User's Manual

CONFIDENTIAL

Initial Program Loader for ICUMXA

User's Manual: Software

SoC for Car Information Terminal Applications
R-Car Series, 3rd Generation R-Car V3H

Renesas Electronics
www.renesas.com

Rev.2.0.3 Aug. 2021

Notice

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

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

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Contact information

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Trademark

How to Use This Manual

- **[Readers]**
  This manual is intended for engineers who develop products which use the R-Car V3H processor.

- **[Purpose]**
  This manual is intended to give users an understanding of the functions of the R-Car V3H processor device driver and to serve as a reference for developing hardware and software for systems that use this driver.

- **[How to Read This Manual]**
  It is assumed that the readers of this manual have general knowledge in the fields of electrical
  — engineering, logic circuits, microcontrollers.
  → Read this manual in the order of the CONTENTS.
  — To understand the functions of a multimedia processor for R-Car V3H
  → See the R-Car V3H User's Manual.
  — To know the electrical specifications of the multimedia processor for R-Car V3H
  → See the R-Car V3H Data Sheet.

- **[Conventions]**
  The following symbols are used in this manual.
  Data significance: Higher digits on the left and lower digits on the right
  Note: Footnote for item marked with Note in the text
  Caution: Information requiring particular attention
  Remark: Supplementary information
  Numeric representation: Binary ... ××××, 0b××××, or ××××B
  Decimal ... ××××
  Hexadecimal ... 0x×××× or ××××H
  Data type: Double word … 64 bits
  Word … 32 bits
  Half word ... 16 bits
  Byte ... 8 bits

Table of Contents

1. Overview

1.1 Overview

This document explains about R-Car Series, 3rd Generation Initial Program Loader for ICUMXA (hereinafter called "Loader"). Loader provides the binary image loading function from QSPI FLASH of external flash device. A loading images are RTOS, CA53 program and Secure Firmware (hereinafter called "Secure FW"). In addition to that Loader will load a certificate for integrity check.
After that, Loader will boot up the other CPU (Cortex-R7 and Cortex-A53) and execute Secure FW.
The Loader is not support for Safety Requirement Specification. If adaptation of these specifications is required, please implement by user.

1.2 Function

This chapter explains the details of the functions provided by Loader.
Figure 1.1 shows the software components related to Loader and the scope of Loader. The scope of Loader is indicated by the blue frame.
Loader is loaded and executed by the BootROM program. Loader loads binary images (RTOS, CA53 program, and Secure FW) from QSPI FLASH and executes an integrity check loaded images using a certificate. When Loader succeeds in an integrity check of the binary image, it boots up the binary images for each CPU.

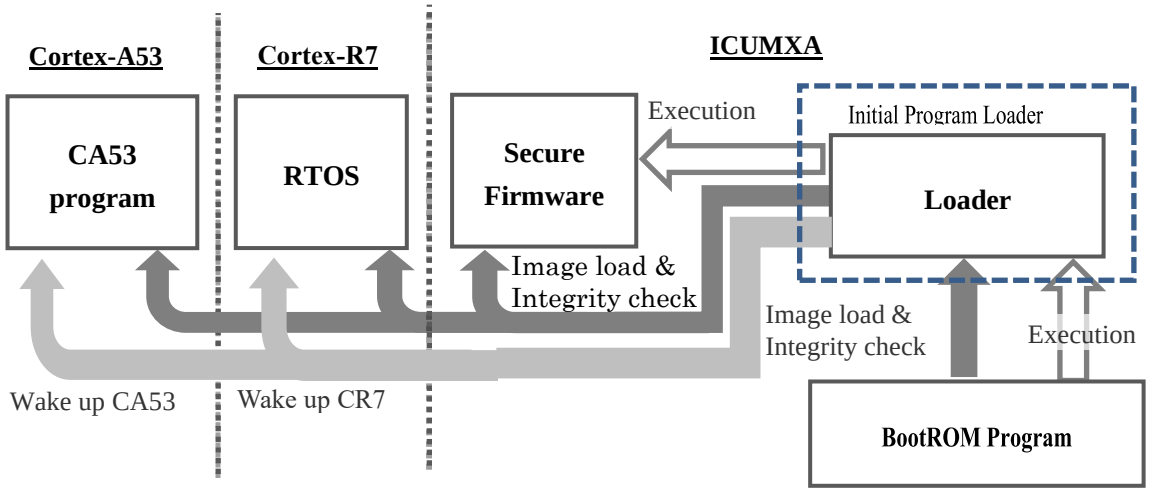


Figure 1.1 Scope of the Internal Program Loader

List of functions provided by the Loader is shown in Table 1-1.

Table 1-1 Function list of the Loader

Functions	Explanation
Hardware initialization	Initialize the configuration and the hardware used by the system.
Display the booting message	Output the message (software version, etc.) to SCIF.
Decide the boot mode.	The boot mode is decided from LCS state and MD pins, and Loader execute processing according to the boot mode.
Loading the image	Load binary images (RTOS, CA53 program, Secure FW), certificate images and a header image from QSPI FLASH.
Integrity check for the image using a certificate	If the boot mode of the Loader is the Secure mode, the binary images execute an integrity check by the certificate.
Release HW resources used by Loader	Release HW resource used by Loader. The Loader reinitializes HW to the initial value of HW.
Starting RTOS	Loader boots up Cortex-R7 and executes a RTOS binary image on Cortex-R7.
Starting CA53 program	Loader boots up Cortex-A53 and executes a CA53 program binary image on Cortex-CA53. If CA53 program#1 binary image is Secure Monitor, Loader writes a boot parameter of Secure Monitor in the memory area of Secure Monitor before booting Cortex-A53.
Starting Secure FW	Execute a Secure FW binary image after Loader processing was finish.

1.2.1 Hardware initialization

Loader initializes a following hardware. Detail on these functions is shown in Chapter 3.3.

- CPU initialization
- WDT initialization
- SCIF initialization
- Start supplying clock signal to module
- PFC and GPIO initialization
- DMA initialization
- Access protection setting
- SDRAM/PHY initialization
- QoS initialization

1.2.2 Display the booting message

This function displays various information during Loader execution.
The information is output to SCIF. If a user wants to get information on PC, connect a USB cable to DEBUG SERIAL-0 on a Condor / Condor-I board.
Figure 1.2 shows the information that Loader outputs to the terminal. However, the displayed values are an example.

```
N:ICUMXA Loader Rev.2.0.2
N:Built : 07:31:07, Jul  2 2021
N:PRR is R-Car V3H Ver2.0
N:LCM state is CM
N:Access Protection Enable.
N:Normal boot(ICUMX)
N:======== key cert info ========
destination address:0xfdefe400
physical destination address:0xeb2fe400
source address:0xeb200f00
size:0x00000400
N:======== content cert info ========
destination address:0xfdefb800
physical destination address:0xeb2fb800
source address:0x08180000
size:0x00000400
N:======== content cert ========
address:0xfdefbc00 size:0x00001000
N:======== RTOS image load info ========
load address    = 0xeb200000
image size      = 0x000b4000
source address  = 0x081c0000
N:DDR3200(rev.0.41)
N:[COLD_BOOT]
N:QoS setting(rev.0.10)
N:DRAM refresh interval 1.89 usec
N:Periodic Write DQ Training
N:======== CA53 Program image load info ========
load address    = 0x46400000
image size      = 0x00080000
source address  = 0x082c0000
N:======== CA53 Program image load info ========
load address    = 0x50000000
image size      = 0x00100000
source address  = 0x08840000
N:======== SecureFW image load info ========
load address    = 0xeb2b4000
image size      = 0x00044800
source address  = 0x080c0000

Dummy FW Program
Dummy FW Program boot end

Dummy RTOS Program
Dummy RTOS Program boot end
```

Figure 1.2 Booting message of Condor / Condor-I board

Table 1-2 shows a list of displayed information and meaning.

Table 1-2 Meaning of booting message

Item	Explanation
ICUMXA Loader Rev.0.1.0	Loader software version.
Built : 08:36:04, May 23 2018	The time and date the Loader was built.
PRR is R-Car V3H Ver1.0 LCM state is CM	LSI version and LCS state. The version is acquired by reading the PRR register of the LSI. LCS state is acquired by executing the API of BootROM.
Access Protection Enable.	Enabling and Disabling an access protection function.
Normal boot(ICUMX)	Boot mode. When boot mode is Normal mode, Normal boot (ICUMX). When boot mode is Secure Mode, Secure boot (ICUMX).
======== key cert info ======== … ======== content cert info ======== … ======== content cert ========	Certificate information. The following information when Loader transfers the certificate is displayed. ・Destination address (including physical address). ・Source address. ・Size of transfer.
======== RTOS image load info ========	RTOS binary image information. ・Destination address (load address) ・Size of binary image. ・Source address.
N:[BoardType=12]N:BL2: DDR3200(rev.0.28rc05)N:[COLD_BOOT]N:..0	Information of SDRAM initialize processing. ・Clock rate. ・Source code version.
QoS is default setting(rev.0.02)	Setting of QoS (Default setting / No setting) and revision.
======== CA53 Program image load info ========	CA53 Program binary image information. The information displayed is the same as “RTOS binary image information.”
======== SecureFW image load info ========	Secure FW binary image information. The information displayed is the same as “RTOS binary image information.”

1.2.3 Decide the boot mode.

Loader uses the information of LCS state and MD pins to decide a boot mode.
Boot mode has two states. One is Secure Mode and the other is Normal Mode. In case of Secure Mode, Loader executes an integrity check by the certificate to a loaded binary image. In Other case, Loader don't execute an integrity check.

Figure 1.3 shows a processing flow for deciding the boot mode.

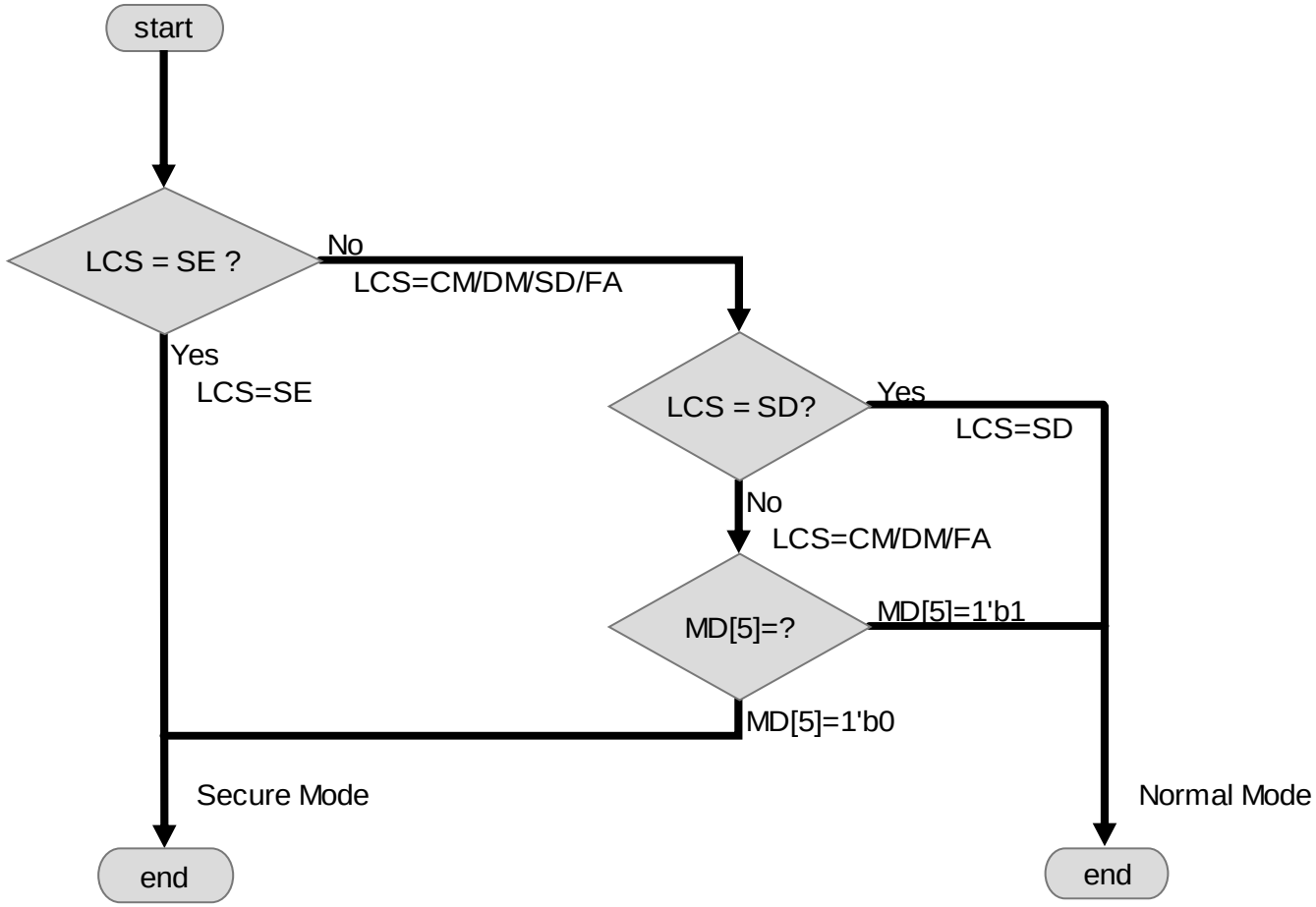


Figure 1.3 Processing flow for deciding the boot mode

1.2.4 Loading the image

Loader loads a binary image to RAM (RT-SRAM or DRAM) from QSPI FLASH.
Binary images loaded by Loader are shown below.

- Key certificate*1
- Cert header
- RTOS
- CA53 Program#[X]*2
- Secure FW
- Certificates of each binary images.*3

*1) Only this binary image is transferred from RT-SRAM. Because, key certificate has been loaded in RT-SRAM by BootROM.
*2) [X] is a number from 1 to 8.
*3) The number of certificates is same as the number of binary images.

The data flow of the binary image loaded by Loader is shown in Figure 1.4.

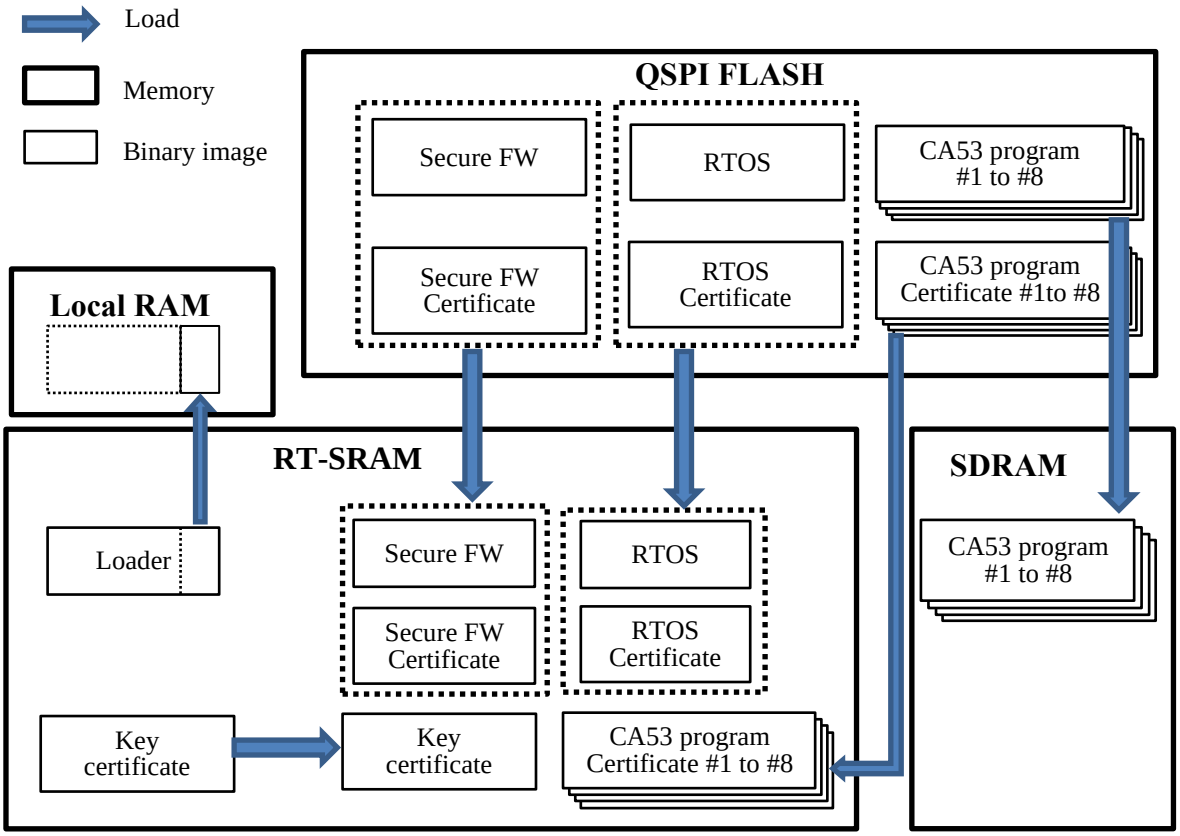


Figure 1.4 Data flow of a binary image.

Transfer to RAM is executed by RT-DMAC. A data transfer size is always aligned in units of 256 bytes. Therefore, if the image size does not match the alignment of 256 bytes, a transfer size is adjusted to 256 bytes alignment.

A source and destination address of binary images can be changed by the user. An information of the source address is stored in Cert Header. An information of the destination address is stored in the certificate corresponding to each binary image.
Loader reads and analyzes Cert Header and certificates to decide the source and destination addresses of the binary image.

Figure 1.5 shows the structure of the Cert Header.
The Cert Header include data, which is the FLASH address of RTOS, CA53 Program#[X] and Secure FW. The FLASH address is set by the offset address from the head address of the flash device.
The number of CA53 Programs is included in the head area of the Cert Header.
Each field of the Cert Header is 32-bit little endian.

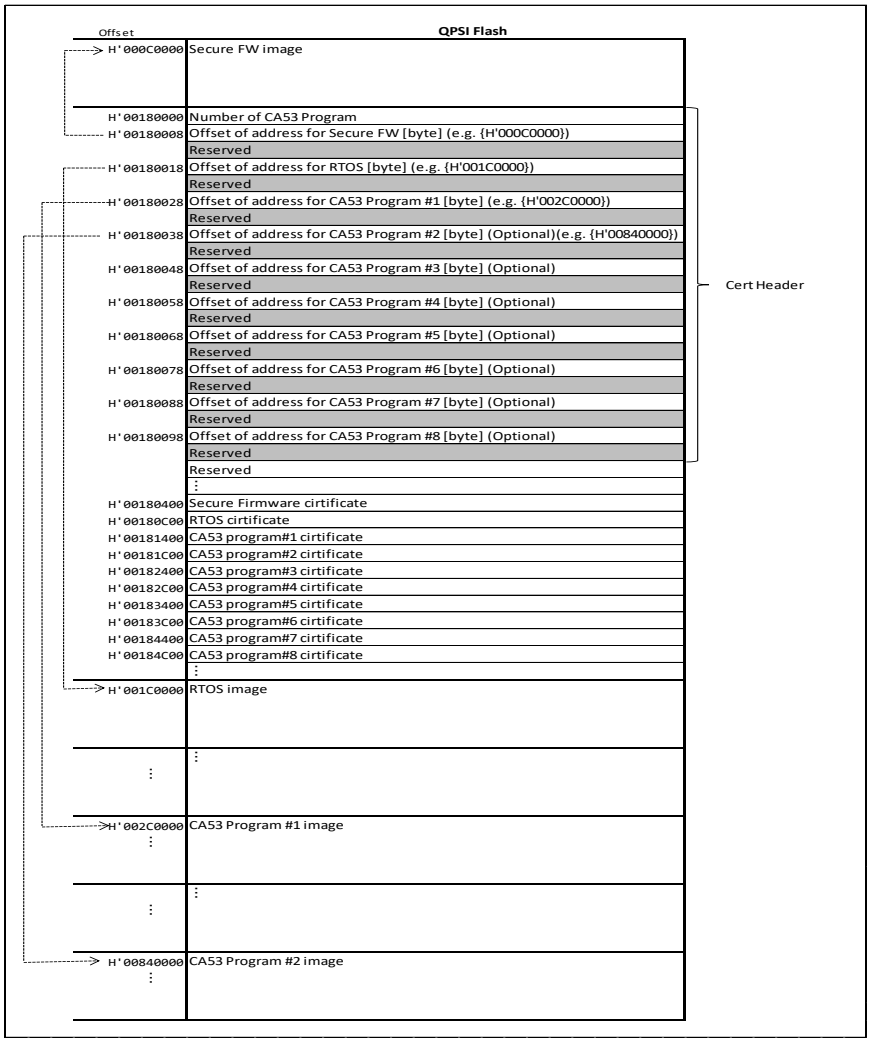


Figure 1.5 Structure of the Cert Header

1.2.5 Integrity check for the image using a certificate

Loader executes an integrity check by the certificate to a loaded binary image. If an integrity check is successful, Loader decides that the loaded binary image can be executed. If the check is failed, Loader outputs an error message and stop executing.

Figure 1.6 shows flow of integrity check in Secure Mode.
Loader executes Secure Boot API at integrity check. The API is provided by BootROM, Loader can verify the integrity of binary image by executing the API with address of certificate binary image as an argument.

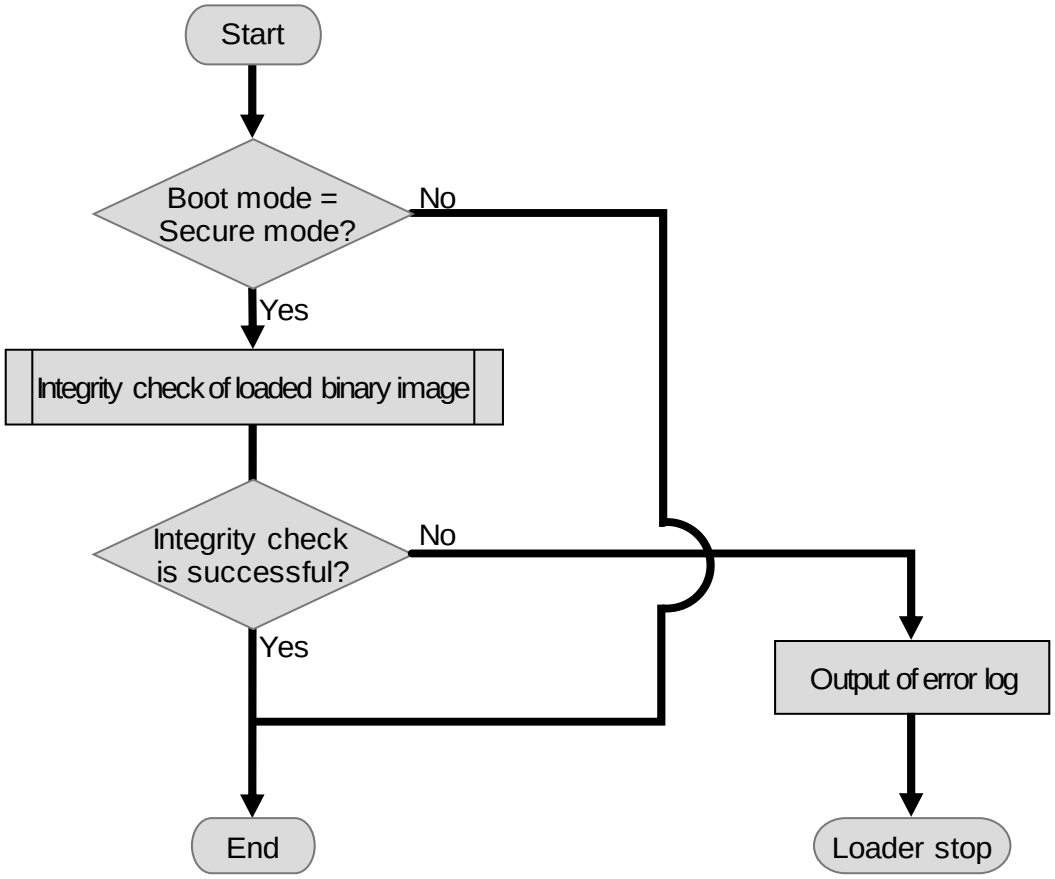


Figure 1.6 Flow of integrity check in Secure mode

1.2.6 Releasing used resources

Loader uses several HW resources to be executing Loader. When Loader processing finished, HW resources are released by Loader. The used HW resources are reinitialized to an initial value of HW.
However, settings of HW for running software to be executed after Loader are not reinitialized.

1.2.7 Starting RTOS

After transferring the binary image of RTOS to RAM, Loader boots up the Cortex-R7 and executes RTOS on Cortex-R7.

1.2.8 Starting CA53 program

After transferring the binary image of CA53 Program#[X] to RAM, Loader boots up the Cortex-A53 and executes CA53 Program#1 on Cortex-A53. At the timing of booting Cortex-A53, all binary images of CA53 Program#[X] are transferred in RAM.
Cortex-A53 is a cluster device that has four cores. Loader boots up core 0 in the Cortex-A53 cluster.

If CA53 program#1 binary image is Secure Monitor, Loader writes a boot parameter of Secure Monitor in a memory area of Secure Monitor before booting Cortex-A53. A location of the boot parameter of the Secure Monitor is an address offset by H'22200 from the Secure Monitor entry point.

1.2.9 Starting Secure FW

After transferring the binary image of Secure FW and all Loader processing is finished, Loader executes Secure FW. Loader jumps the program counter to Secure FW entry point. The entry point is the top address of the binary image of Secure FW.

1.3 References

1.3.1 Related Document

The following table shows the document related to this function.

Table 1-3 Related Document

Number	Issue	Title
1	Renesas Electronics	R-CarV3H System Evaluation Board “Condor” / “Condor-I” Condor : RTP0RC77980SEB0010SA01 Condor-I : RTP0RC77980SEBS012SA01
2	Renesas Electronics	R-Car Series, 3rd Generation User's Manual: Hardware
3	Renesas Electronics	R-Car Series, 3rd Generation Safety Application Note
4	Renesas Electronics	R-Car Series, 3rd Generation Hardware Description For Functional Safety

1.4 Restrictions

There is no permanent restriction.

2. Terminology

The following table shows the terminology to this function.

Table 2-1 Terminology

Terms	Explanation
API	Application Programming Interface
AXI	Advanced eXtensible Interface
CA53	Cortex-A53
CPG	Clock Pulse Generator
CR7	Cortex-R7
DBSC	DRAM Bus State Controller
DDR4	Double-Data-Rate4
DMAC	Direct Memory Access Controller
DRAM	Dynamic RAM
ECC	Error Correction Code
LCM	Life Cycle management
LCS	LCM State
LPDDR4-SDRAM	Low Power DDR4 SDRAM
LifeC	Life cycle Count
RT-DMAC	Real Time Direct Memory Access Controller
BootROM	BootROM program
PFC	Pin Function Controller
PRR	Product Register
QoS	Quality of Service
QSPI	Quad Serial Peripheral Interface
RAM	Random Access Memory
ROM	Read Only Memory
RPC	Reduced Pin Count
SCIF	Serial Communication Interface with FIFO
SDRAM	Synchronous DRAM
SPI	Serial Peripheral Interface

3. Operating Environment

3.1 Hardware Environment

The following table lists the hardware needed to use this function.

Table 3-1 Hardware environment (R-Car Series, 3rd Generation)

Name	Explanation
Evaluation Board	R-Car V3H System Evaluation Board (Condor / Condor-I) Renesas Electronics
Host PC (Windows and Ubuntu)	Windows 10 is recommended for Windows host PC. This is used to compile Loader and execute the terminal software. Ubuntu 16.04 is recommended for Ubuntu host PC. However, it is used only for compiling sample software (Dummy RTOS and Dummy CA53 Program).

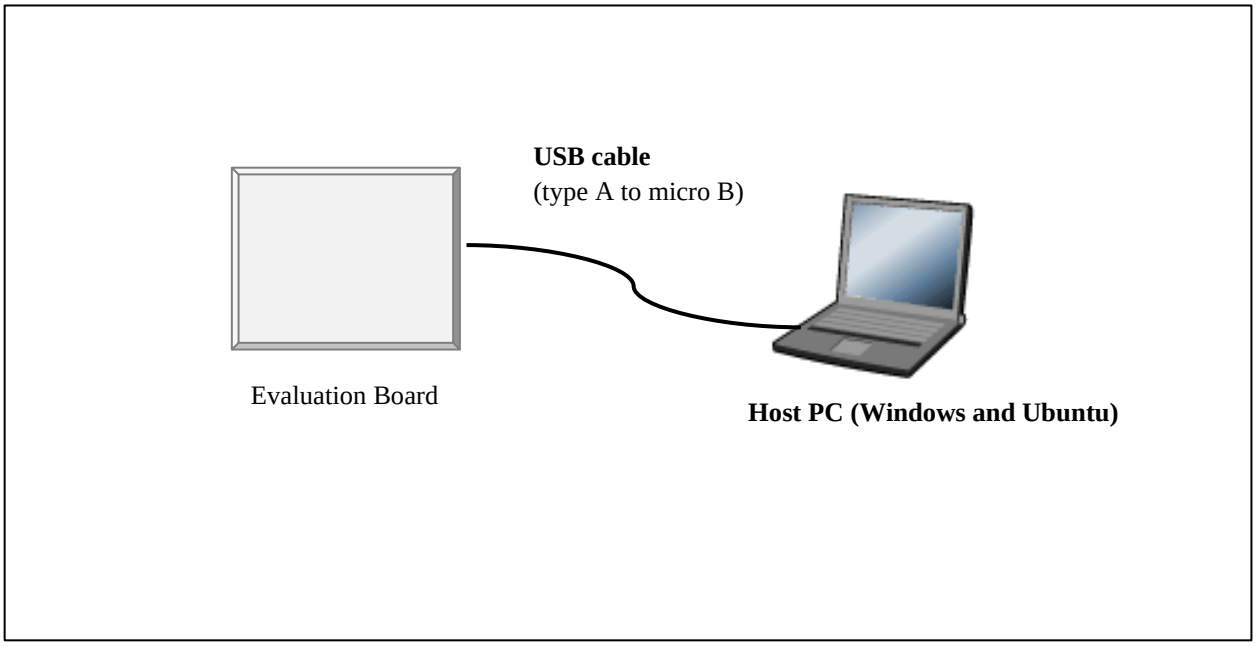


Figure 3.1 Recommended Environment

3.2 Module configuration

Shows module configuration of Condor / Condor-I. And Table 3-2 shows explanation of Hardware resource using in.

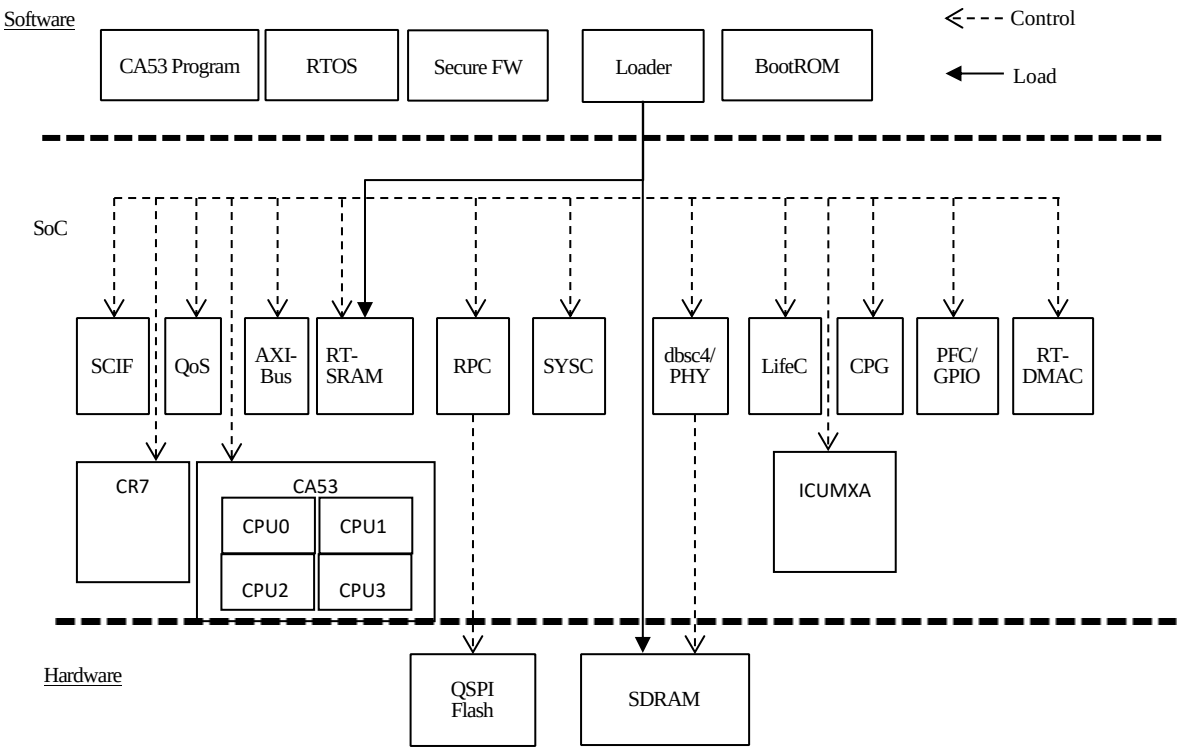


Figure 3.2 Module configuration

Table 3-2 Hardware resource

Component	Uses
RPC/QSPI	Reading the QSPI Flash.
PFC/GPIO	Setting the External pin.
LifeC	Setting the Secure protection of IP on SoC.
CPG	Setting the access protection of reset/stop module in CPG.
RT-SRAM	Executing the Loader.
DBSC4/PHY	Initialization for access to the LPDDR4-SDRAM.
AXI-Bus	Setting the access protection of System RAM and SDRAM.
RT-DMAC	Accelerating software loading with the DMA.
SCIF	Displaying starting message and log message.
QSPI Flash	Loading a source data from this device (External device).
LPDDR4-SDRAM	Loading a destination data from this device (External device).
CR7	Starting the CPU when RTOS is executed.
CA53	Starting the CPU when CA53 program is executed.

3.3 Processing Flow Diagram

Figure 3.3 and Figure 3.4 show processing flow of Loader.
Figure 3.5 show sequence of Loader.
Detailed contents of the flow are shown in the next section.

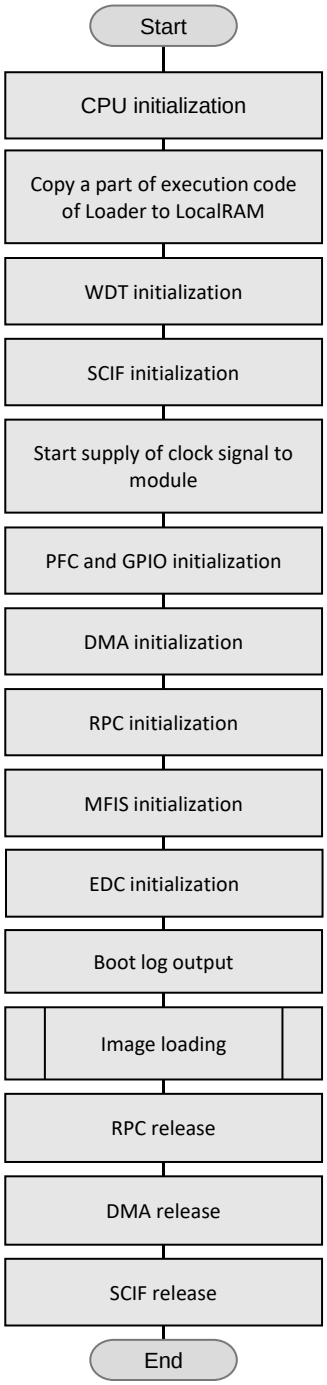


Figure 3.3 Flow of Loader processing

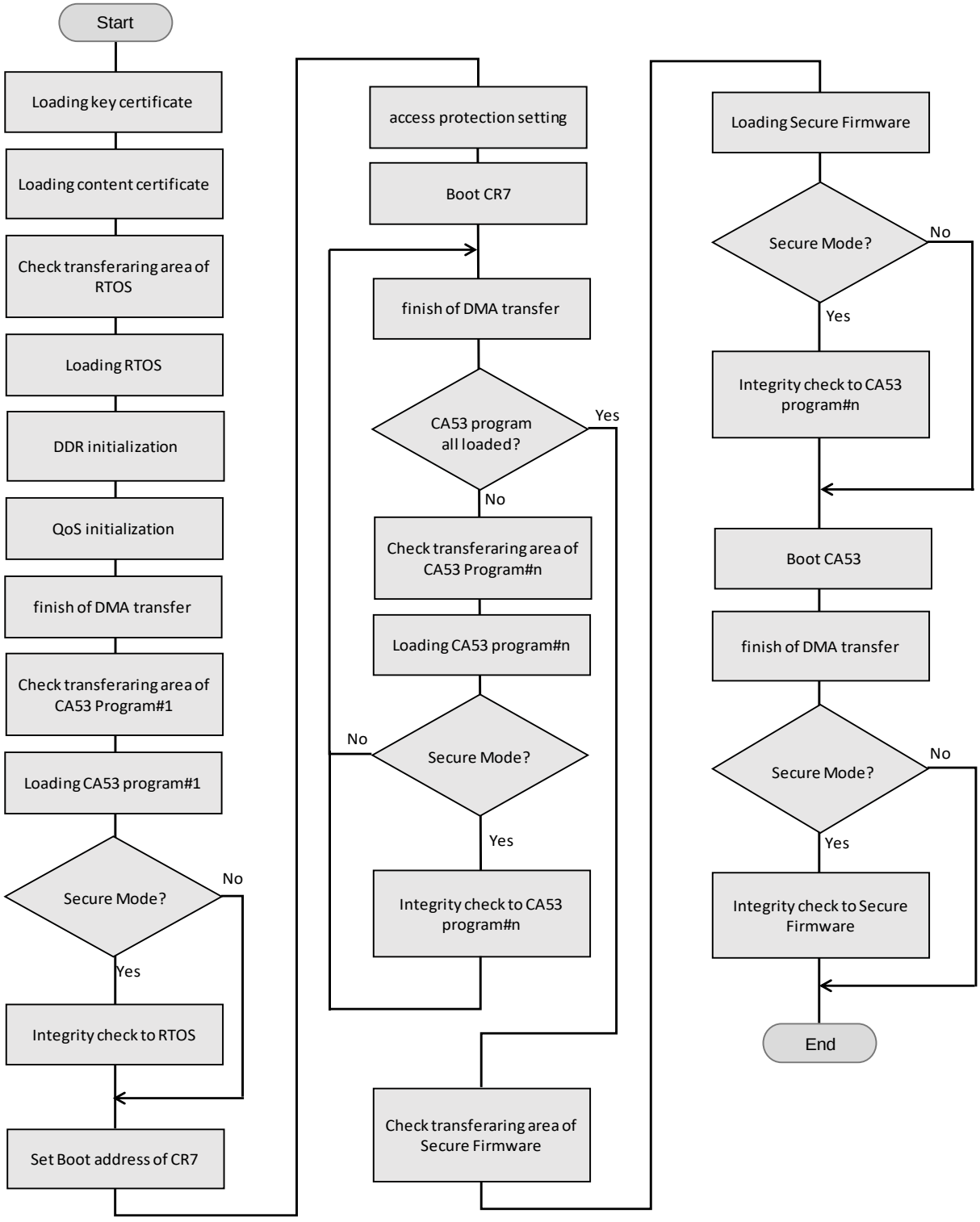


Figure 3.4 Flow of image loading

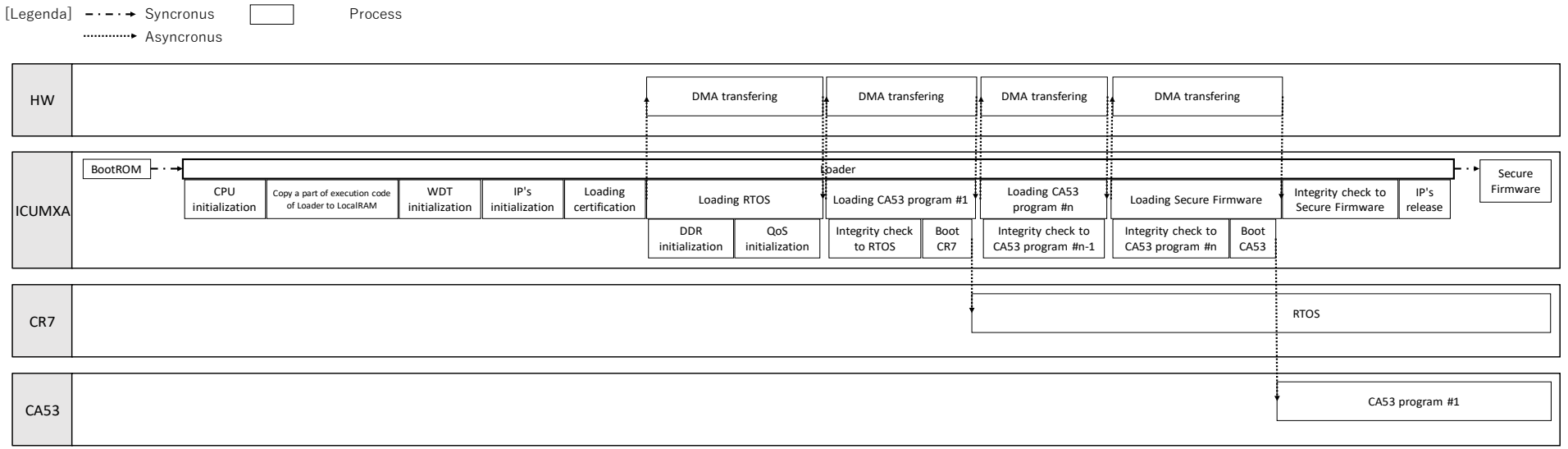


Figure 3.5 Sequence of Loader

3.3.1 CPU initialization

Table 3-3 shows CPU initializing process and its explanation.

Table 3-3 CPU initializing process

Process	Explanation
Initialization of the general-purpose register	R1-R29 in the general-purpose register is cleared to zero.
Clear stack area of BootROM	Clear zero the stack area used by the Boot ROM.
Initialization of the RAM	.bss section is cleared to zero.
Stack configuration	Initializing of Stack pointer (SP).

3.3.2 Copy a part of execution code of Loader to Local RAM

The Loader copies data from RT-SRAM to Local RAM. The data is a part of Loader executed on Local RAM.

3.3.3 WDT initialization

Loader sets Window Watchdog Timer included in the ICUMXA.
Window Watchdog Timer setting is shown in Table 3-4.

Table 3-4 List of Window Watchdog Timer setting

Item	Setting
error mode	Reset mode
Window-open period	Window-open period is 100%
Enable/disable the 75% interrupt request	Enable
Overflow interval time	187msec (2^13 / 32.8KHz * 0.75)

3.3.4 SCIF initialization

Loader sets SCIF 0 to output logs to the outside.
The terminal settings to output of log to PC is shown in Table 3-5.

Table 3-5 List of the terminal setting for PC

Item	Setting
Baud rate	115200bps
Data size	8bit
Parity	None
Stop bit	1bit
Flow control	None

3.3.5 Start supplying clock signal to module

Loader starts supplying the clock signal to modules for subsequent software (RTOS, CA53 program, and Secure FW).
Other than registers in Table 3-6 are maintained initial value of HW.
Detail of the setting value refer to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 8A.1.3”.

Table 3-6 List of Module Stop setting

Register Name	Setting value	Initial value of HW
RMSTPCR1	0xFDDFFFFF	0xFFFFFFFF
RMSTPCR3	0xFF7FFF19	0xFFFFFFDF
RMSTPCR5	0x009FFFDE	0x009FFFFE
RMSTPCR8	0x00F3EFFC	0x00F3FFFC
RMSTPCR9	0x03F1A017	0x03F1E017
SMSTPCR1	0xF5FFFFFF	0xFFFFFFFF
SMSTPCR7	0xFFFFFF3F	0xFFFFFFFF
SCMSTPCR9	0xFFFDFFFF	0xFFFFFFFF

3.3.6 PFC and GPIO initialization

Loader sets a peripheral device of PFC and GPIO to execute Condor / Condor-I board. The setting of the register shown in Table 3-7 and Table 3-8 is necessary to initialize for Condor / Condor-I board. Detail of the set value refers to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 6B.1.3” and “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 7.1.4”.

Table 3-7 List of the PFC setting

Register Name	Setting value
GPSR0	0x00000000
GPSR1	0x0E66FFFF
GPSR2	0x00000000
GPSR3	0x0001FFC0
GPSR4	0x01BFFFFF
GPSR5	0x00000FFF
IPSR0	0x00000000
IPSR1	0x00000000
IPSR2	0x00000000
IPSR3	0x00000000
IPSR4	0x00000000
IPSR5	0x44000000
IPSR6	0x44444444
IPSR7	0x04400004
IPSR8	0x00400000
IPSR9	0x00000000
IPSR10	0x00000000
IOCTRL0	0x00000000
IOCTRL1	0x00000000
IOCTRL2	0x00000000
IOCTRL3	0x00000000
IOCTRL4	0x00000000
IOCTRL5	0x00000000
IOCTRL6	0x00000000
IOCTRL7	0x00000000
IOCTRL8	0x00000000
IOCTRL9	0x00000000
IOCTRL10	0x00000000
IOCTRL11	0x00000000
IOCTRL12	0x00000000
IOCTRL13	0x00000000
IOCTRL14	0x00000000
IOCTRL15	0x00000000
IOCTRL16	0x00000000
IOCTRL17	0x00000000
IOCTRL18	0x00000000
IOCTRL19	0x00000000
IOCTRL30	0xFFFFFFFF
IOCTRL31	0xFFFFFFFF

IOCTRL32	0x0000001F
IOCTRL33	0x00000000
IOCTRL40	0x00000000
PUEN0	0x00000700
PUEN1	0x7E01C700
PUEN2	0x003F0000
PUEN3	0x07000000
PUEN4	0x0381E800
PUD0	0x80000000
PUD1	0x1B01C77C
PUD2	0x00000000
PUD3	0x0F800008
PUD4	0x03807C00
MOD_SEL0	0x00000200

Table 3-8 List of the GPIO setting

Register Name	Setting value
POSNEG0	0x00000000
POSNEG1	0x00000000
POSNEG2	0x00000000
POSNEG3	0x00000000
POSNEG4	0x00000000
POSNEG5	0x00000000
IOINTSEL0	0x00000000
IOINTSEL1	0x00000000
IOINTSEL2	0x00000000
IOINTSEL3	0x00000000
IOINTSEL4	0x00000000
IOINTSEL5	0x00000000
OUTDT0	0x00000000
OUTDT1	0x00010000
OUTDT2	0x00000000
OUTDT3	0x00000000
OUTDT4	0x00400000
OUTDT5	0x00007000
INOUTSEL0	0x00000000
INOUTSEL1	0x00010000
INOUTSEL2	0x00000000
INOUTSEL3	0x00000000
INOUTSEL4	0x00400000
INOUTSEL5	0x00007000

3.3.7 DMA initialization

Loader performs initial setting for transferring memory from memory by DMA. Loader performs DMA transfer using RT-DMAC.

3.3.8 RPC initialization

Loader sets RPC so that can access the entire area of Flash memory.
NOTE) In the HW initial setting, entire areas of the flash memory can't be accessed.

3.3.9 MFIS initialization

Loader sets the following.

- The enable / disable of error check.
- The output destination of the detected error.
- The hold timer control for error request.

Table 3-9 shows MFIS register setting values.

Table 3-9 List of the MFIS setting

Register Name	Setting value
MFIERRCTLR0	0xBFFBFB70
MFIERRCTLR1	0x3607FFE8
MFIERRCTLR2	0xFFFFFFFF
MFIERRCTLR3	0x9FFFFFFF
MFIERRCTLR4	0x13B9E8FF
MFIERRCTLR5	0x00000000
MFIERRCTLR6	0x00000000
MFIERRCTLR7	0xFFFF6D7F
MFIERRCTLR8	0x01FF0FFF
MFIERRCTLR9	0xFF00175F
MFIERRCTLR10	0x00000000
MFIERRCTLR11	0x00000000
MFIERRCTLR12	0x2550780F
MFIERRCTLR13	0x00003F7F
MFIERRTGTR0	0x8F080000
MFIERRTGTR1	0x00000400
MFIERRTGTR2	0x00000000
MFIERRTGTR3	0x00000000
MFIERRTGTR4	0x02000000
MFIERRTGTR5	0x00000000
MFIERRTGTR6	0x00000000
MFIERRTGTR7	0x02AA002A
MFIERRTGTR8	0x007F00A0
MFIERRTGTR9	0xFD000000
MFIERRTGTR10	0x00000000
MFIERRTGTR11	0x00000000
MFIERRTGTR12	0x00000000
MFIERRTGTR13	0x00000000

MFIEXTRQHLDCNTR	0x80000001

3.3.10 EDC initialization

Loader sets EDC registers according to chapter 4.3.4, 4.3.16 and 4.3.23 in Related Document [3].
Table 3-10 shows EDC register setting values.

Table 3-10 List of the EDC setting

Register Name	Setting value
EDCINTEN0	0xFC04EFFF
EDCINTEN1	0xFDFBE000
EDCINTEN2	0xDFFFFFFC
EDCINTEN3	0x7C57FFBF
EDCINTEN5	0xFFFFFFFF
EDCINTEN6	0xC0FE3BF1
EDCINTEN7	0x000007FF
EDC_CFG	0x00000001

3.3.11 Boot log output

Information on Loader is display. The following is an information to display in this processing.

- Loader version
- The time and date the Loader was built
- LSI version and LCS state
- Enabling and disabling an access protection function
- Boot mode

Refer to Chapter 1.2.2 about the example the information to display.

3.3.12 Image loading

The Loader execute loading of binary images and integrity check, boot up each CPU.
Refer to Figure 3.4 for the flow of this process. The next chapter explains each process in this flow.

3.3.12.1 Loading key certificate

Loader transfers the key certificate to RT-SRAM using RT-DMAC ch0. This processing does not end until the DMA transfer is completed.
The key certificate is transferred to the RT-SRAM by BootROM. Therefore, Loader transfers from RT-SRAM to RT-SRAM. Because, the area of the key certificate used by the BootROM is used as the image transfer destination of the RTOS binary image.

3.3.12.2 Loading content certificate

Loader transfers the content certificate to RT-SRAM from QSPI FLASH using RT-DMAC ch0. This processing does not end until the DMA transfer is completed.
The content certificates are as many as binary images. The number of content certificates is decided by Cert Header. One content certificate is transferred to RT-SRAM by one DMA transfer. And repeat DMA transfer for the number of content certificates.

3.3.12.3 Check transferable area of [binary image].

Loader checks the transferable area before transferring the binary image.
Loader executes the following confirmation using the transfer source address, transfer destination address, and image size values obtained from the content certificate.

- Transfer source address is within Flash RAM range
- Transfer destination address is within RT-SRAM or SDRAM
- The image to be transferred will not overlap with the binary image that has already been transferred.

3.3.12.4 Loading [binary image]

Loader transfers the binary image (RTOS, CA53 program #1 to #8, Secure FW) to RT-SRAM or SDRAM from QSPI FLASH using RT-DMAC ch0.
In parallel with the DMA transfer, Loader executes an integrity check of the loaded binary image and boot up the CPU. However, a binary image to be integrity check is not loaded when a binary image of RTOS is loading, Loader executes initialize DRAM and QoS.

3.3.12.5 SDRAM/PHY initialization

The Loader execute of initialization to use SDRAM.
This process is executed in parallel with the "Loading RTOS" processing.

3.3.12.6 QoS initialization

Loader sets the registers of DBSC and AXI-Bus to set QoS.
This process is executed in parallel with the "Loading RTOS" processing.

3.3.12.7 Finish of DMA transfer

Check the end of DMA transfer and clear the channel of DMA used.
Resetting the window watchdog timer count is repeated until the DMA transfer is completed.

3.3.12.8 Integrity check to [binary image]

Loader executes an integrity check of the loaded binary images (RTOS, CA53 ptogram#1 to #8, Secure FW).
If the check succeeds, proceed to the next processing. If the check failed, Loader outputs an error log and stops processing.

3.3.12.9 Set boot address of CR7

Loader sets the entry point of the loaded binary image to boot address.
If Loader changes the boot address of CR7, Loader needs the highest safety level.
Therefore, Loader temporarily changes to the highest safety level.

3.3.12.10 Access protection setting

Loader controls the LifeC, AXI-Bus, and RT-SRAM registers to enable the access protection.
The following section shows the access protection settings of each module sets by Loader.

(1) LifeC protection setting

Loader sets access protection of internal bus between Master and Slave by LifeC.
Table 3-11 shows LifeC register setting values. Detail of the set value refers to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 68.1.3”.

Table 3-11 List of the access protection setting for LifeC

Module	Register Name	Setting value
LifeC	SEC_SRC	0x0000001F
	SEC_SEL0	0xFFFFFFFF
	SEC_SEL1	0xFFFFFFFF
	SEC_SEL2	0xFFFFFFFF
	SEC_SEL3	0xFFFFFFFF
	SEC_SEL4	0xFFFFFFFF
	SEC_SEL5	0xFFFFFFFF
	SEC_SEL6	0xFFFFFFFF
	SEC_SEL7	0xFFFFFFFF
	SEC_SEL8	0xFFFFFFFF
	SEC_SEL9	0xFFFFFFFF
	SEC_SEL10	0xFFFFFFFF
	SEC_SEL11	0xFFFFFFFF
	SEC_SEL12	0xFFFFFFFF
	SEC_SEL13	0xFFFFFFFF
	SEC_SEL14	0xFFFFFFFF
	SEC_SEL15	0xFFFFFFFF
	SEC_GRP0CR0	0x00000000
	SEC_GRP1CR0	0x00000000
	SEC_GRP0CR1	0x00000000
	SEC_GRP1CR1	0x00000000
	SEC_GRP0CR2	0x00000000
	SEC_GRP1CR2	0x00000000
	SEC_GRP0CR3	0x00000000
	SEC_GRP1CR3	0x00000000
	SEC_GRP0COND0	0x00000000
	SEC_GRP1COND0	0x00000000
	SEC_GRP0COND1	0x00000000
	SEC_GRP1COND1	0x00000000
	SEC_GRP0COND2	0x00000000
	SEC_GRP1COND2	0x00000000
	SEC_GRP0COND3	0x00000000
	SEC_GRP1COND3	0x00000000
	SEC_GRP0COND4	0x00000000
	SEC_GRP1COND4	0x00000000
	SEC_GRP0COND5	0x00000000
	SEC_GRP1COND5	0x00000000
	SEC_GRP0COND6	0x00000000
	SEC_GRP1COND6	0x00000000
	SEC_GRP0COND7	0x00000000
	SEC_GRP1COND7	0x00000000
	SEC_GRP0COND8	0x00000000
	SEC_GRP1COND8	0x00000000
	SEC_GRP0COND9	0x00000000
	SEC_GRP1COND9	0x00000000
	SEC_GRP0COND10	0x00000000
	SEC_GRP1COND10	0x00000000
	SEC_GRP0COND11	0x00000000
	SEC_GRP1COND11	0x00000000
	SEC_GRP0COND12	0x00000000
	SEC_GRP1COND12	0x00000000
	SEC_GRP0COND13	0x00000000
	SEC_GRP1COND13	0x00000000
	SEC_GRP0COND14	0x00000000
	SEC_GRP1COND14	0x00000000
	SEC_GRP0COND15	0x00000000
	SEC_GRP1COND15	0x00000000
	SEC_READONLY0	0x00000000

Module	Register Name	Setting value
	SEC_READONLY1	0x00000000
	SEC_READONLY2	0x00000000
	SEC_READONLY3	0x00000000
	SEC_READONLY4	0x00000000
	SEC_READONLY5	0x00000000
	SEC_READONLY6	0x00000000
	SEC_READONLY7	0x00000000
	SEC_READONLY8	0x00000000
	SEC_READONLY9	0x00000000
	SEC_READONLY10	0x00000000
	SEC_READONLY11	0x00000000
	SEC_READONLY12	0x00000000
	SEC_READONLY13	0x00000000
	SEC_READONLY14	0x00000000
	SEC_READONLY15	0x00000000
	SAFE_GRP0CR0	0x00000000
	SAFE_GRP1CR0	0x00000000
	SAFE_GRP0CR1	0x00000000
	SAFE_GRP1CR1	0x00000000
	SAFE_GRP0CR2	0x00000000
	SAFE_GRP1CR2	0x00000000
	SAFE_GRP0CR3	0x00000000
	SAFE_GRP1CR3	0x00000000
	SAFE_GRP0COND0	0x00000000
	SAFE_GRP1COND0	0x00000000
	SAFE_GRP0COND1	0x00000000
	SAFE_GRP1COND1	0x00000000
	SAFE_GRP0COND2	0x00000000
	SAFE_GRP1COND2	0x00000000
	SAFE_GRP0COND3	0x00000000
	SAFE_GRP1COND3	0x00000000
	SAFE_GRP0COND4	0x00000000
	SAFE_GRP1COND4	0x00000000
	SAFE_GRP0COND5	0x00000000
	SAFE_GRP1COND5	0x00000000
	SAFE_GRP0COND6	0x00000000
	SAFE_GRP1COND6	0x00000000
	SAFE_GRP0COND7	0x00000000
	SAFE_GRP1COND7	0x00000000
	SAFE_GRP0COND8	0x00000000
	SAFE_GRP1COND8	0x00000000
	SAFE_GRP0COND9	0x00000000
	SAFE_GRP1COND9	0x00000000
	SAFE_GRP0COND10	0x00000000
	SAFE_GRP1COND10	0x00000000
	SAFE_GRP0COND11	0x00000000
	SAFE_GRP1COND11	0x00000000
	SAFE_GRP0COND12	0x00000000
	SAFE_GRP1COND12	0x00000000
	SAFE_GRP0COND13	0x00000000
	SAFE_GRP1COND13	0x00000000
	SAFE_GRP0COND14	0x00000000
	SAFE_GRP1COND14	0x00000000
	SAFE_GRP0COND15	0x00000000
	SAFE_GRP1COND15	0x00000000
	SAFE_READONLY0	0x00000000
	SAFE_READONLY1	0x00000000
	SAFE_READONLY2	0x00000000
	SAFE_READONLY3	0x00000000
	SAFE_READONLY4	0x00000000

Module	Register Name	Setting value
	SAFE_READONLY5	0x00000000
	SAFE_READONLY6	0x00000000
	SAFE_READONLY7	0x00000000
	SAFE_READONLY8	0x00000000
	SAFE_READONLY9	0x00000000
	SAFE_READONLY10	0x00000000
	SAFE_READONLY11	0x00000000
	SAFE_READONLY12	0x00000000
	SAFE_READONLY13	0x00000000
	SAFE_READONLY14	0x00000000
	SAFE_READONLY15	0x00000000

(2) RT-SRAM protection settings

Loader sets the access protection of the RT-SRAM area.
Table 3-12 shows RT-SRAM register setting values.
Detail of the set value refers to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 22.1.4”.

Table 3-12 List of the access protection setting for RT-SRAM

Module	Register Name	Setting value
RT-SRAM	SECDIV0D	0x00000000
	SECDIV1D	0x000FF000
	SECDIV2D	0x000FF000
	SECDIV3D	0x000FF000
	SECDIV4D	0x000FF000
	SECDIV5D	0x000FF000
	SECDIV6D	0x000FF000
	SECDIV7D	0x000FF000
	SECDIV8D	0x000FF000
	SECDIV9D	0x000FF000
	SECDIV10D	0x000FF000
	SECDIV11D	0x000FF000
	SECDIV12D	0x000FF000
	SECDIV13D	0x000FF000
	SECDIV14D	0x000FF000
	SECCTR0D	0x00000000
	SECCTR1D	0x00000000
	SECCTR2D	0x00000000
	SECCTR3D	0x00000000
	SECCTR4D	0x00000000
	SECCTR5D	0x00000000
	SECCTR6D	0x00000000
	SECCTR7D	0x00000000
	SECCTR8D	0x00000000
	SECCTR9D	0x00000000
	SECCTR10D	0x00000000
	SECCTR11D	0x00000000
	SECCTR12D	0x00000000
	SECCTR13D	0x00000000
	SECCTR14D	0x00000000
	SECCTR15D	0x00000000

(3) System RAM protection settings

Loader sets the access protection of the System RAM area.
Table 3-13 shows AXI-Bus register setting values for System RAM.

Detail of the set value refers to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 15.1.4”.

Table 3-13 List of the access protection setting for System RAM

Module	Register Name	Setting value
AXI-bus	SPTDIVCR0	0x000E6300
	SPTDIVCR1	0x000FFFFF
	SPTDIVCR2	0x000FFFFF
	SPTDIVCR3	0x000FFFFF
	SPTDIVCR4	0x000FFFFF
	SPTDIVCR5	0x000FFFFF
	SPTDIVCR6	0x000FFFFF
	SPTDIVCR7	0x000FFFFF
	SPTDIVCR8	0x000FFFFF
	SPTDIVCR9	0x000FFFFF
	SPTDIVCR10	0x000FFFFF
	SPTDIVCR11	0x000FFFFF
	SPTDIVCR12	0x000FFFFF
	SPTDIVCR13	0x000FFFFF
	SPTDIVCR14	0x000FFFFF
	SPTCR0	0x00000000
	SPTCR1	0x00000000
	SPTCR2	0x00000000
	SPTCR3	0x00000000
	SPTCR4	0x00000000
	SPTCR5	0x00000000
	SPTCR6	0x00000000
	SPTCR7	0x00000000
	SPTCR8	0x00000000
	SPTCR9	0x00000000
	SPTCR10	0x00000000
	SPTCR11	0x00000000
	SPTCR12	0x00000000
	SPTCR13	0x00000000
	SPTCR14	0x00000000
	SPTCR15	0x00000000

(4) SDRAM protection settings

Loader sets the access protection of the SDRAM area.
Table 3-14 shows AXI-Bus register setting values for SDRAM.
Detail of the set value refers to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 15.1.4”.

Table 3-14 List of the access protection setting for SDRAM

Module	Register Name	Setting value
AXI-bus	DPTDIVCR0	0x00040000
	DPTDIVCR1	0x002FFFFF
	DPTDIVCR2	0x002FFFFF
	DPTDIVCR3	0x002FFFFF
	DPTDIVCR4	0x002FFFFF
	DPTDIVCR5	0x002FFFFF
	DPTDIVCR6	0x002FFFFF
	DPTDIVCR7	0x002FFFFF
	DPTDIVCR8	0x002FFFFF
	DPTDIVCR9	0x002FFFFF
	DPTDIVCR10	0x002FFFFF
	DPTDIVCR11	0x002FFFFF
	DPTDIVCR12	0x002FFFFF
	DPTDIVCR13	0x002FFFFF
	DPTDIVCR14	0x002FFFFF
	DPTCR0	0x00000000
	DPTCR1	0x00000000
	DPTCR2	0x00000000
	DPTCR3	0x00000000
	DPTCR4	0x00000000
	DPTCR5	0x00000000
	DPTCR6	0x00000000
	DPTCR7	0x00000000
	DPTCR8	0x00000000
	DPTCR9	0x00000000
	DPTCR10	0x00000000
	DPTCR11	0x00000000
	DPTCR12	0x00000000
	DPTCR13	0x00000000
	DPTCR14	0x00000000
	DPTCR15	0x00000000

3.3.12.11 Boot CR7

Loader supplies power to the CR7. After that, Loader release the reset of the CR7 to boot up the CR7.

3.3.12.12 Boot CA53

Loader supplies power to the CA53 and sets the entry point of the loaded binary image to boot address. After that, Loader release the reset of the CA53 to boot up the CA53.
If CA53 program #1 is a Secure Monitor, Loader writes the boot parameters of Secure Monitor to a specific address before boots up CA53.

3.3.13 Release RPC

Loader reinitialize RPC setting to HW initial value.

3.3.14 Release DMA

Loader reinitialize DMA setting to HW initial value.

3.3.15 Release SCIF

Loader reinitialize SCIF setting to HW initial value.

4. Memory

4.1 Memory layout

An example of a memory layout preset by Loader is shown in the Figure 4.1.
The source address, destination address and the number of transfer image of binary image is changeable.
Refer to Chapter 6.3.4.6 to change these information.
NOTE) Don't overwrite ICUMXA Loader on RT-SRAM when loading the binary image of RTOS.

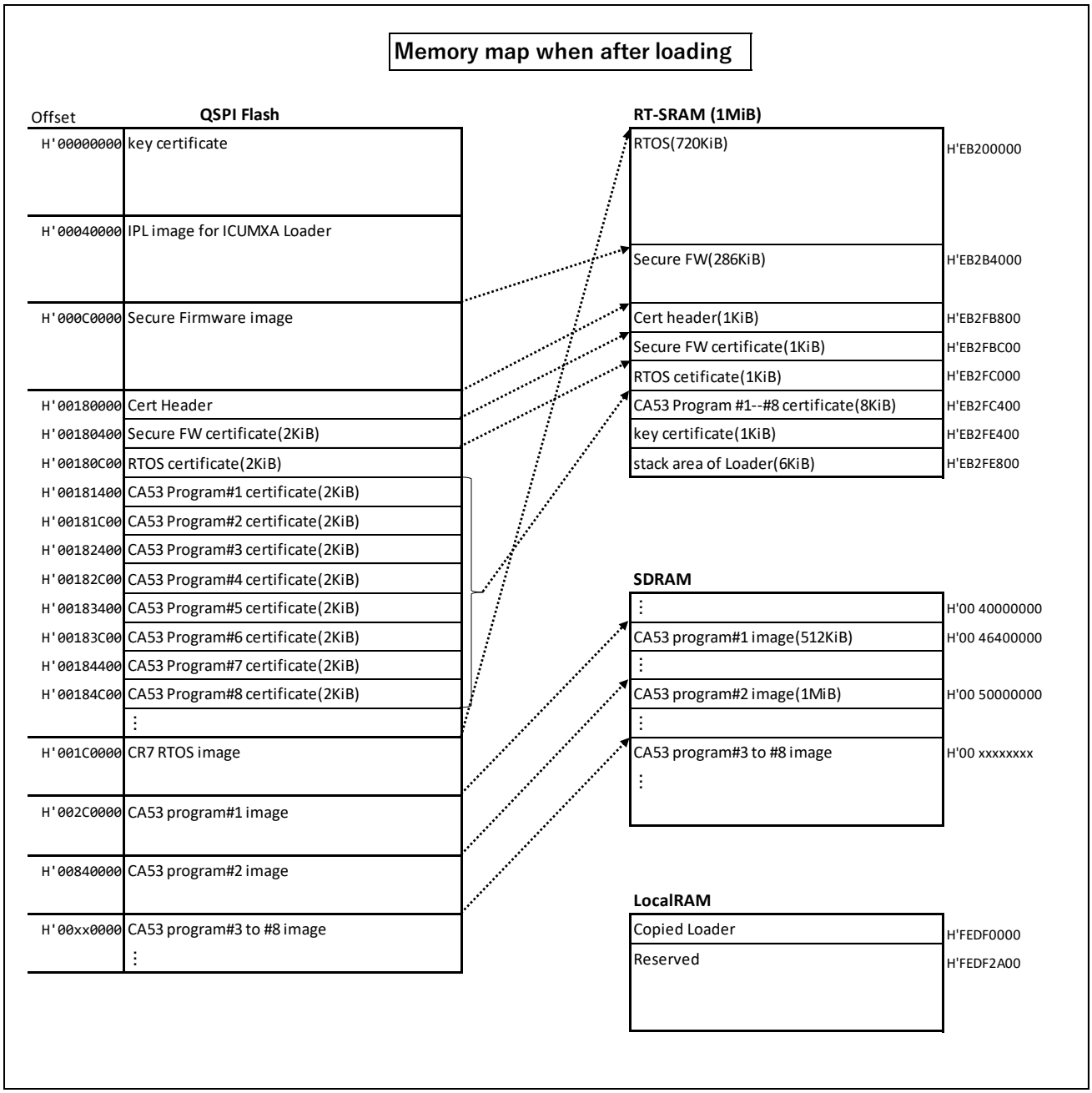


Figure 4.1 Memory layout

4.2 Release image

Refer to Table 4-1 about information of release image.
The table shows the information needed when writing data to Flash.

Table 4-1 Information list of release image

Filename	Program Top Address	Flash Save Address	Description
bootparam_sa0.srec	H'EB200000 *1	H'000000	Loader (Boot parameter)
icumxa_loader.srec	H'EB2D8000	H'040000	Loader
dummy_fw.srec	H'EB2B4000	H'0C0000	Dummy firmware (Instead of secure firmware)
cert_header_sa6.srec	H'EB200000 *2	H'180000	Loader (Certificate)
dummy_rtos.srec	H'EB200000	H'1C0000	Dummy RTOS (Instead of CR7 main OS)
bl31.srec	H'46400000	H'2C0000	BL31 Monitor
u-boot-elf-condor.srec	H'50000000	H'840000	U-boot

Note *1) Loader loads key certificate from H'EB200F00 to H'EB2FE400. For more detail, refer to 3.3.12.1.

Note *2) Loader loads 'Cert Header', 'Secure FW certificate', 'RTOS certificate' and 'CA53 program#n(n=1-8) certificate' from Flash to H'EB2FB800. These are reduced in size and loaded.

5. External Interface

There is no external interface for this module.

6. Integration

6.1 Directory configuration

Figure 6.1 shows the directory configuration when "V3H_ICUMXA_Loader_ <date> .zip" is expanded.

```
V3H_ICUMXA_Loader
    ├──common              # Common files directory
    │      ├──log          # log output directory
    │      └──mem_io       # memory mapped I/O directory
    ├──image_load          # Image loading directory
    ├──include             # Include directory
    ├──ip                  # IP's control directory
    │      ├──cpg          # CPG setting directory
    │      ├──ddr          # DDR setting directory
    │      ├──dma          # DMA driver directory
    │      ├──pfc          # Pin Function setting directory
    │      ├──qos          # QoS setting directory
    │      ├──rpc          # RPC driver directory
    │      └──wdt          # WDT driver directory
    ├──loader              # Loader main directory
    ├──protect             # Access protection directory
    │      ├──lifec        # LifeC setting directory
    │      └──memory       # memory protect setting directory
    ├──remap               # address remapping control directory
    ├──rom_api             # BootROM API Interface directory
    ├──tools               # Dummy certification
    │      └──dummy_create # Dummy create tools
    └────  Makefile        # Makefile
```

Figure 6.1 Directories for Loader of R-Car V3H

6.2 Integration Procedure

There is no external interface for this module.

6.3 How to

This chapter shows how to compile Loader.

6.3.1 Environment of components

This chapter describes how to prepare the environment for compiling Loader.
To check the operation of Loader, a sample source codes of the program to be loaded by Loader are included.

- Loader
- Dummy FW (Secure FW sample source code)
- Dummy RTOS (RTOS sample source code)
- Dummy CA53 program (CA53 Program#1 source code)

Table 6-1 to Table 6-4 show the environment for compiling each source code.

Table 6-1 Environment for compiling Loader

Item	tool name	Remarks
OS	Windows professional SP1(64bit)	-
compiler	Green Hills Software V800 Development Tool.	Release : 9.6.2 COMP_Version : 2015.1.7 MULTI_Version : 6.1.6
Terminal	Cygwin.	Cygwin Version : 2.10.0-1 Make Version : 4.2.1.2
Source code	V3H_ICUMXA_Loader_<date>.zip	Rev.0.2.0

Table 6-2 Environment for compiling Dummy FW

Item	File name	Remarks
OS	Windows professional SP1(64bit)	
compiler	Green Hills Software V800 Development Tool	Release : 9.6.2 COMP_Version : 2015.1.7 MULTI_Version : 6.1.6
Terminal	Cygwin	Cygwin Version : 2.10.0-1 Make Version : 4.2.1.2
source code	Dummy_FW_<date>.zip	

Table 6-3 Environment for compiling Dummy RTOS

Item	File name	Remarks
OS	Ubuntu 14.04LTS (64bit)	-
compiler	gcc-linaro-5.2-2015.11-2-x86_64_arm-eabi.tar.xz	-
source code	Dummy_RTOS_<date>.tar.gz	-

Table 6-4 Environment for compiling Dummy CA53

Item	File name	Remarks
OS	Ubuntu 14.04LTS (64bit)	-
compiler	gcc-linaro-5.2-2015.11-2-x86_64_aarch64-elf.tar.xz	-
source code	Dummy_CA53_Program_<date>.tar.gz	-

6.3.1.1 Prepare the compiling environment for Windows

This chapter explains the Windows compilation environment.

Source codes of Loader and Dummy FW are compiled on Windows.

(1) Prepare the GHS Compiler

The GHS compiler must be prepared by the user.

Install the GHS compiler according to Green Hills Software Products Installation Guide (GHS License Package).

(2) Prepare the Cygwin

Download the Cygwin installer from the site shown below.

Cygwin (https://www.cygwin.com/)

Launch the downloaded installer and install according to the instructions of the installer.

Install at least the package "make: The GNU version of the 'make' utility" always.

(3) Prepare the source code

Expand to the source code WORK directory in Table 6-1 and Table 6-2.

A directory of WORK is expressed as an arbitrary path of the user. The following explanation assumes WORK path.

6.3.1.2 Prepare the compiling environment for Linux

This chapter explains about compiling a Linux environment.

Source codes of Dummy RTOS and Dummy CA53 Program are compiled on Linux

(1) Prepare the GCC compiler

Create a WORK directory. Prepare the GCC compiler according to the following procedure.

```
$ cd $WORK
$ wget http://releases.linaro.org/components/toolchain/binaries/5.2-
  2015.11-2/arm-eabi/gcc-linaro-5.2-2015.11-2-x86_64_arm-eabi.tar.xz
$ tar xvf gcc-linaro-5.2-2015.11-2-x86_64_arm-eabi.tar.xz
$ wget https://releases.linaro.org/components/toolchain/binaries/5.2-
  2015.11-2/aarch64-elf/gcc-linaro-5.2-2015.11-2-x86_64_aarch64-elf.tar.xz
$ tar xvf gcc-linaro-5.2-2015.11-2-x86_64_aarch64-elf.tar.xz
```

(2) Prepare the source code

Locate the source code shown Table 6-3 and Table 6-4 to $WORK and Execute the following steps.

```
$ cd $WORK
$ tar xvf Dummy_RTOS_<date>.tar.gz
$ tar xvf Dummy_CA53_Program_<date>.tar.gz
```

6.3.2 Build option

This chapter explain the build options supported by Loader.

Note) Undefined value is treated as reserved word. Please do not use undefined value.

6.3.2.1 LOG_LEVEL

Loader provides a function to output logs. In the source code, you can output the log by executing the functions of NOTICE(), ERROR(), WARN(), INFO() and VERBOSE().
The log has an output level, and you can control the output of a log by specifying a level. A correspondence between log levels and functions is shown in Table 6-5.

This option can set value from 0 to 5. If this option is not set, the value is internally set to 2.

Table 6-5 Values of LOG_LEVEL and valid log display functions

LOG_LEVEL	Valid log display function
0	No log display function.
1	ERROR()
2	NOTICE(), ERROR(). [default]
3	NOTICE(), ERROR(), WARN().
4	NOTICE(), ERROR(), WARN(), INFO().
5	NOTICE(), ERROR(), WARN(), INFO(), VERBOSE().

6.3.2.2 RCAR_REWT_TRAINING

Select "periodic write DQ training" mode. If this option is not set, the value is set 1 internally.
"periodic write DQ training" adjusts write signal timings for LPDDR4 skew correction.
For details, refer to the 7.1.

Table 6-6 Association table for the RCAR_REWT_TRAINING value and Periodic write DQ training

RCAR_REWT_TRAINING	Periodic write DQ training
0	not available
1	available [default]

6.3.2.3 RCAR_DDR_REG_CEHCK

It can select whether execute internal bus interface check of DDR-PHY register.
About “internal bus interface check of DDR-PHY register”, refer to section 6.6 of 1.3.1 Related Document No.3.
By default, it is defined to '0' (Do not check DDR-PHY register).
Note) As a result of DDR-PHY register check, IPL keep running even if fault is detected.

Table 6-7 Association table for the RCAR_DDR_REG_CHECK setting

RCAR_DDR_REG_CHECK	Whether check DDR PHY register
0	Do not execute internal bus interface check of DDR-PHY register [default]
1	Execute internal bus interface check of DDR-PHY register
Others	Prohibited

6.3.2.4 ACC_PROT_ENABLE

Loader provides a function to set either enable or disable to an access protection by build options.
This option can set value 0 or 1. If this option is not set, the value is internally set to 1.

Table 6-8 Values of ACC_PROT_ENABLE and function of access protection setting

ACC_PROT_ENABLE	Access protection function
0	Disable a function of access protection setting.
1	Enable a function of access protection setting. [default]

6.3.2.5 CA53_PROG1_IS_SMONI

Loader provides a function to set a boot parameter of Secure Monitor by build option.
This build option explicitly specifies whether CA53 program#1 is Secure Monitor. If Secure Monitor is specified as the option, set a value in the boot parameter. The value to be set can be specified by another build option described in the next chapter.
This option can set value 0 or 1. If this option is not set, the value is internally set to 1.

Table 6-9 Value of CA53_PROG1_IS_SMONI and contents of CA53 program#1

CA53_PROG1_IS_SMONI	Contents of CA53 program#1
0	CA53 program#1 is not Secure Monitor.
1	CA53 program#1 is Secure Monitor. [default]

6.3.2.6 Secure Monitor parameter

If CA53 program#1 is Secure Monitor, the boot parameters of Secure Monitor are stored in a specific address by Loader. The parameters are generated at build time. Parameters can be changed with the following build options.
NOTE) Don’t set a value larger than the size described in "Bit width".

Table 6-10 Build options list for Secure Monitor parameters

Build option	Default value	Bit width
CA53_PROG2_ATTR	0x00000001	32bit.
CA53_PROG2_PC	0x0000000050000000	64bit
CA53_PROG2_SPSR	0x00000000000003C5	64bit
CA53_PROG2_ARG0	0x0000000000000000	64bit
CA53_PROG2_ARG1	0x0000000000000000	64bit
CA53_PROG2_ARG2	0x0000000000000000	64bit
CA53_PROG2_ARG2	0x0000000000000000	64bit
CA53_PROG2_ARG3	0x0000000000000000	64bit
CA53_PROG2_ARG4	0x0000000000000000	64bit
CA53_PROG2_ARG5	0x0000000000000000	64bit
CA53_PROG2_ARG6	0x0000000000000000	64bit
CA53_PROG2_ARG7	0x0000000000000000	64bit

6.3.3 How to build

This chapter explain the component build procedure.

6.3.3.1 How to build the Loader

This chapter explain the Loader build procedure.

Start Cygwin terminal.

Move to Loader directory and execute the make command.

```
> cd WORK/V3H_ICUMXA_Loader
> make clean
> make
```

Figure 6.2 Loader build example

Note) If post-software (e.g.U-boot) of BL31 or Secure Monitor, build with following build option.
(e.g. U-boot base address is 0x50000000)

```
> make CA53_PROG2_PC=0x50000000
```

Figure 6.3 Loader build option for post-software is U-boot

When the build is completed, the following binary image is output.

- ./build/release/bootparam_sa0.srec
- ./ build/release/cert_header_sa6.srec
- ./ build/release/icumxa_loader.srec

Build options can be used by writing after the make.

For example, if change to LOG_LEVEL=5, execute as follows.

```
> cd WORK/V3H_ICUMXA_Loader
> make clean
> make LOG_LEVEL=5
```

Figure 6.4 Build example using build option

6.3.3.2 How to build the Dummy FW

This chapter explain the Dummy FW build procedure.

Start Cygwin terminal.

Move to Dummy FW directory and execute the make command.

```
> cd WORK/Dummy_FW
> make clean
> make
```

Figure 6.5 Dummy FW build example

When the build is completed, the following binary image is output.

- ./ build/release/dummy_fw.srec

6.3.3.3 How to build the Dummy RTOS

This chapter explain the Dummy RTOS build procedure.

Move to Dummy RTOS directory and execute the make command.

```
$ cd $WORK /Dummy_RTOS
$ make clean
$ CROSS_COMPILE=~/gcc-linaro-5.2-2015.11-2-x86_64_arm-eabi/bin/arm-
  eabi- make
```

Figure 6.6 Dummy RTOS build example

When the build is completed, the following binary image is output.

- ./ dummy_rtos.srec

6.3.3.4 How to build the Dummy CA53 Program

This chapter explain the Dummy CA53 Program build.

```
$ cd $WORK /Dummy_CA53_Program
$ make clean
$ CROSS_COMPILE=~/ gcc-linaro-5.2-2015.11-2-x86_64_aarch64-
  elf/bin/aarch64-elf- make
```

Figure 6.7 Dummy CA53 program build example

When the build is completed, the following binary image is output.

- ./AArch64_output/AArch64_Dummy_CA53_Program.srec

6.3.4 How to customize

This chapter explain the loader customization procedure.

6.3.4.1 How to customize PFC and GPIO initialization

This chapter explains the procedure for customizing the PFC and GPIO initialization processing.
To customize PFC and GPIO initialization, change the file shown in Figure 6.8.

```
V3H_ICUMXA_Loader
 ├──include
 │          pfc.h
 │          pfc_register.h
 └──ip
     └──pfc
          pfc.c
```

Figure 6.8 File containing PFC and GPIO initialization process

The PFC initialization process is executed by “pfc_init” function in pfc.c. Figure 6.9 shows some of the actual code.

```
#define     GPSR1_DIGRF_CLKOUT      ((uint32_t)1U << 27U)
#define     GPSR1_DIGRF_CLKIN       ((uint32_t)1U << 26U)
#define     GPSR1_CANFD_CLK_A       ((uint32_t)1U << 25U)
#define     GPSR1_CANFD1_RX         ((uint32_t)1U << 24U)
#define     GPSR1_CANFD1_TX         ((uint32_t)1U << 23U)
            :
    /* initialize GPIO/perihperal function select */
    pfc_reg_write(PFC_GPSR0, 0x00000000);

    pfc_reg_write(PFC_GPSR1, GPSR1_DIGRF_CLKOUT
                           | GPSR1_DIGRF_CLKIN
                           | GPSR1_CANFD_CLK_A
                           | GPSR1_CANFD0_RX_A
                           | GPSR1_CANFD0_TX_A
                           | GPSR1_AVB_LINK
                           | GPSR1_AVB_PHY_INT
                           | GPSR1_AVB_MDC
                           | GPSR1_AVB_MDIO
                           | GPSR1_AVB_TXCREFCLK
                           | GPSR1_AVB_TD3
                           | GPSR1_AVB_TD2
                           | GPSR1_AVB_TD1
                           | GPSR1_AVB_TD0
                           | GPSR1_AVB_TXC
                           | GPSR1_AVB_TX_CTL
                           | GPSR1_AVB_RD3
                           | GPSR1_AVB_RD2
                           | GPSR1_AVB_RD1
                           | GPSR1_AVB_RD0
                           | GPSR1_AVB_RXC
                           | GPSR1_AVB_RX_CTL
```

```
                                    | GPSR1_IRQ0);
                :
```

Figure 6.9 Source code of PFC initialization process

A various module is assigned to each bit of PFC registers.
If information of module assigned to the PFC registers is required, refer to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 68.2”.
Loader provides definitions for accessing bits of PFC registers. Use that definition to set PFC.
Writing values to PFC registers, a specific procedure is required. Loader provides an API of “pfc_reg_write” to simplify the procedure. Use that function to write the PFC register.
As shown in Figure 6.9, set the PFC register in an API of “pfc_reg_write”.
Set the register address in the first argument, and the bit access definition in the second argument.

6.3.4.2 How to customize SDRAM setting

This chapter explains the procedure for customizing the SDRAM setting.
To customize SDRAM setting, change the file shown in Figure 6.10.

```
V3H_ICUMXA_Loader
└──ddr
        boot_init_dram.c
        boot_init_dram.h
        boot_init_dram_config.c
        boot_init_dram_regcheck.c
        boot_init_dram_regdef.h
        ddr.mk
        ddr_regdef.h
        dram.subfunc.c
        dram.subfunc.h
        init_dram_tbl_check.h
        init_dram_tbl_h3.h
        init_dram_tbl_h3ver2.h
        init_dram_tbl_m3.h
        init_dram_tbl_m3n.h
```

Figure 6.10 File containing SDRAM setting

SDRAM settings is executed by InitDram() function implemented in boot_init_dram.c.

Internal bus interface check of DDR-PHY register is executed by InitDram_regcheck() function implemented in boot_init_dram_regcheck.c. It can select whether execute InitDram_regcheck() function by specifying build option “RCAR_DDR_REG_CHECK”. Refer to 6.3.2.3 about “RCAR_DDR_REG_CHECK”.
Note) IPL does NOT check the return value of InitDram_regcheck(). So users need to implement what IPL should do if InitDram_regcheck() is failed.

6.3.4.3 How to customize QoS setting

This chapter explains the procedure for customizing the QoS setting.
To customize QoS setting, change the file shown in Figure 6.11.

```
V3H_ICUMXA_Loader
 ├──include
 │        qos.h
 │        qos_mstat.h
 │        qos_qoswt.h
 └──ip
     └──qos
          qos.c
```

Figure 6.11: File containing QoS setting

The DBSC setting process is executed by an API of “dbsc_setting” in qos.c.
The setting of QoS processing is executed by an API of “qos_init” in qos.c.
Loader provides a structure of "QOS_MSTAT_SETTING_TABLE" to set QoS in qos_mstat.h. Members of the structure are offset from top address of register, a setting value of MSTAT_FIX and a setting value of MSTAT_BE.
The setting of QoS is customized by changing these members.
Shown in Figure 6.12 a part of the actual code.

```
typedef struct{
        uint16_t offset;
        uint64_t mstat_fix;
        uint64_t mstat_be;
} QOS_MSTAT_SETTING_TABLE;

const QOS_MSTAT_SETTING_TABLE mstat_tbl[] = {
        /*offset,       MSTAT_FIX value,              MSTAT_BE value       */
        0x0000U,        0x000000000000FFFFU,          0x0010001005EFFC01U,
        0x0008U,        0x000000000000FFFFU,          0x0010001005EFFC01U,
        0x0010U,        0x000000000000FFFFU,          0x0010001005EFFC01U,
        0x0018U,        0x000000000000FFFFU,          0x0010001005EFFC01U,
```

Figure 6.12 Table of the setting value of QoS

A register address range of arbitration setting are shown in Table 6-11.

Register name	Address range	Setting value
QOSBW_FIX_QOS_BANK0	0xE67E0000—0xE67E031F	MSTAT_FIX
QOSBW_FIX_QOS_BANK1	0xE67E1000—0xE67E131F	MSTAT_FIX
QOSBW_BE_QOS_BANK0	0xE67E2000—0xE67E231F	MSTAT_BE
QOSBW_BE_QOS_BANK1	0xE67E3000—0xE67E331F	MSTAT_BE

Table 6-11 Arbitration setting table

6.3.4.4 How to customize access protection setting

This chapter explains the procedure for customizing the access protection setting.
To customize access protection setting, change the file shown in Figure 6.13.

```
V3H_ICUMXA_Loader
 ├──include
 │        acc_prot_init.h
 │        acc_prot_lifec.h
 │        acc_prot_memory.h
 └──protect
    ├──lifec
    │     acc_prot_lifec.c
    ├──memory
    │     acc_prot_memory.c
    │   acc_prot_init.c
```

Figure 6.13 File containing access protection setting

(1) LifeC protection setting

The LifeC protection setting is executed by an API of “acc_prot_lifec” in acc_prot_lifec.c.
Loader provides a structure of "LIFEC_SETTING_TABLE" to set access protection of LifeC. Members of the structure are address of LifeC and a setting value of LifeC. The access protection setting of LifeC is customized by changing these members.
Shown in Figure 6.14 a part of the actual code.

```
void acc_prot_lifec(void)
{
        uint32_t loop;
        const LIFEC_SETTING_TABLE lifec_reg_tbl[] = {
                {       LIFEC_SEC_GRP0CR2  ,       0x00000000U  },
                {       LIFEC_SEC_GRP1CR2  ,       0x00000000U  },
                {       LIFEC_SAFE_GRP0CR2 ,       0x00000000U  },
                {       LIFEC_SAFE_GRP1CR2 ,       0x00000000U  },
                                :
```

Figure 6.14 Table of the access protection setting value of LifeC

The access protection setting of LifeC is realized by setting plural registers of LifeC. The rule for determining the access level, refer to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 68.3.2” and “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 68.3.3”.
Various module is assigned to each bit of a register of LifeC. If information of module assigned to the LifeC registers is required, refer to “R-Car Series, 3rd Generation User's Manual: Hardware: Chapter 68.2”.

For example, if set Table 6-12 is required, change the blue part of the source code shown in Figure 6.15

Table 6-12 Example LifeC setting

Port	Module name	Security access level	Safety access level
Master	SCEG Secure core	3	2

```
void acc_prot_lifec(void)
{
        uint32_t loop;
        const LIFEC_SETTING_TABLE lifec_reg_tbl[] = {
                {       LIFEC_SEC_GRP0CR2  ,       0x00020000U  },
                {       LIFEC_SEC_GRP1CR2  ,       0x00020000U  },
                {       LIFEC_SAFE_GRP0CR2 ,       0x00000000U  },
                {       LIFEC_SAFE_GRP1CR2 ,       0x00020000U  },
                                :
```

Figure 6.15 Example of changing access protection setting of LifeC

(2) Memory protection setting

The memory protection setting is executed by APIs in acc_prot_memory.c.
It is possible to divide the memory area and set the access level individually by setting the AXI - Bus register and RT - SRAM register.
Functions are provided for each target memory. A list of target memory and functions is shown in Table 6-13.
Detailed contents of the functions are shown in the next section.

Table 6-13 Target memory and functions list of memory access protection

Target	Function
RT-SRAM	acc_prot_rt_sram()
System RAM	acc_prot_system_ram()
SDRAM	acc_prot_dram()

(a) RT-SRAM protection setting

The API of "acc_prot_rt_sram" provides access protection settings for RT-SRAM.
Loader provides a structure of "RT_SRAM_PROT" to set access protection of RT-SRAM. Table 6-14 shows the members of structure. This structure is structured by sixteen arrays.

Table 6-14 the Member of "RT_SRAM_PROT"

Member name	Explain
addr_off	Set the top address of the area to be protected on the RT-SRAM. For this parameter, specify the offset value from the top of RT-SRAM. *1
prot.reg_sec	The setting value of security level that can be written to the RT-SRAM access protection register.
prot.reg_saf	The setting value of safety level that can be written to the RT-SRAM access protection register.
prot.acc_sec	The setting value of security level that can be read and written to the specified memory area.
prot.acc_saf	The setting value of safety level that can be read and written to the specified memory area.

*1) The RT-SRAM area is divided by the area of addr_off[x] to addr_off[x+1].

The access protection setting of RT-SRAM is customized by changing these members.
Figure 6.16 shows a part of the actual code.

```
void acc_prot_rt_sram(void)
{
                    :
        RT_SRAM_PROT rt_sram_prot_data[16] = {
        /*   register              access                                  */
        /*   address               sec      safety    security          safety   */
        [0] = {RT_SRAM_ADDR_END,{W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
        [1] = {RT_SRAM_ADDR_END,{W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
        [2] = {RT_SRAM_ADDR_END,{W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
                    :
        [15] = {RT_SRAM_ADDR_END,{W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}}
```

Figure 6.16 Table of access protection setting value of RT-SRAM

In the rt_sram_prot_data[x].addr_off, must not to use a value smaller than the offset set in rt_sram_prot_data[x-1].addr_off.
The access protection setting can be set for each level of Security and Safety. According to the following rules, these setting values are defined in acc_prot_memory.h.

- A_BCDE
  - A : Set the character of R or W
    R indicates read permission
    W indicates write permission
  - B : Set the character of Y or N
    Y indicates read or write permission of Level 3 in Security or Safety.
    N indicates read or write non-permission of Level 3 in Security or Safety.
  - C : Set the character of Y or N
    Y indicates read or write permission of Level 2 in Security or Safety.
    N indicates read or write non-permission of Level 2 in Security or Safety.
  - D : Set the character of Y or N
    Y indicates read or write permission of Level 1 in Security or Safety.

N indicates read or write non-permission of Level 1 in Security or Safety.

- E : Set the character of Y or N
  Y indicates read or write permission of Level 0 in Security or Safety.
  N indicates read or write non-permission of Level 0 in Security or Safety.

For example, in case of setting as Table 6-15 is required, change the blue part of the source code shown in Figure 6.17

Table 6-15 Example access protection setting of RT-SRAM

RAM range	Security and Safety Level	Register write from Security	Register write from Safety	RAM read and write from Security	RAM read and write from Safety
0xEB200000 to 0xEB203FFF	3	Write	Write	Read / Write	Read / Write
	2	Not Access	Not Access	Read / Write	Read
	1	Not Access	Not Access	Read	Read
	0	Not Access	Not Access	Not Access	Not Access
0xEB204000 to 0xEB2FFFFF	3	Write	Write	Read / Write	Read / Write
	2	Write	Write	Read / Write	Read / Write
	1	Write	Write	Read / Write	Read / Write
	0	Write	Write	Read / Write	Read / Write

```
void acc_prot_rt_sram(void)
{
                        :
        RT_SRAM_PROT rt_sram_prot_data[16] = {
        /*      register                access                                  */
        /*      address                 sec       safety    security            safety   */
        [0] = {0x00000000U,         {W_YNNN,W_YNNN, R_YYYN | W_YYNN, R_YYYN | W_YNNN}},
        [1] = {0x00004000U,         {W_YYYY, W_YYYY,   R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
        [2] = {RT_SRAM_ADDR_END,{W_YYYY, W_YYYY,   R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
                        :
        [15] = {RT_SRAM_ADDR_END,{W_YYYY, W_YYYY,   R_YYYY | W_YYYY, R_YYYY | W_YYYY}}
```

Figure 6.17 Example of changing access protection setting of RT-SRAM

(b) System RAM protection setting

The API of "acc_prot_rt_sram" provides access protection settings of System RAM.
Loader provides a structure of "SYSTEM_RAM_PROT" to set access protection of System RAM. Table 6-16 shows the members of structure. This structure is structured by sixteen arrays.

Table 6-16 the Member of "SYSTEM_RAM_PROT"

Member name	Explain
addr	Set the top address of the area to be protected on the System RAM. *1
prot.reg_sec	The setting value of security level that can be written to the System RAM access protection register.
prot.reg_saf	The setting value of safety level that can be written to the System RAM access protection register.
prot.acc_sec	The setting value of security level that can be read and written to the specified memory area.
prot.acc_saf	The setting value of safety level that can be read and written to the specified memory area.

*1) The System RAM area is divided by the area of addr[x] to addr[x+1].

The access protection setting of System RAM is customized by changing these members.
Figure 6.18 shows some of the actual code.

```
void acc_prot_system_ram(void)
{
                        :
        SYSTEM_RAM_PROT system_ram_prot_data[16] = {
        /*      register                    access                                  */
        /*      address                       sec       safety     security            safety    */
        [0] = {SYSTEM_RAM_ADDR_END,{W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
        [1] = {SYSTEM_RAM_ADDR_END,{W_YYYY, W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
        [2] = {SYSTEM_RAM_ADDR_END,{W_YYYY, W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
                        :
        [15] = {SYSTEM_RAM_ADDR_END,{W_YYYY, W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}}
```

Figure 6.18 Table of access protection setting value of System RAM

In the system_ram_prot_data[x].addr must not to use a value smaller than the address set in the system_ram_prot_data[x-1].addr.
The setting values are defined the same as (a) RT-SRAM protection setting.

(c) SDRAM protection setting

The API of "acc_prot_dram()" provides setting values for SDRAM access protection settings.
Loader provides a structure of "DRAM_PROT" to set access protection of DRAM. Table 6-17 shows the members of structure. This structure is structured by sixteen arrays.

Table 6-17 the Member of "DRAM_PROT"

Member name	Explain
addr	Set the top address of the area to be protected on the SDRAM (40bit). *1
prot.reg_sec	The setting value of security level that can be written to the SDRAM access protection register.
prot.reg_saf	The setting value of safety level that can be written to the SDRAM access protection register.
prot.acc_sec	The setting value of security level that can be read and written to the specified memory area.
prot.acc_saf	The setting value of safety level that can be read and written to the specified memory area.

*1) The SDRAM area is divided by the area of addr[x] to addr[x+1].

The access protection setting of SDRAM is customized by changing these members.
Figure 6.19 shows some of the actual code.

```
void acc_prot_dram(void)
{
                        :
        DRAM_PROT dram_prot_data[16] = {
        /*    register                access                                    */
```

```
/*    address                    sec       safety     security            safety   */
[0] = {DRAM_ADDR_END, {W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
[1] = {DRAM_ADDR_END, {W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
[2] = {DRAM_ADDR_END, {W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
                :
[15] = {DRAM_ADDR_END, {W_YYYY,W_YYYY, R_YYYY | W_YYYY, R_YYYY | W_YYYY}},
```

Figure 6.19 Table of access protection setting value of SDRAM

In the dram_prot_data[x].addr, must not to use a value smaller than the address set in the dram_prot_data[x-1].
The setting values are defined the same as (a) RT-SRAM protection setting.

6.3.4.5 How to customize check transferable area

This chapter explains the procedure for customizing the check transferable area process.
To customize check transferable area process, change the file shown in Figure 6.20.

```
V3H_ICUMXA_Loader
├──include
│        image_load.h
└──image_load
         Image_load.c
```

Figure 6.20 File containing check transferable area process

The check transferable area process is executed by “check_load_area” function in image_load.c. Figure 6.21 shows some of the actual code.
When customizing the check transferable area process, be aware of destination address and size integer overflow.

```
static ADDRESS_RANGE placed_image[MAX_PLACED] = {
        [0] = {IPL_TOP, IPL_END}, /* Overwrite after RTOS range check. */
        [1] = {0U,0U},
:
:
/* Check image size */
if (len == 0U) {
        ERROR("image size error\n");
        panic;
}

/* Check whether source is overflow */
if (src > (UINT32_MAX - len)) {
        ERROR("overflow is occurred at source\n");
        ERROR("source address = 0x%x image size = 0x%x\n", src, len);
        panic;
} else {
        src_end = src + len - 1U;
}
:
:
/* Check source address range. */
if ((src < FLASH_BASE) || (FLASH_END < src_end)) {
        ERROR("check load area (source address)\n");
```

```
                ERROR("source address = 0x%x image size = 0x%x\n", src, len);
                panic;
        }
        :
        :
                /* check overlap */
                if ((dst >= placed_image[loop].topadd)
                        && (dst <= placed_image[loop].endadd)) {
                        overlap = OVERLAP_FLAG;
                } else if ((dst_end >= placed_image[loop].topadd)
                        && (dst_end <= placed_image[loop].endadd)) {
                        overlap = OVERLAP_FLAG;
                } else if ((dst < placed_image[loop].topadd)
                        && (placed_image[loop].endadd < dst_end)) {
                        overlap = OVERLAP_FLAG;
                } else {
                        loop++;
```

Figure 6.21 Source code of check transferable area process

The capacity of the transfer source QSPI Flash and transfer destination RAM is defined in image_load.h, and it can be adjusted by customizing the definitions. The information is shown in Table 6-18.

Table 6-18 Definition list of infomation of FlashROM and RAM

Define name	Default value	Description
FLASH_BASE	0x08000000U	QSPI Flash top address
FLASH_SIZE	0x04000000U	QSPI Flash size
FLASH_END	(FLASH_BASE + FLASH_SIZE) – 1U	QSPI Flash end address
DRAM_BASE	0x40000000U	SDRAM top address
DRAM_SIZE	0x80000000U	SDRAM size
DRAM_END	(DRAM_BASE + DRAM_SIZE) – 1U	SDRAM end address
RTSRAM_BASE	0xEB200000U	RT-SRAM top address
RTSRAM_SIZE	(1024U-18U)*1024U	RT-SRAM size excluding unusable areas.
RTSRAM_END	(RTSRAM_BASE + RTSRAM_SIZE) – 1U	RT-SRAM end address
IPL_TOP	0xEB2D8000U	Loader top address
IPL_END	0xEB2FFFFFU	Loader end address

6.3.4.6 How to customize dummy create

This chapter explains the procedure for customizing the dummy create.
The dummy create is a function to create a dummy certificate. A dummy certificate is a certificate that can be used when Loader executes in normal mode.
An integrity check is not executed in Normal Mode, the minimum information necessary for Loader to transfer the binary image needs to be stored in the certificate.
A certificate that contains the minimum necessary information is a dummy certificate.
To customize dummy create configuration, change the file shown in Figure 6.22.

```
V3H_ICUMXA_Loader
└─tools
    └──dummy_create
        │   sa0.c
        │   sa0.ld
        │   sa6.c
        └──sa6.ld
```

Figure 6.22 File containing dummy create configuration

The Loader output a dummy certificate along with the binary image of Loader at the Loader is compiled.
The following binary image is output.

- bootparam_sa0.srec
- cert_header_sa6.srec

(1) bootparam_sa0.srec

The bootparam_sa0.srec includes the following information.

- Load destination address of Loader
- Loader image size[word]

The srec file is generated by compiling sa0.c.
The file of sa0.c contains entry point of the Loader and Loader image size. The information is shown in Table 6-19.

Table 6-19 Definition list of changeable information of sa0.c

Define name	Default value	Description
LOADER_ADDR	0xEB2D8000	Load entry point of the Loader
LOADER_SIZE	128 * 1024 / 4	Loader image size[word]*1

*1) Image size is set in a “word” unit. (e.g. image size is: 0x00008000[word] = 0x00020000[byte])

(2) cert_header_sa6.srec

The cert_header_sa6.srec includes the following information.

- Cert Header
- Certificates of each binary images (only the transfer address and the image size are included)

The srec file is generated by compiling sa6.c.
The file of sa6.c contains Cert Header and certificate information. The information is shown in Table 6-20.

Table 6-20 Definition list of changeable information of sa6.c

Define name	Default value	Description
CA53_IMAGE_NUM	0x00000002	Cert Header : Number of CA53 Program
SECURE_FW_SRC_ADDRESS	0x000C0000	Cert Header : Offset of address for Secure FW
RTOS_SRC_ADDRESS	0x001C0000	Cert Header : Offset of address for RTOS
CA53_PROG_01_SRC_ADDRESS	0x002C0000	Cert Header : Offset of address for CA53 Program #1
CA53_PROG_02_SRC_ADDRESS	0x00840000	Cert Header : Offset of address for CA53 Program #2
CA53_PROG_03_SRC_ADDRESS	0x00000000	Cert Header : Offset of address for CA53 Program #3
CA53_PROG_04_SRC_ADDRESS	0x00000000	Cert Header : Offset of address for CA53 Program #4
CA53_PROG_05_SRC_ADDRESS	0x00000000	Cert Header : Offset of address for CA53 Program #5
CA53_PROG_06_SRC_ADDRESS	0x00000000	Cert Header : Offset of address for CA53 Program #6
CA53_PROG_07_SRC_ADDRESS	0x00000000	Cert Header : Offset of address for CA53 Program #7
CA53_PROG_08_SRC_ADDRESS	0x00000000	Cert Header : Offset of address for CA53 Program #8
SECURE_FW_ADDRESS	0xEB2B4000	Certificate(Secure FW) : Load address
SECURE_FW_DST_SIZE	0x00011200	Certificate(Secure FW) : Image size[word] *1
RTOS_ADDRESS	0xEB200000	Certificate (RTOS) : Load address
RTOS_DST_SIZE	0x0002D000	Certificate(RTOS) : Image size[word] *1
CA53_PROG_01_ADDRESS	0x46400000	Certificate(CA53 Program #1) : Load address
CA53_PROG_01_SIZE	0x00020000	Certificate(CA53 Program #1) : Image size[word] *1
CA53_PROG_02_ADDRESS	0x50000000	Certificate(CA53 Program #2) : Load address
CA53_PROG_02_SIZE	0x00040000	Certificate(CA53 Program #2) : Image size[word] *1
CA53_PROG_03_ADDRESS	0x00000000	Certificate(CA53 Program #3) : Load address
CA53_PROG_03_SIZE	0x00000000	Certificate(CA53 Program #3) : Image size[word] *1
CA53_PROG_04_ADDRESS	0x00000000	Certificate(CA53 Program #4) : Load address
CA53_PROG_04_SIZE	0x00000000	Certificate(CA53 Program #4) : Image size[word] *1
CA53_PROG_05_ADDRESS	0x00000000	Certificate(CA53 Program #5) : Load address
CA53_PROG_05_SIZE	0x00000000	Certificate(CA53 Program #5) : Image size[word] *1
CA53_PROG_06_ADDRESS	0x00000000	Certificate(CA53 Program #6) : Load address
CA53_PROG_06_SIZE	0x00000000	Certificate(CA53 Program #6) : Image size[word] *1
CA53_PROG_07_ADDRESS	0x00000000	Certificate(CA53 Program #7) : Load address
CA53_PROG_07_SIZE	0x00000000	Certificate(CA53 Program #7) : Image size[word] *1
CA53_PROG_08_ADDRESS	0x00000000	Certificate(CA53 Program #8) : Load address
CA53_PROG_08_SIZE	0x00000000	Certificate(CA53 Program #8) : Image size[word] *1

*1) Image size is set “word” unit. (e.g. the image size of the CA53 Program #1 is : 0x00020000 [word] = 0x00080000[byte])

The following shows the examples of customization according to purpose.

Example 1: Change the source address a binary image of CA53 Program #1

In case of changing the CA53 Program #1 binary image position of QSPI FLASH (source address) from 0x002C0000 to 0x00640000.

- Step1: Edit the defined value in code of sa6.c.

To change the value of the source address of CA53 program #1 is as shown in Figure 6.23.
After that please compile Loader and output cert_header_sa6.srec file.

```
/* Source address on flash for CA53 program(1) */
#define CA53_PROG_01_SRC_ADDRESS        (0x00640000U)
```

Figure 6.23 Example of changing source address of CA53 program #1

- Step2: Writing to QSPI Flash of binary image

When writing a binary image to QSPI FLASH, please change the writing position of QSPI FLASH and the binary image file to be written as shown in Table 6-21.

Table 6-21 Information list for writing a binary image of example 1

Filename	Program Top Address	Flash Save Address	Description
bootparam_sa0.srec	H'EB200000	H'000000	Loader(Boot parameter)
icumxa_loader.srec	H'EB2D8000	H'040000	Loader
cert_header_sa6.srec	H'EB200000	H'180000	Loader(Certification)
dummy_fw.srec	H'EB2B4000	H'0C0000	Secure FW
dummy_rtos.srec	H'EB200000	H'1C0000	RTOS
dummy_ca53_program_01.srec	H'46400000	H'640000	CA53 Program #1

Example 2: Change the destination address a binary image of CA53 Program #1.

In case of changing the CA53 Program #1 binary image of the destination address from 0x46400000 to 0x49000000.

- Step1: Edit the defined value in code of sa6.c.

To change the value of the destination address of CA53 program #1 is as shown in Figure 6.24.
After that please compile Loader and output cert_header_sa6.srec file.

```
/* Destination address for CA53 program(1) */
#define CA53_PROG_01_ADDRESS     (0x49000000U)
#define CA53_PROG_01_ADDRESSH    (0x00000000U)
```

Figure 6.24 Example of changing destination address of CA53 program #1

- Step2: Writing to QSPI Flash of binary image.

When writing a binary image to QSPI FLASH, please change the writing position of program top address and the binary image file to be written as shown in Table 6-22.

Table 6-22 Information list for writing a binary image of example 2

Filename	Program Top Address	Flash Save Address	Description
bootparam_sa0.srec	H'EB200000	H'000000	Loader(Boot parameter)
icumxa_loader.srec	H'EB2D8000	H'040000	Loader
cert_header_sa6.srec	H'EB200000	H'180000	Loader(Certification)
dummy_fw.srec	H'EB2B4000	H'0C0000	Secure FW
dummy_rtos.srec	H'EB200000	H'1C0000	RTOS
dummy_ca53_program_01.srec*1	H'49000000	H'2C0000	CA53 Program #1

*1) If change the destination address of the binary image, it is necessary to change the entry point and rebuild the binary image.

Example 3: Add the transferring a binary image of CA53 Program #2 to #4.

In case of adding transfer of the binary image of CA53 Program #2 to #4.
Table 6-23 shows the information of the binary image to be added.

Table 6-23 Added the transferring images of Example 3

filename	Program Top Address	Flash Save Address	Image size[word]
<CA53 Program #2 image name>.srec	H'48000000	H'740000	H'400000
<CA53 Program #3 image name>.srec	H'50000000	H'640000	H'40000
<CA53 Program #4 image name>.srec	H'50140000	H'1AC0000	H'40000

- Step1: Edit the defined value in code of sa6.c.

To change the information of the CA53 program#2 to #4 is as shown in Figure 6.25.
After that please compile Loader and output cert_header_sa6.srec file.

```
/* CA53 program load num */
#define CA53_IMAGE_NUM                  (0x00000004U)
...

/* ----------- customized ----------- */
/* Reserved */
#define CA53_PROG_02_SRC_ADDRESS        (0x00740000U)
/* Reserved */
#define CA53_PROG_03_SRC_ADDRESS        (0x00640000U)
/* Reserved */
#define CA53_PROG_04_SRC_ADDRESS        (0x01AC0000U)
...

/* ----------- customized ----------- */
/* Reserved */
#define CA53_PROG_02_ADDRESS            (0x48000000U)
#define CA53_PROG_02_ADDRESSH           (0x00000000U)
#define CA53_PROG_02_SIZE               (0x00400000U)
/* Reserved */
#define CA53_PROG_03_ADDRESS            (0x50000000U)
#define CA53_PROG_03_ADDRESSH           (0x00000000U)
#define CA53_PROG_03_SIZE               (0x00040000U)
/* Reserved */
#define CA53_PROG_04_ADDRESS            (0x50140000U)
#define CA53_PROG_04_ADDRESSH           (0x00000000U)
#define CA53_PROG_04_SIZE               (0x00040000U)
```

Figure 6.25 Example of add certificate information of added binary image

- Step2: Writing to QSPI Flash of binary image.

When writing a binary image to QSPI FLASH, please change the writing position of program top address and the binary image file to be written as shown in Table 6-24.

Table 6-24 Information list for writing a binary image of example 3

Filename	Program Top Address	Flash Save Address	Description
bootparam_sa0.srec	H'EB200000	H'000000	Loader(Boot parameter)
icumxa_loader.srec	H'EB2D8000	H'040000	Loader
cert_header_sa6.srec	H'EB200000	H'180000	Loader(Certification)
dummy_fw.srec	H'EB2B4000	H'0C0000	Secure FW
dummy_rtos.srec	H'EB200000	H'1C0000	RTOS
dummy_ca53_program_01.srec	H'46400000	H'2C0000	CA53 Program #1
<CA53 Program #2 image name>.srec	H'48000000	H'740000	CA53 Program #2
<CA53 Program #3 image name>.srec	H'50000000	H'640000	CA53 Program #3
<CA53 Program #4 image name>.srec	H'50140000	H'1AC0000	CA53 Program #4

7.Appendix

7.1 Periodic write DQ training

7.1.1 Outline

The adjustment of write signal timing for LPDDR4 skew correction is needed as to amount of temperature change of R-Car chip.

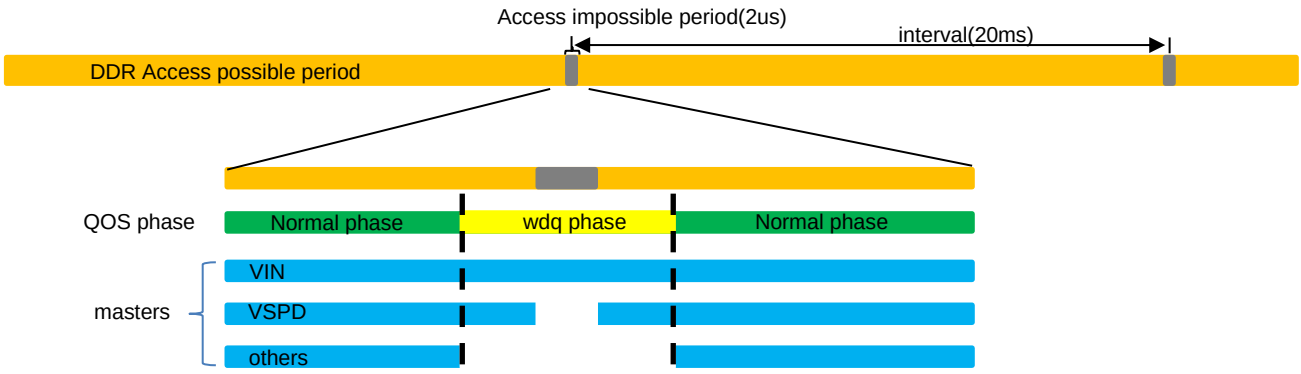
Renesas proposes “periodic write DQ training” as a measure of the adjustment.

The interval of periodic write DQ training is 20ms and a write DQ training period is about 2us. In the period, each of read/write requests cannot access DDR. Therefore, sensitive latency master like VSPD or VIN may report error because of latency deterioration.

In order to resolve the issue, Renesas prepares another QoS control during periodic write DQ training to realize the adjustment of write signal timing without latency deterioration.

7.1.2 QoS control of periodic write DQ training

DBSC preferentially accepts access requests of VIN, ISP, Ethernet and GigaEther and VSPD to reduce the latency deterioration impact. VIN, ISP, Ethernet and GigaEther requests are always buffered not only in “Normal phase” but also in “wdq phase”. VSPD requests are buffered except for “Access impossible period” in wdq phase. The others are only buffered in “Normal phase”.
In case that interval of periodic write DQ training is set to 20ms, there is hardly each master bandwidth reduction. (0.02% or less)



REVISION HISTORY	Initial Program Loader User's Manual for ICUMXA: Software

Rev.	Date	Description	
		Page	Summary
0.1.0	Jan. 31, 2018	—	New creation.
0.2.0	Jun. 29, 2018	—	Renamed of chapter. 1.2.2 Display the booting message Previous name is "Display the starting message". 1.2.3 Decide the boot mode Previous name is "Decide the starting mode". 3.2 Module configuration Previous name is "Module Configuration". 3.3.2 PFC and GPIO initialization Previous name is "PFC initialization". 3.3.2 Start supply of clock signal to module Previous name is "Module stop initialization". 3.3.8 Access protection setting Previous name is "Security and Safety setting". 4.2 Memory layout Previous name is "Memory Layout". 6.3.1 Environment of components Previous name is "Components". 6.3.5.4 How to build the Loader Previous name is "Build the Loader". 6.3.5.5 How to build the Dummy FW Previous name is "Build the Dummy FW". 6.3.6.3 How to build the Dummy RTOS Previous name is "Build Dummy RTOS. 6.3.6.3.6.4 How to build the Dummy CA53 Program Previous name is "Build the Dummy CA53 Program". 6.3.7.1 How to customize PFC and GPIO initialization Previous name is "PFC / GPIO setting". 6.3.7.3 How to customize QoS setting Previous name is "QoS arbitration setting". 6.3.7.5 How to customize dummy create Previous name is "dummy create settings".
		—	Removed of Chapter 3.3.6 RPC initialization 4.1 Memory Constitution 6.3.2 Environment 6.3.5 Prepare the GHS compiler and build 6.3.5.1 Prepare the GHS compiler 6.3.5.2 Prepare the Make for Windows 6.3.5.3 Prepare the source code 6.3.6 Prepare the GCC compiler and build 6.3.6.1 Prepare the compiler 6.3.6.2 Prepare the source code 6.3.7.2 SDRAM setting 6.3.7.4 Image load area check setting
		—	Added of Chapter 3.3.6.1 LifeC protection setting 3.3.6.2 RTOS protection setting 3.3.6.3 System RAM protection setting 3.3.6.4 SDRAM protection settings 3.3.7 Boot log output 3.3.8 Image loading 3.3.8.1 Loading key certificate 3.3.8.2 Loading content certificate 3.3.8.3 Loading [binary image] 3.3.8.4 SDRAM/PHY initialization 3.3.8.5 QoS initialization

Rev.	Date	Description	
		Page	Summary
			3.3.8.6 Finish of DMA transfer 3.3.8.7 Integrity check to [binary image] 3.3.8.8 Boot [CPU] 3.3.9 Stop supply of the clock signal to module 3.3.10 Release SCIF 6.3.1.1 Prepare the compiling environment for Windows 6.3.1.2 Prepare the compiling environment for Linux 6.3.2.1 LOG_LEVEL 6.3.2.2 RACR_QOS_TYPE 6.3.2.3 ACC_PROT_ENABLE 6.3.2.4 LIFEC_NON_SECURE_MASTER 6.3.2.5 CA53_PROG1_IS_SMONI 6.3.2.6 Secure Monitor parameter 6.3.3 How to build 6.3.4.3 How to customize access protection setting
		—	Moved of chapter 3.3.2 SCIF initialization Previous number of chapter is 3.3.4. 3.3.4 PFC and GPIO initialization Previous number of chapter is 3.3.2. 3.3.5 DMA initialization Previous number of chapter is 3.3.7. 3.3.6 Access protection setting Previous number of chapter is 3.3.8. 3.3.8.4 SDRAM/PHY initialization setting Previous number of chapter is 3.3.5. 3.3.8.5 QoS initialization Previous number of chapter is 3.3.9. 4.1 Memory Layout Previous number of chapter is 3.3.9. 6.3.2 Build option Previous number of chapter is 6.3.3. 6.3.3.1 How to build the Loader Previous number of chapter is 6.3.5.4. 6.3.3.2 How to build the Dummy FW Previous number of chapter is 6.3.5.4. 6.3.3.3 How to build the Dummy RTOS Previous number of chapter is 6.3.5.4. 6.3.3.4 How to build the Dummy CA53 program Previous number of chapter is 6.3.5.4. 6.3.4 How to customize Previous number of chapter is 6.3.7. 6.3.4.1 How to customize PFC and GPIO initialization Previous number of chapter is 6.3.7.1 6.3.4.2 How to customize QoS setting Previous number of chapter is 6.3.7.3 6.3.4.4 How to customize dummy create Previous number of chapter is 6.3.7.5
		1	1.1 Overview Changed contents of description.
		1	1.2 Function Changed contents of description. Changed name of Table 1-1. Changed Explanation of description in Table 1-1.
		2	1.2.1 Hardware initialization Changed contents of description.
		3	1.2.2 Display the booting message Changed contents of description. Changed name of Figure 1-2.

Rev.	Date	Description	
		Page	Summary
			Updated Example in booting message of Figure 1-2.
		5	1.2.3 Decide the boot mode Changed contents of description. Changed name of Figure 1-3. Changed Figure 1-3.
		6	1.2.4 Loading the image Changed contents of description. Changed name of Figure 1-4 and 1-5. Changed Figure 1-4 and 1-5. Removed Figure 1-6 and 1-7.
		8	1.2.5 Integrity check for the image using a certificate Changed contents of description. Changed name of Figure 1-6. Changed Figure 1-6.
		9	1.2.6 Releasing used resource Changed contents of description.
		9	1.2.7 Starting RTOS Changed contents of description.
		9	1.2.8 Starting CA53 program Changed contents of description.
		9	1.2.9 Starting Secure FW Changed contents of description.
		12	2 Terminology Added LCM in Table 2-1. Changed description of LCS in Table 2-1.
		13	3.1 Hardware Environment Changed description of host PC in Table 3-1 and Figure 3-1.
		14	3.2 Module configuration Fixed of typo. Changed description of AXI-Bus uses.
		16	3.3 Processing Flow Diagram Changed contents of description. Changed Figure 3-3 and Figure 3-4.
		18	3.3.1 CPU initialization Added initialization of the RAM in Table 3-3.
		18	3.3.2 SCIF initialization Changed contents of description. Removed Table 3-8. Changed name of Table3-5.
		18	3.3.3 Module stop initialization Changed contents of description. Changed name of Table 3-6. Changed Table 3-6.
		19	3.3.4 PFC and GPIO initialization Changed contents of description. Changed Name of Table 3-7. Removed address and HW initial value of Table 3-7.
		21	3.3.5 DMA initialization Changed contents of description.
		21	3.3.6 Access protection setting Changed contents of description. Changed separated the LifeC protection setting as a Chapter.
		27	3.3.8.4 SDRAM/PHY initialization Changed contents of description.
		27	3.3.8.5 QoS initialization Changed contents of description.
		29	4.1 Memory layout Changed contents of description.

Rev.	Date	Description	
		Page	Summary
			Changed Figure 4-1.
		31	6.1 Directory configuration Changed contents of description. Changed Figure 6-1.
		33	6.3 How to Changed contents of description.
		33	6.3.1 Environment of components Changed contents of description. Changed name of Table 6-1 to 6-4. Changed Table 6-1 to 6-4.
		35	6.3.2.1 Build option Changed contents of description.
		35	6.3.2.1 LOG_LEVEL Changed contents of description. Changed name of Table 6-5. Changed the item name of Table 6-5.
		36	6.3.2.1 RCAR_QOS_TYPE Changed contents of description. Changed name of Table 6-5.
		39	6.3.3.1 How to build the Loader Changed contents of description. Added Figure 6-2 and 6-3.
		40	6.3.3.2 How to build the Dummy FW Changed contents of description. Added Figure 6-4.
		40	6.3.3.3 How to build the Dummy RTOS Changed contents of description. Added Figure 6-5.
		40	6.3.3.4 How to build the Dummy CA53 program Changed contents of description. Added Figure 6-6.
		41	6.3.4 How to customize Changed contents of description.
		41	6.3.4.1 How to customize PFC and GPIO initialization Changed contents of description. Added Figure 6-7 and Figure 6-8. Removed Table 6-8.
		42	6.3.4.2 How to customize QoS setting Changed contents of description. Added Figure 6-9 and Figure 6-10. Added Table 6-11. Removed Figure 6-2 Removed Table 6-10.
		48	6.3.4.4 How to customize dummy create Changed contents of description. Changed contents of description in bootparam_sa0.srec. Changed contents of description in cert_header_sa6.srec. Changed contents of description in Example 1 to 3. Added Figure 6-17 to 6-20. Changed name of Table 6-18 to 6-23. Changed Table 6-18 to 6-21, and Table 6-23.
0.3.0	Aug. 24, 2018	—	Changed memory map. 1.2.4 Loading the image Added the Local RAM in Figure 1-4. 4.1 Memory layout Changed the memory map of Figure 4-1. 6.3.4.4 (2) cert_header_sa6.srec Changed the default values in Table 6-19. 6.3.4.4 (2) Example1:

Rev.	Date	Description	
		Page	Summary
			6.3.4.4 (2) Example2: Changed Program Top Address 6.3.4.4 (2) Example3: Add the transferring a binary image of CA53 Program #2 to #4 Changed Table 6-22 and Table 6-23. Changed Figure 6-20.
		—	Changed boot sequence. 3.3 Processing Flow Diagram Changed the flow chart of Figure 3-3 and Figure 3-4. 3.3.2 Data copy to Local RAM 3.3.8.8 Set boot address of CR7 3.3.8.10 Boot CR7 Added of chapter. 3.3.8.9 Access protection setting Moved of chapter from 3.3.6. 3.3.8.11 Boot CA53 Renamed of chapter. Changed description of CPU to CA53.
		37	6.3.2.5 Secure Monitor parameter Changed the default value of CA53_PROG2_PC.
		37	6.3.2.4 LIFEC_NON_SECURE_MONITOR Removed of chapter.
0.4.0	Oct. 4, 2018	16	3.3 Processing Flow Diagram Changed the flow chart of Figure 3-3.
		18	3.3.4 WDT initialization Added of Chapter
		—	Added MFIS setting. 3.3.5 Start supply of clock signal 3.3.12 Stop supply of clock signal to module Changed module stop setting value. 3.3.9 MFIS initialization Added of chapter.
		21	3.3.8 RPC initialization Added of chapter.
		23	3.3.11.3 Loading [binary image] Fixed of typo.
0.5.0	Dec. 21, 2018	-	Changed the execution timing of WDT initialization. 3.3 Processing Flow Diagram Changed flow chart of Figure 3-3. 3.3.2 WDT initialization 3.3.3 Data copy to Local RAM 3.3.4 SCIF initialization Moved of chapter
		-	Changed module standby setting. 3.3.5 Start supply of clock signal to module Changed description of chapter and Table 3-6. 3.3.12 Stop supply of clock signal to module Removed of chapter.
		18	3.3 Processing Flow Diagram Added Figure 3-5.
		30	3.3.12 Release RPC Added of Chapter.
		33	6.1 Directory configuration Changed Figure 6-1.
1.0.0	Apr. 24, 2019	-	Update QoS setting. 6.3.2.2 RCAR_REWT_TRAINING Change build option from “RCAR_QOS_TYPE” to “RCAR_REWT_TRAINING”. 6.3.4.2 How to customize QoS setting Changed Figure 6-9.

Rev.	Date	Description	
		Page	Summary
		-	Changed Memory map 3.3.11.2 Loading content certificate Changed description in the loading process of content certificate. 4.1 Memory layout Changed the memory map of Figure 4-1.
		1	1.1. Overview Added the description that the loader is not support the Safety Requirement Specification.
		24	3.3.11.3 Loading [binary image] Added SDRAM to the transfer destination of the binary image.
		24	3.3.11.6 Finish of DMA transfer Added the counter reset process of Window Watchdog Timer in description.
		31	4.1 Memory layout Added specification that must not overwrite ICUMXA Loader until completed loading binary image of RTOS.
		46	6.3.4.3 How to customize access protection setting (2) Removed unnecessary restrictions.
		55	7. Appendix Added of Chapter
1.0.1	Jun. 18, 2019	55	7.1.2 QoS control of periodic write DQ training Updated modules related to periodic write DQ training.
1.0.3	Jul. 24, 2019	-	Added of Chapter 6.3.2.3 RCAR_DDR_REG_CHECK 6.3.4.2 How to customize SDRAM setting
		10	1.3.1 Related Documant Added related document.
1.0.6	Dec. 13, 2019	-	Added Check Transferable area 3.3 Processing Flow Diagram Changed Figure 3.4. 3.3.11.3 Check transferable area of [binary image] 6.3.4.5 How to customize check transferable area Added of Chapter
2.0.0	Feb. 17, 2021	1	1.2 Function Figure 1.1 Modified ‘LOADER’ to ‘Secure Firmware’.
		16	3.3 Processing Flow Diagram Figure 3.3 Added DMA release.
		17	3.3 Processing Flow Diagram Figure 3.4 Modified DDR initialization and QoS initialization order.
		30	3.3 Processing Flow Diagram Added 3.3.13 Release DMA.
2.0.1	Apr. 13, 2021	32	4.1 Memory layout Figure 4.1 Modified program top address of CA53 program#2 image.
		33	4 Memory Added 4.2 Release image.
		42	6.3.2.6 Secure Monitor parameter Table 6.10 Modified default value of CA53_PROG2_PC from 0x0000000048000000 to 0x0000000050000000.
		56	6.3.4.6 How to customize dummy create Table 6.20 Modified CA53_PROG_01_SIZE from 0x00010000 to 0x00020000. Modified CA53_PROG_02_ADDRESS from 0x48000000 to 0x50000000. Modified CA53_PROG_02_SIZE from 0x00400000 to 0x00040000.

Rev.	Date	Description	
		Page	Summary
2.0.2	Jul. 5, 2021	3	1.2.2 Display the booting message Added Condor-I to description. Figure 1.2 Booting message of Condor / Condor-I board Updated boot log.
		10	1.3.1 Related Document Table 1.3 Related Document Added Condor-I.
		13	3.1 Hardware Environment Table 3.1 Hardware environment (R-Car Series, 3rd Generation) Added Condor-I.
		14	3.2 Module configuration Added Condor-I to description.
		20	3.3.5 Start supplying clock signal to module Table 3.6 List of Module Stop setting Modified Setting value of “SMSTPCR1” from 0xF7FFFFFF to 0xF5FFFFFF. Removed row of “SMSTPCR8” because its Setting value is same as Initial value of HW.
		21	3.3.6 PFC and GPIO initialization Added Condor-I to description.
		33	4.2 Release image Table 4.1 Information list of release image Modified release image and Description.
		42	6.3.2.6 Secure Monitor parameter Table 6.10 Build options list for Secure Monitor parameters Modified Default value of “CA53_PROG2_ATTR” from 0x00000000 to 0x00000001.
		43	6.3.3.1 How to build the Loader Added Figure 6.3 Loader build option for post-software is U-boot.
2.0.3	Aug. 26, 2021	16	3.3 Processing Flow Diagram Figure 3.3 Added EDC initialization.
		24	3.3 Processing Flow Diagram Added 3.3.10 EDC initialization.

CONFIDENTIAL

Initial Program Loader for ICUMXA
User's Manual: Software

Publication Date: Rev.0.1.0 Jan. 31, 2018
Rev.2.0.3 Aug. 26, 2021

Published by: Renesas Electronics Corporation

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

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

Renesas Electronics America Inc. Milpitas Campus
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics America Inc. San Jose Campus
6024 Silver Creek Valley Road, San Jose, CA 95138, USA
Tel: +1-408-284-8200, Fax: +1-408-284-2775

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai 200333, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, #06-02 Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 3.1

Initial Program Loader for ICUMXA
User's Manual

Renesas Electronics Corporation