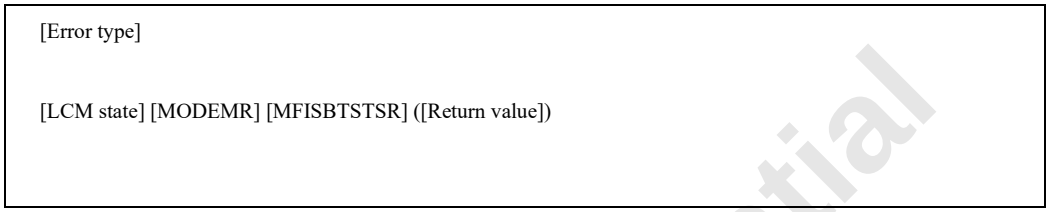


Figure 9-20 Error massage format

[Error Type]
- No Boot
- Verification Failed
- Parameter Error

[LCM state]

CM	: H'00000000
DM	: H'00000010
SE(Secure)	: H'00000050
FA	: H'000000F2

[MODEMR]
Mode Monitor Register (MODEMR) value.
For details, refer to section 14 of R-Car V4H HW UM.

[MFISBTSTSR]
Boot Status Register (MFISBTSTSR) value.
For details, refer to Section 9.7.2.

[Return value]
This value is error code for "Verification Failed" or "Parameter Error" and is invalid for other errors.
Table 9-24 ROM_SecureBootVerify and Table 9-26 ROM_SecureBootCompare show the relationship between error code and error causes.

(1) No Boot

This error occurs the following cases:

- Boot device selection (MD[4:1]) is reserved or No boot when BCFG1[9:8] is selected as “No Boot”.
- Integrity check failed three times in SCIF Download mode.
- Integrity check failed when BCFG1[9:8] is selected as “No Boot”.

By default, this error does not occur because BCFG1[9:8] is selected as SCIF download.

```
No Boot
0x00000050 0xA0010180 0x00000008
```

Figure 9-21 Output message example (No Boot)

(2) Verification Failed

This error occurs in cases of verification failure in Secure Boot.

Refer to Table 9-24 ROM_SecureBootVerify and Table 9-26 ROM_SecureBootCompare a about return value.

```
Verification Failed!!!
0x00000050 0xA0010184 0x0000000C 0xF1000006

 SCIF Download mode (with verification)
 (C) Renesas Electronics Corp.

-- Load Program to RT-SRAM ---------------
please send !
```

Figure 9-22 Output message example SCIF download mode (Verification Failed)

(3) Parameter Error

This error occurs the following cases:

- IPL address and size are out of range. In this case the return value is 0x00000001 or 0x00000002.
- Key length in content cert is invalid value. In this case the return value is 0x00000003.
- In case of boot device selection is enabled (BCFG2[3:0] is not zero), MD pin settings (MD[4: 1]) and boot device settings (BCFG2[3:0]) do not match. In this case the return value is 0xFFFFFFFF.

```
Parameter Error!!
0x00000050 0xA0010184 0x0000000C 0x00000002
 SCIF Download mode (with verification)
 (C) Renesas Electronics Corp.

-- Load Program to RT-SRAM ---------------
please send !
```

Figure 9-23 Output message example (Parameter Error)

9.6 How to Call Secure Boot Function from Software

Boot ROM program has the Secure Boot function. This function is available not only from the “launching IPL” function but also from software. For example, the “verified” IPL calls the Secure Boot function in order to verify OS kernel binary. This section describes how to call the Secure Boot function from software.

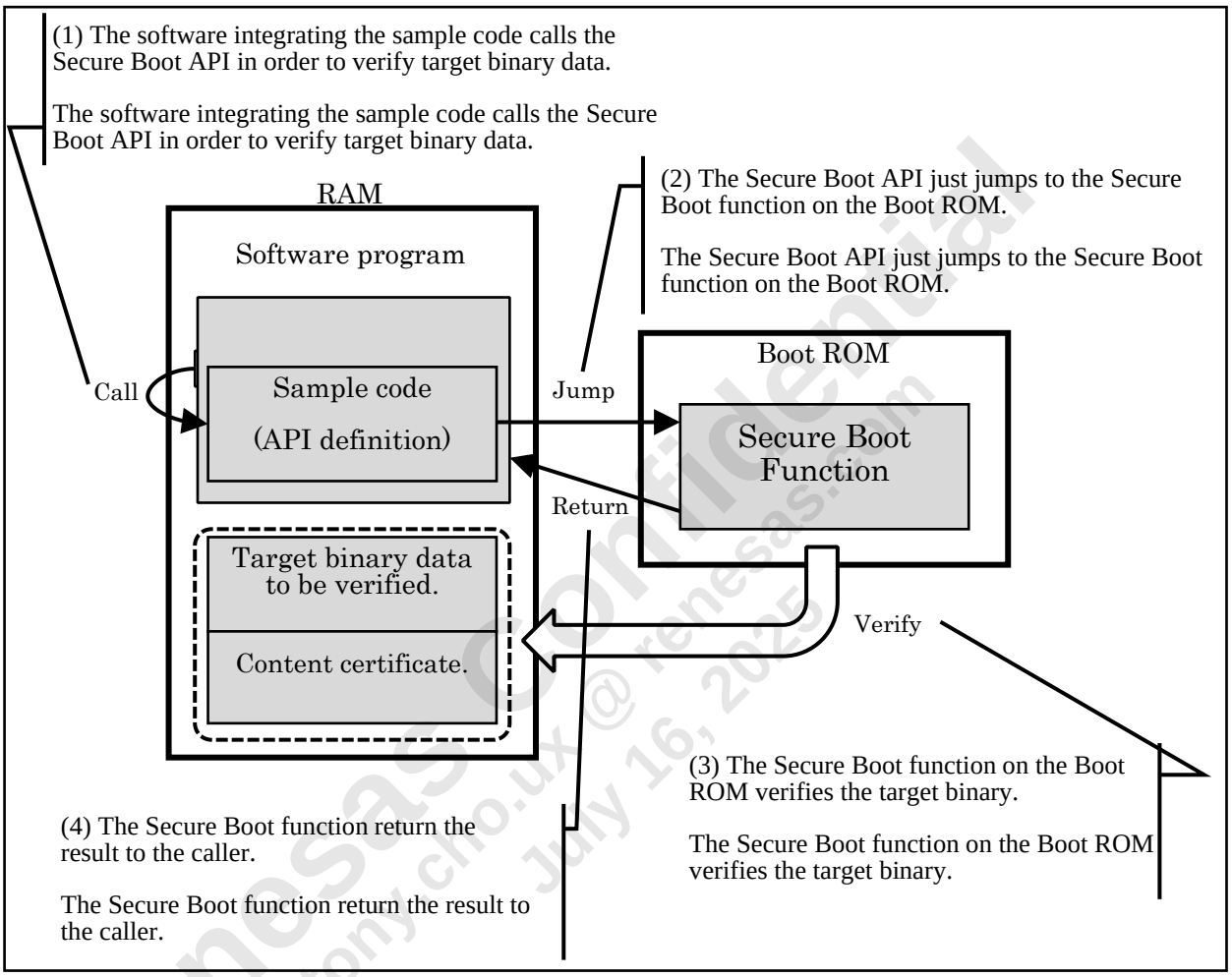


Figure 9-24 Calling secure boot function from software

9.6.1 Secure boot API specification

This section describes Secure boot API specifications.

Table 9-19 Address of secure boot API

	Secure Boot API	ICUMX	Arm AArch32
R-Car V4H	ROM_GetSOC_ID()	H'0110 4408	Not support
	ROM_GetLcs()	H'0110 4418	Not support
	ROM_Crypto()	H'0110 44C0	Not support
	ROM_SecureBootVerify()	H'0110 44C8	Not support
	ROM_SecureBootDecrypt()	H'0110 44D0	Not support
	ROM_SecureBootComapre()	H'0110 44D8	Not support
	ROM_SecureDebug()	H'0110 44E0	Not support

Secure boot API runs at the Code Fetch Remap Access area. Therefore, do not change the ICUMX_CFREMAP register settings.

About details of the secure boot API function, see Table 9-20 Secure Boot API List, Table 9-21 ROM_GetSOC_ID, Table 9-22 ROM_ GetLCS, Table 9-23 ROM_Crypto, Table 9-24 ROM_SecureBootVerify, Table 9-25 ROM_SecureBootDecrypt, Table 9-26 ROM_SecureBootCompare and Table 9-27 ROM_SecureDebug.

Table 9-20 Secure Boot API List

No	Function name	Explanation
1	uint32_t ROM_GetSOC_ID(ROM_SocId_Result_t socIdBuff, uint32_t size)	This function can get the device's unique SOC_ID as hashed AES_CMAC (KDR) of [HBK].
2	uint32_t ROM_GetLCS(uint32_t *pLcs, uint32_t lcs_size)	This function can get the LCM State.
3	uint32_t ROM_Crypto(uint32_t enc, const uint32_t *iv, const uint32_t *hash, const uint32_t *in, uint32_t *out, uint32_t in_out_size)	This function can encrypt with KCE, and decrypt the input data.
4	uint32_t ROM_SecureBootVerify(uint32_t *pKeyCert, uint32_t *pContentCert)	This function can verify the Key certificate and content certificate.
5	uint32_t ROM_SecureBootDecrypt(uint32_t *pContentCert)	This function can decrypt the image according to verified content certificate.
6	uint32_t ROM_SecureBootCompare(uint32_t *pContentCert, uint32_t *hash, uint32_t hash_size)	This function can verify image according to verified content certificate.
7	uint32_t ROM_SecureDebug(uint32_t *pDebugCertPkg)	This function can verify the Debug certificate.

(1) ROM_GetSOC_ID()

Table 9-21 ROM_GetSOC_ID

Function name	uint32_t ROM_GetSOC_ID(ROM_SocId_Result_t socIdBuff, uint32_t size)	
Function type	Synchronous	
Argument	ROM_SocId_Result_t socIdBuff	The derived SOC ID. Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses. The definition of ROM_Socid_Result_t is as follows: uint32_t[8]
	socIdBuff, uint32_t size	Size of socIdBuff, requires 32 bytes of space.
Return value (Typical example)	0	SOC ID calculation success
	0x0B000001	Error of the illegal input parameter
	0x0B000002	Error of the illegal KDR value
	0x0B000004	Error of the illegal SCP value
	0x0B000008	Error of the illegal OEM hash value
	0x0B000010	Error of the illegal LCM state for operation
	0x0B000100	Error the hash is not programmed / The LCM state is temporary security-disabled by Secure Debug Function.
	0xF2000001	Error of crypto driver input
	0xF5000002	Error of the illegal LCM state value
	0xF6000001	Error of HAL fatal error - Hardware engine fault
Outline	Get SOC ID. There is this function on Boot ROM.	
Notes	Notes on this function are as follows: - It doesn't prescribe the operation when the error occurs. - This function accesses the area remapped with ICUMX_SICREMAP2M13 and ICUMX_SICREMAP2M14 registers. Therefore, do not change these registers from the BootROM settings. - This function uses the Secure Engine, therefore, no other software can use the Secure Engine while executing this function. - The stack area is used in 1 Kbyte. ICUMX Local RAM cannot be used as a stack. - This function only works when the LCS value is SE.	

(2) ROM_GetLcs()

Table 9-22 ROM_ GetLCS

Function name	uint32_t ROM_GetLCS(uint32_t *pLcs, uint32_t lcs_size)		
Function type	Synchronous		
Argument	uint32_t *pLcs	Buffer that returns the current LCM state. Addresses must be aligned on 32-bit boundaries. Addresses must be in the ICUMX Local RAM, RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses. ICUMX Local RAM is specified in physical address.	
		0x00000000	Current LCM state is CM.
		0x00000001	Current LCM state is DM.
		0x00000003	Current LCM state is SD. This state is temporary security-disabled by Secure Debug Function.
		0x00000005	Current LCM state is SE.
		0x00000007	Current LCM state is FA.
	uint32_t lcs_size	Size of pLcs, requires 4 bytes of space.	
Return value (Typical example)	0	Lcs calculation success	
	0x0B000001	Error of the illegal input parameter	
	0x0B000002	Error of the illegal KDR value	
	0x0B000004	Error of the illegal SCP value	
	0x0B000010	Error of the illegal LCM state for operation	
	0xF5000002	Error of the illegal LCM state value	
Outline	Get LCM state. There is this function on Boot ROM.		
Notes	Notes on this function are as follows: - It doesn't prescribe the operation when the error occurs. - This function accesses the area remapped with ICUMX_SICREMAP2M13 and ICUMX_SICREMAP2M14 registers. Therefore, do not change these registers from the BootROM settings. - The stack area is used in 1 Kbyte.		

(3) ROM_Crypto()

Table 9-23 ROM_Crypto

Function name	uint32_t ROM_Crypto(uint32_t enc, const uint32_t *iv, const uint32_t *hash, const uint32_t *in, uint32_t *out, uint32_t in_out_size)	
Function type	Synchronous	
Argument	uint32_t enc	0: Encryption with AES-128-CTR algorithm, use Built-in Code Encryption Key (KCE). 1: Encryption with AES-192-CTR algorithm, use Built-in Code Encryption Key (KCE). 2: Encryption with AES-256-CTR algorithm, use Built-in Code Encryption Key (KCE). 3: Decryption with AES-128-CTR algorithm, use Built-in Provisioning Key (KOEM).
	const uint32_t *iv	Top address of the AES initialization vector. Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses. In case of encryption, split encryption is possible by incrementing iv for each block (16 bytes / block).
	uint32_t *hash	In case of encryption, top address of the SHA-256 hash of input plaintext data. In case of decryption, top address of the SHA-256 hash of output plaintext data. Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
	const uint32_t *in	In case of encryption, top address of the input plaintext data. In case of decryption, top address of the input ciphertext data. Addresses must be aligned on 256-byte boundaries. The in and out must point to the same address. Addresses must be in the RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
	uint32_t *out	In case of encryption, top address of the output ciphertext data. In case of decryption, top address of the output plaintext data Addresses must be aligned on 256-byte boundaries. The in and out must point to the same address. Addresses must be in the RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
	uint32_t in_out_size	Input data size (byte). It must be specified as a multiple of 16.
Return value (Typical example)	0x00000000	Encrypt and decrypt success
	0x00000001	Encrypt and decrypt failed (Hash results do not match)
	0x0B000002	Error of the illegal KDR value
	0x0B000004	Error of the illegal SCP value

	0x0B000008	Error of the illegal OEM HASH value
	0x0B000010	Error of the illegal LCM state for operation
	0x0B000020	Error of the illegal SESSION KEY value - Provisioning Key has been used in this session.
	0xF2000001	Error of crypto driver input
	0xF6000001	Error of HAL fatal error - Hardware engine fault
Outline	Encrypt and decrypt the input data. Encryption uses Built-in Code Encryption Key (KCE) The encryption algorithm supports AES-128-CTR, AES-192-CTR and AES-256-CTR Decryption uses Built-in Provisioning Key (KOEM). The decryption algorithm supports AES-128-CTR. There is this function on Boot ROM.	
Notes	Notes on this function are as follows: - Decryption using the Provisioning Key can be executed only once after the LSI reset is released. - This function uses the Secure Engne and ICUMX AES engine, therefore, no other software can use the Secure Engine and ICUMX AES engine while executing this function. - This function accesses the area remapped with ICUMX_SICREMAP2M13 and ICUMX_SICREMAP2M14 registers. Therefore, do not change these registers from the BootROM settings. - The stack area is used in 1 Kbyte. ICUMX Local RAM cannot be used as a stack. - This function works regardless of the LCS value for encryption, and works only when the LCS value is SE for decryption.	

(4) ROM_SecureBootVerify()

Table 9-24 ROM_SecureBootVerify

Function name	uint32_t ROM_SecureBootVerify(uint32_t *pKeyCert, uint32_t *pContentCert)	
Function type	Synchronous	
Argument	uint32_t *pKeyCert	Top address of the secure boot key certificate. Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0/1 (including Extended mode), RT-VRAM0 mirror space, System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
	uint32_t *pContentCert	Top address of the content certificate. Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
Return value (Typical example)	0	Integrity check success / But the case except LCM state SE does not compare it with HBK of the OTP.
	0x0B000001	Error of the illegal input parameter
	0x0B000002	Error of the illegal KDR value
	0x0B000004	Error of the illegal SCP value
	0x0B000008	Error of the illegal OEM hash value
	0x0B000010	Error of the illegal LCM state for operation
	0x0B000100	Error the hash is not programmed / The LCM state is temporary security-disabled by Secure Debug Function.
	0xE0000000	Error of invalid address of certificate
	0xF1000001	Error of INV input param validation
	0xF1000003	Error of cert header validation
	0xF1000004	Error of incorrect cert version num - The certificate was generated with other products signing tool.
	0xF1000005	Error that the SW version is smaller than the minimum SW version - SW version of secure boot key certificate is smaller than SW version of OTP. - SW version of content certificate is smaller than SW version of secure boot key certificate.
	0xF1000006	Error of Public key hash validation / The HBK and the hash calculation result of Root public key in key certificate did not match. / The Secure boot public key hash of Key cert and the hash calculation result of Secure boot public key in content certificate did not match.
	0xF1000007	Error of RSA signature integrity check
	0xF1000008	Error of workspace size too small validation
	0xF100000C	Error of unsupported RSA algorithm
	0xF100000D	Error of the illegal cert version id - Incorrect SW version type in the certificate (other than 3 and 4). - SW version type does not match between secure boot key certificate and content certificate. / Incorrect SW version value in the certificate (greater than 127).
	0xF1000010	Error that illegal number of SW comp
	0xF1000018	Error of incorrect cert type
	0xF1000019	Error of the illegal HBK index
	0xF100001C	Error of the illegal encryption key length

	0xF2000001	Error of crypto driver input
	0xF5000002	Error of the illegal LCM state value
	0xF6000001	Error of HAL fatal error - Hardware engine fault
Outline	Verify the Secure boot key certificate and content certificate using RSASSA-PSS signature verification. The supported RSA key lengths are 2048 bits, 3072 bits, and 4096 bits. Before referencing the image address and image size in the content certificate, it is necessary to verify the certificate using this function. There is this function on Boot ROM.	
Notes	Notes on this function are as follows: - It doesn't prescribe the operation when the error occurs. - This function uses the Secure Engine, therefore, no other software can use the Secure Engine while executing this function. - This function accesses the area remapped with ICUMX_SICREMAP2M13 and ICUMX_SICREMAP2M14 registers. Therefore, do not change these registers from the BootROM settings. - The stack area is used in 3 Kbyte. ICUMX Local RAM cannot be used as a stack. - In case of LCM state = SE, this function verifying the root public key stored into the secure boot key certificate. - In case of LCM state = CM, DM and FA, this function skips verifying the root public key stored into the secure boot key certificate, because the HBK is blank. - The image address and image size in the content certificate must be within RT-VRAM0/1 (including Extended mode), RT-VRAM0 mirror space, System RAM or DRAM (Over4G mirror space) range, and address must be aligned on 512 byte boundary in physical address space.	

(5) ROM_SecureBootDecrypt()

Table 9-25 ROM_SecureBootDecrypt

Function name	uint32_t ROM_SecureBootDecrypt(uint32_t *pContentCert)	
Function type	Synchronous	
	uint32_t *pContentCert	Top address of the content certificate verified by the ROM_SecureBootVerify. Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0/1 (including Extended mode), RT-VRAM0 mirror space, System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
Return value (Typical example)	0	Image decryption success
	0xF1000001	Error of INV input param validation
	0xF1000003	Error of cert header validation
	0xF1000004	Error of incorrect cert version num
	0xF1000008	Error of workspace size too small validation
	0xF1000010	Error that illegal number of SW comp
	0xF1000011	Error that SW comp size is null
	0xF1000018	Error of incorrect cert type
	0xF1000019	Error of the illegal HBK index
	0xF100001C	Error of the illegal encryption key length
	0xF100001D	The image was not encrypted, skipped the decryption process.
	0xF2000001	Error of crypto driver input
Outline	Decrypts the image specified in content certificate. Decryption uses Built-in Code Encryption Key (KCE). The decryption algorithm supports AES-128-CTR, AES-192-CTR and AES-256-CTR. There is this function on Boot ROM.	
Notes	Notes on this function are as follows: - It doesn't prescribe the operation when the error occurs. - This function uses the ICUMX AES engine, therefore, no other software can use the ICUMX AES engine while executing this function. - This function accesses the area remapped with ICUMX_SICREMAP2M13 and ICUMX_SICREMAP2M14 registers. Therefore, do not change these registers from the BootROM settings. - The stack area is used in 3 Kbyte. ICUMX Local RAM cannot be used as a stack. - The image address and image size in the content certificate must be within RT-VRAM0/1 (including Extended mode), RT-VRAM0 mirror space, System RAM or DRAM (Over4G mirror space) range, and address must be aligned on 512 byte boundary in physical address space.	

(6) ROM_SecureBootComapre()

Table 9-26 ROM_SecureBootCompare

Function name	uint32_t ROM_SecureBootCompare(uint32_t *pContentCert, uint32_t *hash, uint32_t hash_size)	
Function type	Synchronous	
	uint32_t *pContentCert	Top address of the content certificate verified by the ROM_SecureBootVerify. Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0/1 (including Extended mode), RT-VRAM0 mirror space, System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
	uint32_t *hash	Top address to store the hash value of the image. Addresses must be aligned on 32-bit boundaries. Addresses must be in the ICUMX Local RAM, RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses. The first 32 bytes are copied hash value from content certificate. The last 32 bytes are the hash value of image calculated by this API.
	uint32_t hash_size	Size of hash, requires 64 bytes of space.
Return value (Typical example)	0	Integrity check success. The hash value of the image matched the hash value of the certificate.
	0xE0000000	Error of invalid address of certificate
	0xF1000001	Error of INV input param validation
	0xF1000003	Error of cert header validation
	0xF1000004	Error of incorrect cert version num
	0xF1000008	Error of workspace size too small validation
	0xF1000009	Error of SW component hash validation / An error when image size is not a multiple of 16. / The image is encrypted with wrong AES key in encrypt boot.
	0xF1000010	Error that illegal number of SW comp
	0xF1000011	Error that SW comp size is null
	0xF1000018	Error of incorrect cert type
	0xF1000019	Error of the illegal HBK index
	0xF100001C	Error of the illegal encryption key length
	0xF2000001	Error of crypto driver input
	0xF6000001	Error of HAL fatal error - Hardware engine fault
Outline	Calculate the hash (SHA-256) value of the image and check if it matches the hash value of content certificate. If they matched, the image has not been falsificated. If not, the image has been falsificated. There is this function on Boot ROM.	
Notes	Notes on this function are as follows: - It doesn't prescribe the operation when the error occurs. - This function uses the Secure Engine, therefore, no other software can use the Secure Engine while executing this function. - This function accesses the area remapped with ICUMX_SICREMAP2M13 and ICUMX_SICREMAP2M14 registers. Therefore, do not change these registers from the BootROM settings. - The stack area is used in 3 Kbyte. ICUMX Local RAM cannot be used as a stack. - The image address and image size in the content certificate must be within RT-VRAM0/1 (including Extended mode), RT-VRAM0 mirror space, System RAM or DRAM (Over4G mirror space) range, and address must be aligned on 512 byte boundary in physical address space.	

(7) ROM_SecureDebug()

Table 9-27 ROM_SecureDebug

Function name	uint32_t ROM_SecureDebug(uint32_t *pDebugCertPkg)	
Function type	Synchronous	
Argument	uint32_t *pDebugCertPkg	Top address of the debug certificate Addresses must be aligned on 32-bit boundaries. Addresses must be in the RT-VRAM0 (including Extended mode), RT-VRAM0 mirror space, RT-VRAM1 (including Extended mode), System RAM or DRAM (Over4G mirror space) range, and they must be specified in SIC Remap addresses rather than physical addresses.
Return value (Typical example)	0	Integrity check success
	0x0B000001	Error of the illegal input parameter
	0x0B000002	Error of the illegal KDR value
	0x0B000004	Error of the illegal SCP value
	0x0B000008	Error of the illegal OEM hash value
	0x0B000010	Error of the illegal LCM state for operation
	0x0B001000	Integrity check fail
	0xF1000001	Error of INV input param validation
	0xF2000001	Error of crypto driver input
	0xF5000002	Error of the illegal LCM state value
	0xF6000001	Error of HAL fatal error - Hardware engine fault
Outline	Verify the Secure debug certificate. There is this function on Boot ROM.	
Notes	Notes on this function are as follows: - It doesn't prescribe the operation when the error occurs. - This function uses the Secure Engine, therefore, no other software can use the Secure Engine while executing this function. - This function accesses the area remapped with ICUMX_SICREMAP2M13 and ICUMX_SICREMAP2M14 registers. Therefore, do not change these registers from the BootROM settings. - The stack area is used in 1 Kbyte. ICUMX Local RAM cannot be used as a stack. - This function only works when the LCS value is SE.	

9.6.2 Sample code using Secure boot API

This section describes attached sample code for using the Secure boot API.

(1) Sample code for using Secure boot API

This sample code describes how to call the Secure boot API.

Refer to the attached sample code "rcar_rom_api.h" and "rcar_rom_api.c".

(2) Sample code for using ROM_GetSOC_ID

This sample code describes how to get SOC ID.

Refer to the attached sample code "rcar_rom_get_socid_sample_code.c".

(3) Sample code for using ROM_GetLcs

This sample code describes how to get LCM state.

Refer to the attached sample code "rcar_rom_get_lcs_sample_code.c".

(4) Sample code for using ROM_Crypto

This sample code describes data encryption using Built-in Code Encryption Key, and data decryption using Built-in Provisioning Key.

Refer to the attached sample code "rcar_rom_crypto_sample_code.c".

(5) Sample code for using ROM_SecureBootVerify, ROM_SecureBootDecrypt and ROM_SecureBootCompare

This sample code describes sequence of authentication the certificate, decryption the encrypted image, and authentication the image.

Refer to the attached sample code "rcar_rom_secure_boot_sample.c".

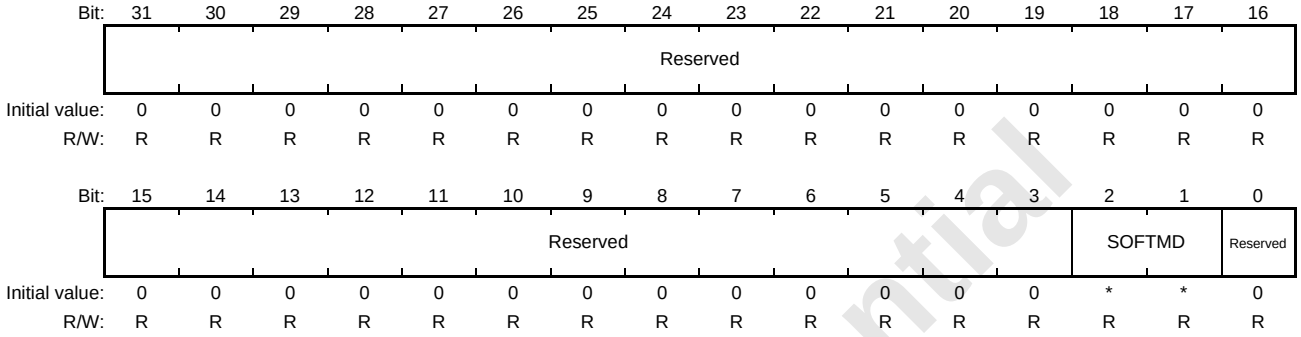
(6) Sample code for using ROM_SecureDebug

This sample code describes how to authenticate the Secure Debug certificate.

Refer to the attached sample code "rcar_rom_secure_debug_sample.c".

9.7 Registers

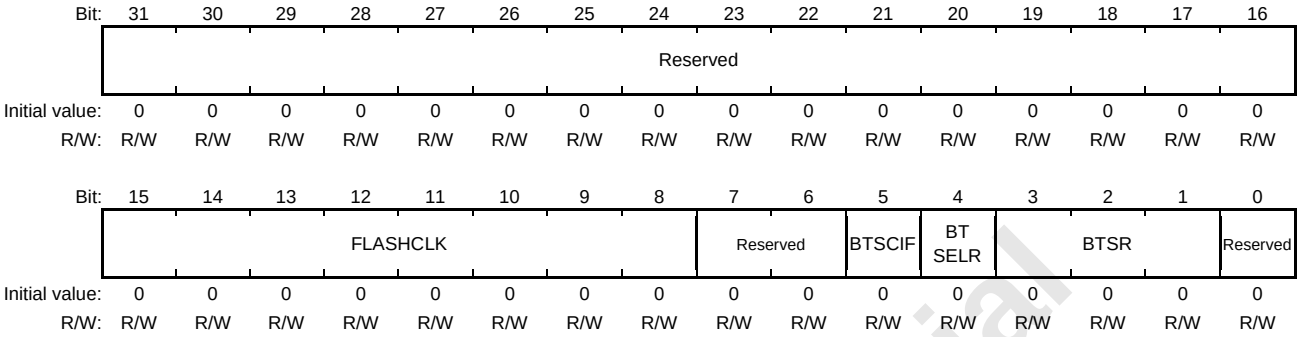
9.7.1 Soft MD Register (MFISSOFTMDR: H'E6260600)



Bit	Bit Name	Initial Value	R/W	Description
31 to 3	-	29'b 0	R	Reserved
2 to 1	SOFTMD[2:1]	*	R	Flash Reset Behavior In case of HyperFlash boot, Serial Flash boot and Octal SPI Flash boot, determines the flash reset behavior before the Boot ROM accesses. Flash Software Reset is available in Serial Flash boot and Octal SPI Flash boot. Show "SOFTMD[2:1]" from e-FUSE module. Any CPU and USER can read. No one can write. 2'b00: No reset. 2'b01: Flash Software Reset is applied. (command "0xF0") 2'b10: Flash Software Reset is applied. (command "0x66->0x99") 2'b11: Reserved
0	SOFTMD[0]	1'b 0	R	Reserved

* : depend on e-FUSE setting

9.7.2 Boot Status Register (MFISBTSTSR: H'E6260604)



Bit	Bit Name	Initial Value	R/W	Description
31 to 16	-	4'b0	R/W(sec) R(non sec)	Reserved
15 to 8	FLASHCLK	8'b0	R/W(sec) R(non sec)	Refer to Flash Clock Parameters in section 31 of R-Car V4H HWUM.
7 to 6	-	2'b0	R/W(sec) R(non sec)	Reserved
5	BTSCIF	1'b0	R/W(sec) R(non sec)	Boot From SCIF bit 0:Boot From Flash or eMMC 1:Boot From SCIF Download Mode
4	BTSELR	1'b0	R/W(sec) R(non sec)	IPL 1st/2nd-Boot Select bit. 0: IPL 1st-Side is loaded. 1: IPL 2nd-Side is loaded. Note : Flash or eMMC Boot Partition Information
3 to 1	BTSR	3'b0	R/W(sec) R(non sec)	BT Status Register bit3: EXT_DEBUGGER Secure Debug Integrity Check result bit. This bit is one when LCM state is CM, DM and FA. This bit is valid when BOOT_OK is 1 in case of LCM is SE (including temporary SD state). 0: Secure Debug integrity check is failed, or not executed. 1: Secure Debug integrity check is succeeded. bit2: SECURE_BOOT Integrity Check execution bit. 0: Integrity Check is not executed. 1: Integrity Check is executed. bit1: BOOT_OK Boot status bit. This bit is not valid when Normal boot in case of Cortex-R52 boot or SCIF DL mode. 0: Boot failed. 1: Boot succeed (including integrity check success when SECURE_BOOT is 1).
0	-	1'b0	R/W(sec) R(non sec)	Reserved

* Note : In case of non-secure access, don't update this register by write transaction.

In case of force SCIF Download Mode, The value of bit4 invalid.

9.7.3 Boot configuration register1 (BCFG1: H'E61BF170)

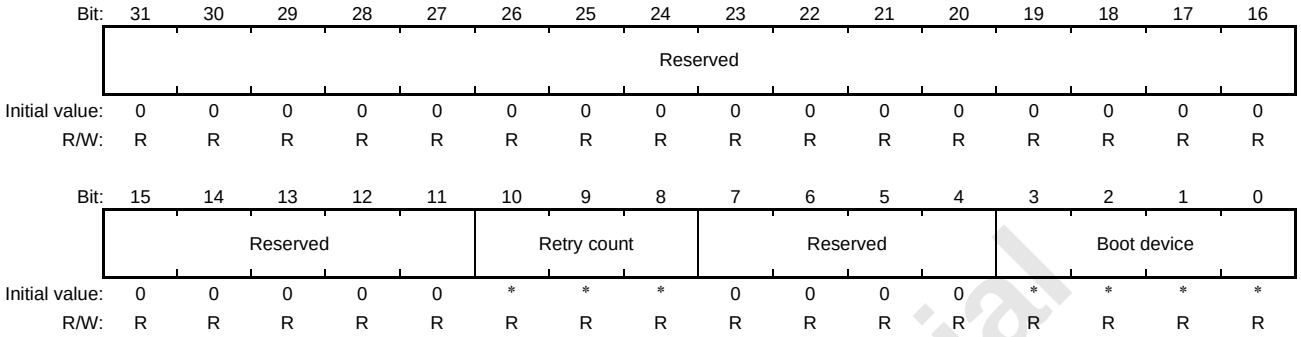
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Initial value:	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				Secure debug		Recovery		Reserved					Revocation		
Initial value:	0	0	0	0	*	*	*	*	0	0	0	0	0	*	*	*
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
31 to 30	-	*	R	Reserved
29 to 12	-	20'b 0	R	Reserved
11 to 10	Secure debug	*	R	Secure debug limit setting. 2'b00: Enable secure debug. (default) others: Debugger connection prohibited
9 to 8	Recovery	*	R	Recovery function setting. 2'b00: Jump SCIF download mode. (default) others: Jump No BOOT.
7 to 3	-	8'b 0	R	Reserved
2 to 0	Revocation	*	R	The Boot ROM program selects an available key from the 4 root keys. Key revocation code 3'b000 1st key available (no revoked) 3'b001 2nd key available (1st key revoked) 3'b011 3rd key available (2nd key revoked) 3'b111 4th key available (3rd key revoked)

* : depend on OTP setting

9.7.4 Boot configuration register2 (BCFG2: H'E61100BC)



Bit	Bit Name	Initial Value	R/W	Description
31 to 11	-	21'b 0	R	Reserved
10 to 8	Retry count	*	R	Number of retries from 0 to 7 (B'000-B'111)
7 to 4	-	4'b0	R	Reserved
3 to 0	Boot device	*	R	4'b0010 HyperFlash 160MHz 4'b0011 HyperFlash 80MHz 4'b0100 Serial Flash 40MHz 4'b0110 Serial Flash 40/80/133MHz 4'b0111 Octal SPI 80/160MHz 4'b1101 eMMC 4'b0000 initial value Others Not support

* : depend on OTP setting

9.8 4 root keys support

This chapter describes 4 root keys.

There are 4 root keys pair, revoke the root key pair using revocation bits of OTP, it is possible to change the root key pair up to 3 times.

In 4 root keys changed the structure of the certificate, increasing the number of root public keys and RSA signatures to 4.

The Structure of secure boot key certificate and secure boot content certificate are shown Figure 9-25.

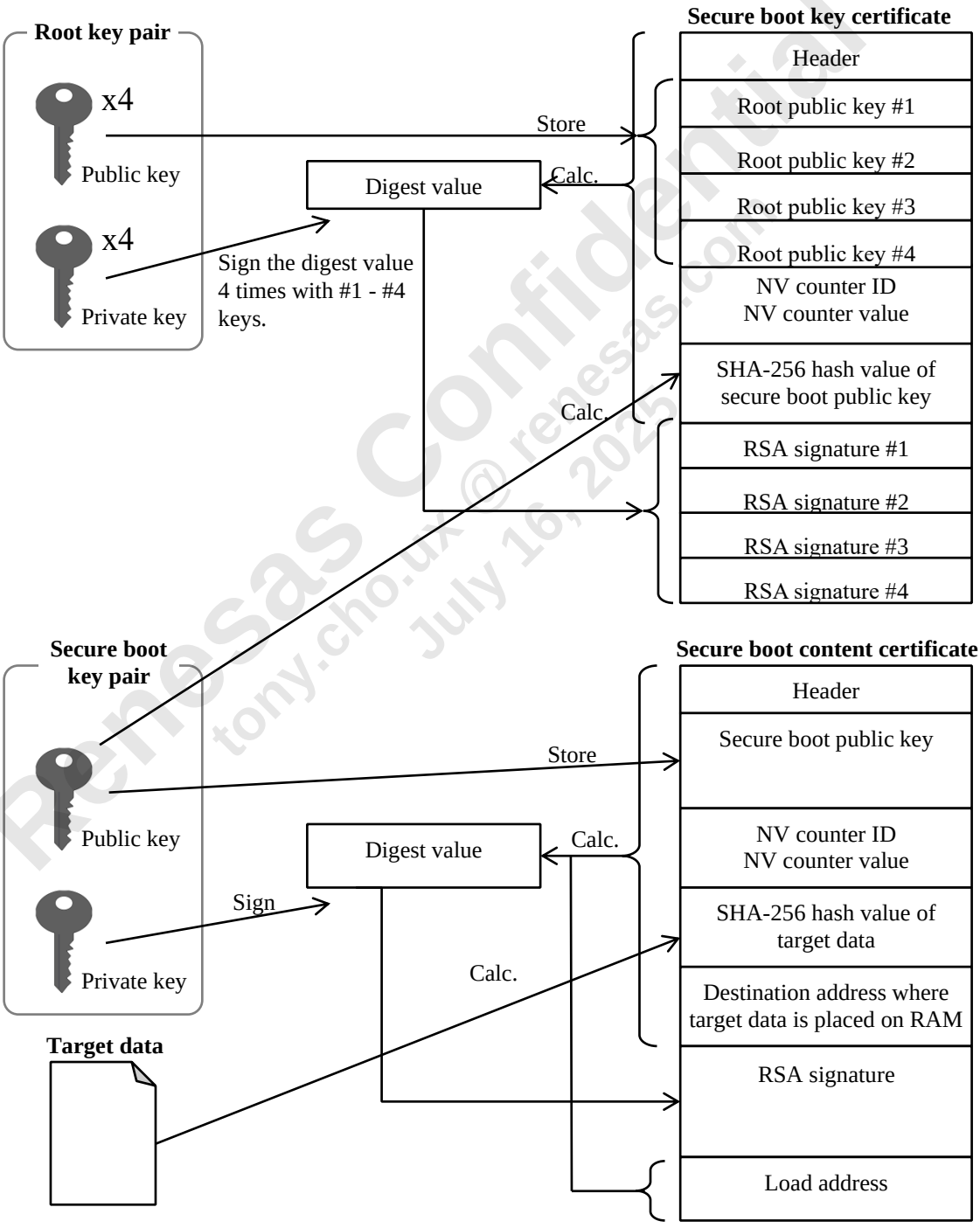


Figure 9-25 Structure of secure boot key certificate and content certificate for 4 root keys

The HBK has been changed to calculated from hash values of 4 root public keys.

The revocation bits of OTP determine the signature used to verify the digest value.

The Boot ROM program selects an available key from the 4 root keys by revocation bits of OTP.

How to read the revocation bits of OTP, refer to Revocation bits of 9.7.3.

Figure 9-26 shows verification of secure boot key certificate for 4 root keys.

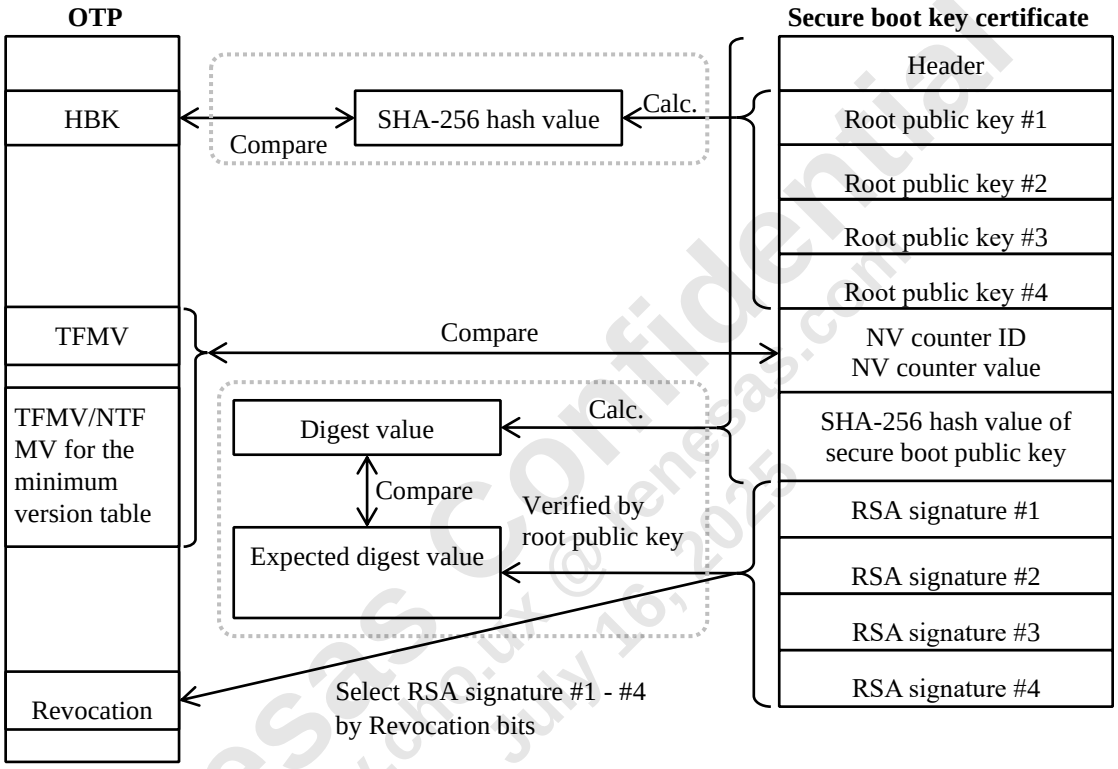


Figure 9-26 Verifying secure boot key certificate for 4 root keys

Updating the revocation bit of OTP, can change the root key used for verification of Secure boot key certificate.

Write position of the revocation bit of OTP, refer to section U10.2 of R-Car V4H HW UM.

The OTP revocation bit addresses and bit assignments are shown in Table 9-28.

Table 9-28 OTP revocation bits address and bit assignments

OTP address	bit name	bit [31:3]	bit [2:0]	description
H'090	Key revocation code [2:0]	Reserved	3'b000	1st key available (no revoked)
		Reserved	3'b001	2nd key available (1st key revoked)
		Reserved	3'b011	3rd key available (2nd key revoked)
		Reserved	3'b111	4th key available (3rd key revoked)
H'091	Key revocation code (copy) [2:0]	Reserved	(*)	(*)

* : Write the same value as Key revocation code [2:0].

Initial state is 1st key available, Key revocation code [2:0] value is 3b'000.

To revoke the 1st key, write 1 to bit0. The 1st key is revoked and 2nd key is available, Key revocation code [2:0] value changes from 3b'000 to 3b'001.

To revoke the 2nd key, write 1 to bit1. The 2nd key is revoked and 3rd key is available, Key revocation code [2:0] value changes from 3b'001 to 3b'011.

To revoke the 3rd key, write 1 to bit2. The 3rd key is revoked and 4th key is available, Key revocation code [2:0] value changes from 3b'011 to 3b'111.

The 4th root keypair cannot be revoked. Write the revocation bits in order from bit0 to bit2.

How to write the OTP, refer to Section U10.2 of R-Car V4H HW UM.

10. Usage Notes

10.1 Pull-up/down settings

Boot ROM program sets pull-up/down to QSPI pins.

Table 10-1 shows pull-up/down settings.

Table 10-1 Pull-up/down settings

MD[4:1]	Descriptions	QSPI0_IO2/QSPI0_IO3	QSPI1_SPCLK	QSPI1_MOSI_IO0/ QSPI1_MISO_IO1/ QSPI1_IO2/QSPI1_IO3
B'0100	Serial Flash 40MHz	Pull-up	Pull-down	Pull-up
B'0110	Serial Flash 40MHz to 80MHz (CLKSELR=0)	Pull-up	Pull-down	Pull-up
	Serial Flash 40MHz to Quad Output/IO (CLKSELR=2~5)	Pull-up(single) to No Pull-up/down (Quad)	Pull-down	Pull-up
	Serial Flash 40MHz (CLKSELR=others, except CLKSELR=1 (reserved))	Pull-up	Pull-down	Pull-up
B'0111	Octal SPI Flash 80/160MHz (CLKSELR=2~5)	Pull-up(single) to No Pull-up/down (octal)	Pull-down(single) to No Pull-up/down (octal)	Pull-up(single) to No Pull-up/down (octal)

R-Car V4H series Security User's Manual

Publication Date: Rev.0.50 Dec. 29, 2021
Rev.0.80 Apr. 28, 2023
Rev.1.00 Mar. 08, 2024
Rev 1.10 Apr. 11, 2025

Published by: Renesas Electronics Corporation

R-Car V4H series Security User's Manual

Renesas Electronics Corporation