RX Family

Firmware Update Module Using Firmware Integration Technology

Introduction

This application note describes the firmware update module using Firmware Integration Technology (FIT). The module is referred to below as the firmware update FIT module.

By using the FIT module, users can easily incorporate firmware update functionality and secure boot functionality into their applications. This application note explains how to use the firmware update FIT module and how to incorporate its API functions into user applications.

The release package associated with this application note includes a demo project. You can confirm the basic operation of the firmware update functionality by following the steps described in section 4, Demo Project, to build an environment to run the demo.

Operation Confirmation Devices

RX130 Group

RX140 Group

RX230, RX231 Group

RX23E-A Group

RX23E-B Group

RX24T Group

RX26T Group

RX65N, RX651 Group

RX66N Group

RX66T Group

RX660 Group

RX671 Group

RX72M Group

RX72N Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Related Application Notes

Application notes related to this application note are listed below. Refer to them in conjunction with this document.

- Firmware Integration Technology User's Manual (R01AN1833)
- RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- RX Family Flash Module Using Firmware Integration Technology (R01AN2184)
- RX Family SCI Module Using Firmware Integration Technology (R01AN1815)
- RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)

Target Compliers

- C/C++ Compiler Package for RX Family from Renesas Electronics
- GCC for Renesas RX
- IAR C/C++ Compiler for RX

For details of the environments on which operation has been confirmed, refer to 6.1, Confirmed Operation Environments.

Contents

1. Overview 8
1.1 About the Firmware Update Module 8
1.2 Configuration of Firmware Update Module 9
1.3 Firmware Update Operation 10
1.3.1 Dual-Bank Method 11
1.3.1.1 Operation of Dual-Bank Method 11
1.3.2 Linear Mode Partial Update Method 12
1.3.2.1 Operation of Linear Mode Partial Update Method 12
1.3.3 Linear Mode Full Update Method 13
1.3.3.1 Operation of Linear Mode Full Update Method 13
1.4 Initial State of Firmware Update 14
1.4.1 Initial State of Dual-Bank Method Settings Utilizing Renesas Image Generator 14
1.4.2 Initial State of Linear Mode Partial Update Method Settings Utilizing Renesas Image Generator 14
1.4.3 Initial State of Linear Mode Full Update Method Settings Utilizing Renesas Image Generator 15
1.4.4 Initial State of Dual-Bank Method Settings Utilizing Bootloader 15
1.4.5 Initial State of Linear Mode Partial Update Method Settings Utilizing Bootloader 16
1.4.6 Initial State of Linear Mode Full Update Method Settings Utilizing Bootloader 16
1.5 Package Contents 17
1.6 API Overview 19

2. API Information 20
2.1 Hardware Requirements 20
2.2 Software Requirements 20
2.3 Supported Toolchains 20
2.4 Header Files 20
2.5 Integer Types 20
2.6 Compile Settings 21
2.7 Sample Project Code Sizes 24
2.8 Arguments 28
2.9 Return Values 28
2.10 Adding the FIT Module to Your Project 28
2.11 “for”, “while” and “do while” statements 29
2.12 Implementation Examples of APIs 30
2.12.1 Example of Bootloader Implementation of Dual-Bank Method 30
2.12.2 Example of Application Program Implementation of Dual-Bank Method 31
2.12.3 Example of Bootloader Implementation of Linear Mode Partial Update Method 32
2.12.4 Example of Application Program Implementation of Linear Mode Partial Update Method 33
2.12.5 Example of Bootloader Implementation of Linear Mode Full Update Method 34
2.12.6 Example Implementation of API when Used in Non-blocking Mode 35

3. API Functions 37
3.1 R_FWUP_Open Function 37
3.2 R_FWUP_Close Function 37
3.3 R_FWUP_IsExistImage Function 37
3.4 R_FWUP_EraseArea Function 38
3.5 R_FWUP_GetImageSize Function 38
3.6 R_FWUP_WriteImage Function 38
3.7 R_FWUP_VerifyImage Function 39
3.8 R_FWUP_ActivateImage Function 39
3.9 R_FWUP_ExecImage Function 39
3.10 R_FWUP_SoftwareReset Function 40
3.11 R_FWUP_SoftwareDelay Function 40
3.12 R_FWUP_GetVersion Function 40
3.13 R_FWUP_WriteImageHeader Function 41
3.14 R_FWUP_WriteImageProgram Function 41
3.15 Wrapper Functions 42
3.15.1 Wrapper Functions (r_fwup_wrap_verify.c, h) 42
3.15.1.1 r_fwup_wrap_sha256_init Function 42
3.15.1.2 r_fwup_wrap_sha256_update Function 42
3.15.1.3 r_fwup_wrap_sha256_final Function 42
3.15.1.4 r_fwup_wrap_verify_ecdsa Function 43
3.15.1.5 r_fwup_wrap_get_crypt_context Function 43
3.15.2 Wrapper Functions (r_fwup_wrap_com.c, h) 44
3.15.2.1 r_fwup_wrap_disable_interrupt Function 44
3.15.2.2 r_fwup_wrap_enable_interrupt Function 44
3.15.2.3 r_fwup_wrap_software_delay Function 44
3.15.2.4 r_fwup_wrap_software_reset Function 45
3.15.3 Wrapper Functions (r_fwup_wrap_flash.c, h) 46
3.15.3.1 r_fwup_wrap_flash_open Function 46
3.15.3.2 r_fwup_wrap_flash_close Function 46
3.15.3.3 r_fwup_wrap_flash_erase Function 46
3.15.3.4 r_fwup_wrap_flash_write Function 47
3.15.3.5 r_fwup_wrap_flash_read Function 47
3.15.3.6 r_fwup_wrap_bank_swap Function 47

4. Demo Project 48
4.1 Demo project Structure 48
4.2 Operating environment preparation 49
4.2.1 Installing TeraTerm 49
4.2.2 Installing the Python execution environment 49
4.2.3 Installing the OpenSSL execution environment 49

4.2.4 Installing the Flash Writer 50
4.2.5 USB serial conversion board 50
4.3 Execution environment preparation 51
4.3.1 Generating Keys for Signature Generation and Verification 51
4.3.2 Preparing the execution environment for Renesas Image Generator 51
4.4 Demo Project Execution Procedure 52
4.4.1 Dual Bank Method 52
4.4.1.1 Execution Environment 52
4.4.1.2 Building The Demo Project 52
4.4.1.3 Creating Initial and Update Images 54
4.4.1.4 Programming the Initial Image 55
4.4.1.5 Executing a Firmware Update 55
4.4.2 Operation of Linear Mode Partial Update Method 56
4.4.2.1 Execution Environment 56
4.4.2.2 Building The Demo Project 56
4.4.2.3 Creating Initial and Update Images 57
4.4.2.4 Programming the Initial Image 59
4.4.2.5 Executing a Firmware Update 59
4.4.3 Operation of Linear Mode Full Update Method 61
4.4.3.1 Execution Environment 61
4.4.3.2 Building The Demo Project 61
4.4.3.3 Creating Initial and Update Images 62
4.4.3.4 Programming the Initial Image 63
4.4.3.5 Executing a Firmware Update 63

5. Renesas Image Generator 65
5.1 Image Generation Methods 65
5.1.1 Initial Image Generation Method 67
5.1.2 Update Image Generation Method 67
5.2 Image File 68
5.2.1 Update Image File 68
5.2.2 Initial Image File 70
5.3 Parameter File 72
5.3.1 Contents of Parameter File 73
5.3.2 How to generate an image with a flash size different from the demo project 76
5.3.3 How to prevent data flash data from being included in the image 77

6. Appendices 78
6.1 Confirmed Operation Environments 78
6.2 Operating Environment for Demo Project 83
6.2.1 Operation Confirmation Environment for RX130 83
6.2.1.1 Memory map of demo project for half-surface update method in linear mode 84

6.2.1.2 Memory map of demo project for full update method in linear mode 87
6.2.2 Operation Confirmation Environment for RX140 90
6.2.2.1 Memory map of demo project for half-surface update method in linear mode 91
6.2.2.2 Memory map of demo project for full update method in linear mode 94
6.2.3 Operation Confirmation Environment for RX231 97
6.2.3.1 Memory map of demo project for half-surface update method in linear mode 98
6.2.3.2 Memory map of demo project for full update method in linear mode 101
6.2.4 Operation Confirmation Environment for RX23E-A 104
6.2.4.1 Memory map of demo project for half-surface update method in linear mode 105
6.2.4.2 Memory map of demo project for full update method in linear mode 108
6.2.5 Operation Confirmation Environment for RX23E-B 111
6.2.5.1 Memory map of demo project for half-surface update method in linear mode 112
6.2.5.2 Memory map of demo project for full update method in linear mode 115
6.2.6 Operation Confirmation Environment for RX24T 118
6.2.6.1 Memory map of demo project for half-surface update method in linear mode 119
6.2.6.2 Memory map of demo project for full update method in linear mode 122
6.2.7 Operation Confirmation Environment for RX26T 125
6.2.7.1 Memory map of dual bank method demo project 126
6.2.7.2 Memory map of demo project for half-surface update method in linear mode 129
6.2.7.3 Memory map of demo project for full update method in linear mode 132
6.2.8 Operation Confirmation Environment for RX65N 135
6.2.8.1 Memory map of dual bank method demo project 136
6.2.8.2 Memory map of demo project for half-surface update method in linear mode 139
6.2.8.3 Memory map of demo project for full update method in linear mode 142
6.2.9 Operation Confirmation Environment for RX66T 145
6.2.9.1 Memory map of demo project for half-surface update method in linear mode 146
6.2.9.2 Memory map of demo project for full update method in linear mode 149
6.2.10 Operation Confirmation Environment for RX660 152
6.2.10.1 Memory map of demo project for half-surface update method in linear mode 153
6.2.10.2 Memory map of demo project for full update method in linear mode 156
6.2.11 Operation Confirmation Environment for RX671 159
6.2.11.1 Memory map of dual bank method demo project 160
6.2.11.2 Memory map of demo project for half-surface update method in linear mode 163
6.2.11.3 Memory map of demo project for full update method in linear mode 166
6.2.12 Operation Confirmation Environment for RX72N 169
6.2.12.1 Memory map of dual bank method demo project 170
6.2.12.2 Memory map of demo project for half-surface update method in linear mode 173
6.2.12.3 Memory map of demo project for full update method in linear mode 176
6.3 How to debug the demo project 179
6.4 Open source license information used in the demo project 188

7. Notes 189
7.1 Notes on Transition from Bootloader to Application. 189
7.2 Notes when using with DATFRX 189
7.3 Security measures for the bootloader area 189
7.4 When creating a new RX130 project in GCC environment 190

Revision History 192

1. Overview

1.1 About the Firmware Update Module

A firmware update is a process in which a device overwrites its own firmware, the software that controls the device's hardware, with a new version of the firmware (called the "update image" in this document) obtained through unspecified means. Firmware updates may be applied to fix bugs, add new functions, or improve performance.

The firmware update module is middleware that, when firmware update functionality is added to the user's system, provides the following functionality as its components:

- Functionality for importing the update image to the MCU via a communication interface
- Functionality for validating the update image (ECDSA NIST P-256 and SHA256 are used for validation.)
- Functionality for programming the update image to the on-chip flash memory (self-programming)
- Functionality for activating the update image

Generally, a firmware update system comprises two programs: an application program providing firmware update functionality and a bootloader providing secure boot functionality used to validate the first program.

The bootloader functionality is essential to the proper functioning of the firmware update. It guarantees that the sequence of processing that composes the firmware update, including validation of the update image, is legitimate.

The firmware update module for the RX Family provides functionality for the following three firmware update methods.

- Dual-bank method
- Linear mode partial update method
- Linear mode full update method

A tool (Renesas Image Generator) for creating firmware images is provided as a utility. Renesas Image Generator can generate the following types of images for use by the firmware update module.

- Initial image: An image file containing the bootloader and application program that is programmed using Flash Writer at the time of initial system configuration (extension: mot).
- Update image: An image file containing the firmware update (extension: rsu).

1.2 Configuration of Firmware Update Module

Figure 1.1 shows the configuration of the modules in the bootloader and application program incorporating the firmware update module, and Table 1.1 lists the modules used in the bootloader and application program.

The update image received by the communication interface is self-programmed to the on-chip flash memory of the target device via the firmware update module and the flash memory driver.

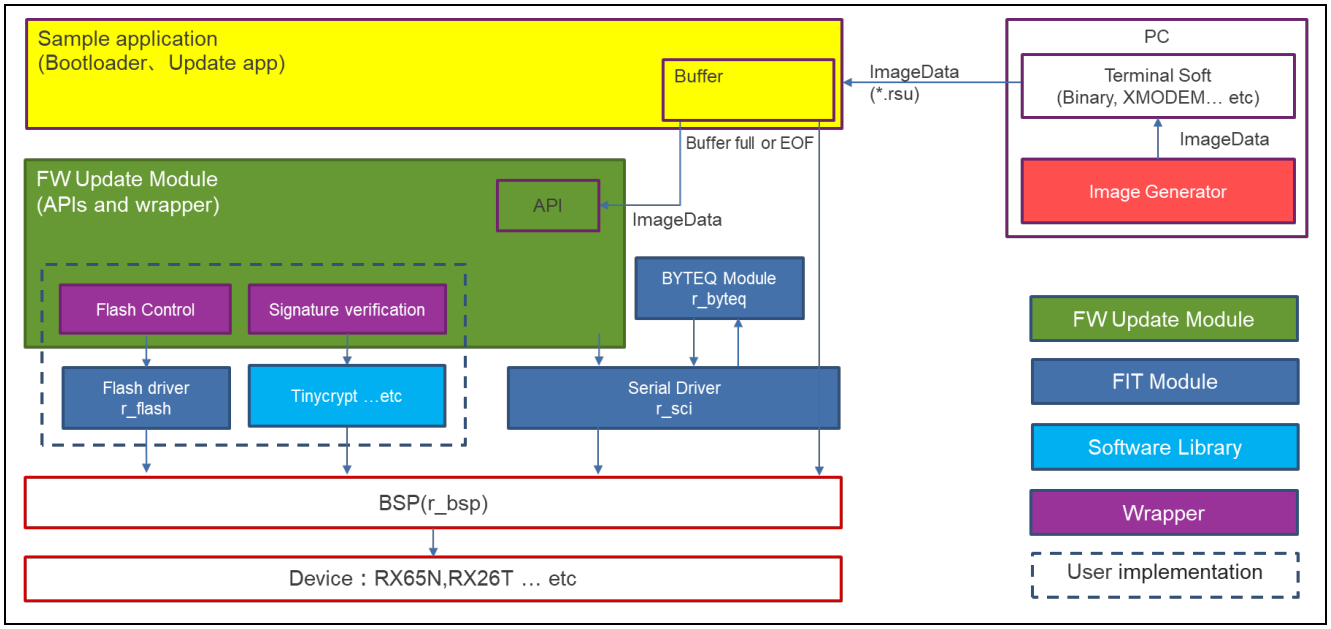


Figure 1.1 Configuration of Modules in Sample Bootloader and Application Program

Table 1.1 List of External Modules Used in Sample Bootloader and Application Program

Type	Application Note (Document No.)	FIT Module
BSP	RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)	r_bsp
Device driver	RX Family Flash Module Using Firmware Integration Technology (R01AN2184)	r_flash
Device driver	RX Family SCI Module Using Firmware Integration Technology (R01AN1815)	r_sci
Middleware	RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)	r_byteq

1.3 Firmware Update Operation

The firmware update module for the RX Family supports both the dual mode and the linear mode of the MCU's on-chip flash memory.

Dual mode uses the hardware's dual bank function to store the firmware to be updated (update image) on the buffer plane and then swap banks with the main plane to provide a dual bank method of updating..

For linear mode, two methods are provided: one in which the firmware update (update image) is stored temporarily on the buffer plane and another in which it is programmed directly to the main plane.

- Main plane: Area for storing the image used for booting
- Buffer plane: Area for storing the image to be applied as an update

The method of writing the update image directly to the main plane allows all of the internal flash memory to be used as the main plane, but since there is no buffer plane, it is not possible to restore the firmware to its pre-update state in the event of an update failure.

The update method support status varies by device and flash memory capacity, as detailed below.

Table 1.2 Supported Update Methods for Each Product

Flash / Product	4MB	2MB	1,5MB	1MB	756KB	512KB	384KB	256KB	128KB
RX130	-	-	-	-	-	b	b	b	b
RX140	-	-	-	-	-	-	-	b	b
RX231/230	-	-	-	-	-	b	b	b	b
RX23E-A	-	-	-	-	-	-	-	b	b
RX23E-B	-	-	-	-	-	-	-	b	b
RX24T	-	-	-	-	-	b	b	b	b
RX26T	-	-	-	-	-	a/b	-	b	b
RX65N/651	-	a/b	a/b	b	b	b	-	-	-
RX66N	a/b	b	-	-	-	-	-	-	-
RX66T	-	-	-	b	-	b	-	b	-
RX660	-	-	-	b	-	b	-	-	-
RX671	-	a/b	a/b	b	-	-	-	-	-
RX72M	a/b	b	-	-	-	-	-	-	-
RX72N	a/b	b	-	-	-	-	-	-	-

a: Dual-Bank Method
b: Linear Mode Partial Update Method / Full Update Method
-: Not supported
red text: Sample program

1.3.1 Dual-Bank Method

The update image is stored in the buffer plane in the on-chip flash memory, and, after it is validated, the banks are swapped, exchanging the main plane and buffer plane.
his method allows the application program to contain the firmware update functionality.
This means that if the firmware update fails before bank swapping occurs, the pre-update image in the main plane can be launched to retry the firmware update.
Since the on-chip flash memory is divided into two portions by the dual-bank functionality, the size of the on-chip flash memory available to store the application program is equal to the size of one of the two portions into which the on-chip flash memory has been divided minus the size of the bootloader.

1.3.1.1 Operation of Dual-Bank Method

The update image is stored in the buffer plane using the dual-bank functionality of the on-chip flash memory, and the firmware update is accomplished by using the bank-swapping functionality to exchange the banks.

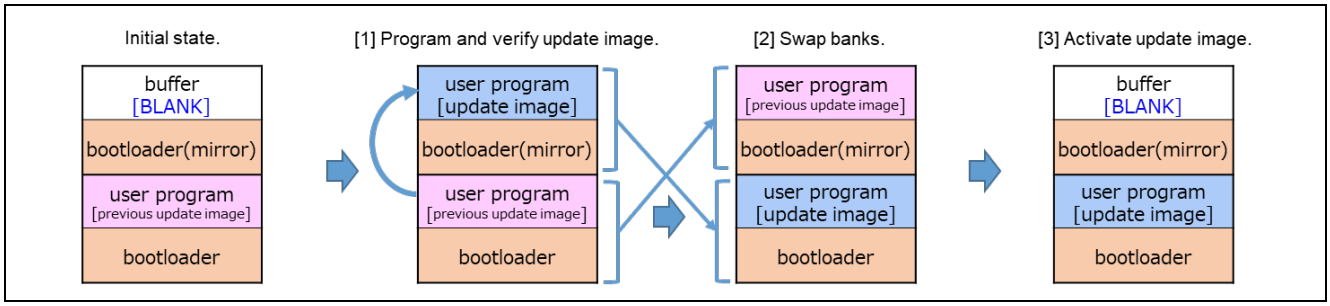


Figure 1.2 Operation of Dual-Bank Method

[1] Program and verify update image.
The previous update image (application program) stored in the main plane is used to program the update image to the buffer plane and verify it.

[2] Swap banks.
If verification is successful, the banks are swapped.

[3] Activate update image.
The buffer plane is erased by the bootloader.
(The demo program does not erase the buffer side. If you need to erase the image before updating for rollback measures, please add a process to erase the buffer side image.)

1.3.2 Linear Mode Partial Update Method

The update image is stored temporarily in the buffer plane in the on-chip flash memory, and, after it is validated, it is self-programmed to the main plane. This method allows the application program to contain the firmware update functionality. This means that if the firmware update fails before self-programming to the main plane occurs, the pre-update image in the main plane can be launched to retry the firmware update. The size that can store the application program is half the size of the remaining internal flash memory minus the bootloader.

1.3.2.1 Operation of Linear Mode Partial Update Method

This method divides the on-chip flash memory into a main plane and a buffer plane and then temporarily stores the update image in the buffer plane. Firmware is updated by storing the update image on the buffer plane and copying it from the buffer plane to the main plane.

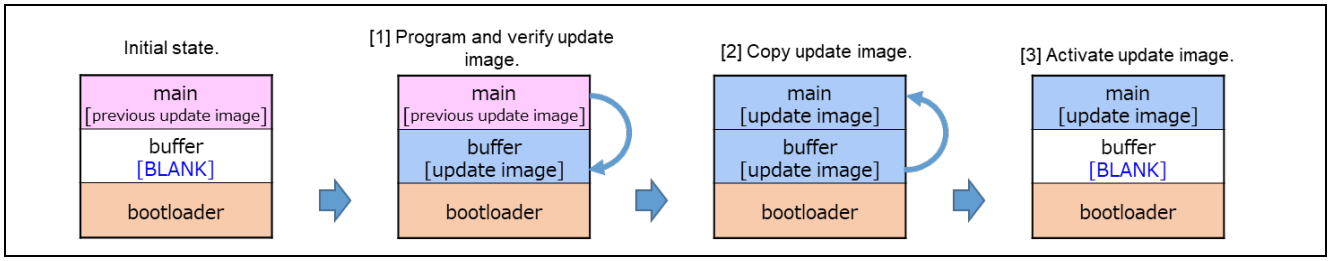


Figure 1.3 Operation of Partial Update Method

[1] Program and verify update image.
The previous update image (application program) stored in the main plane is used to program the update image to the buffer plane and verify it.

[2] Copy update image.
If verification is successful, the system is reset, the main plane is erased by the bootloader, and the updated image is copied from the buffer plane to the main plane.

[3] Activate update image.
The buffer plane is erased by the bootloader.
(The demo program does not erase the buffer side. If you need to erase the image before updating for rollback measures, please add a process to erase the buffer side image.)

1.3.3 Linear Mode Full Update Method

The update image is self-programmed to the main plane, after which it is validated. This method requires the bootloader to contain the firmware update functionality. This means that if the firmware update fails, the bootloader functionality can be used to retry the firmware update. The functionality of the application program cannot be used until the firmware update succeeds.
The size that can store the application program is the remaining size of the internal flash memory minus the bootloader.

1.3.3.1 Operation of Linear Mode Full Update Method

This method of writing the update image directly to the main plane allows all of the internal flash memory to be used as the main plane, but since there is no buffer plane, it is not possible to restore the firmware to its pre-update state in the event of an update failure.

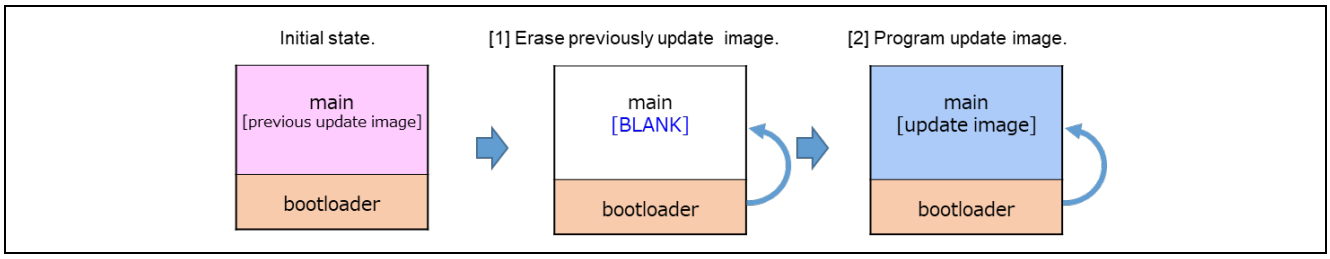


Figure 1.4 Operation of Full Update Method

[1] Erase previously update image.
The previous update image (application program) stored in the main plane configures the data indicating updates to the main plane and then applies a reset. After this, the bootloader runs and erases the initial image from the main plane.

[2] Program update image.
The bootloader downloads the update image from an external source and programs it to the main plane. The programmed update image is verified, and if verification is successful, the update image is activated.

1.4 Initial State of Firmware Update

To set the firmware update system using the firmware update module to the initial state, build the system by writing the initial image generated by the Renesas Image Generator to the built-in flash memory with a flash writer or similar device.

As an alternative method, it is also possible to build the system by first writing only the bootloader with a flash writer, etc., and then writing the updated image of the application program with the bootloader function.

1.4.1 Initial State of Dual-Bank Method Settings Utilizing Renesas Image Generator

The following figure shows the construction of the initial state of the dual-bank method using the Renesas Image Generator.

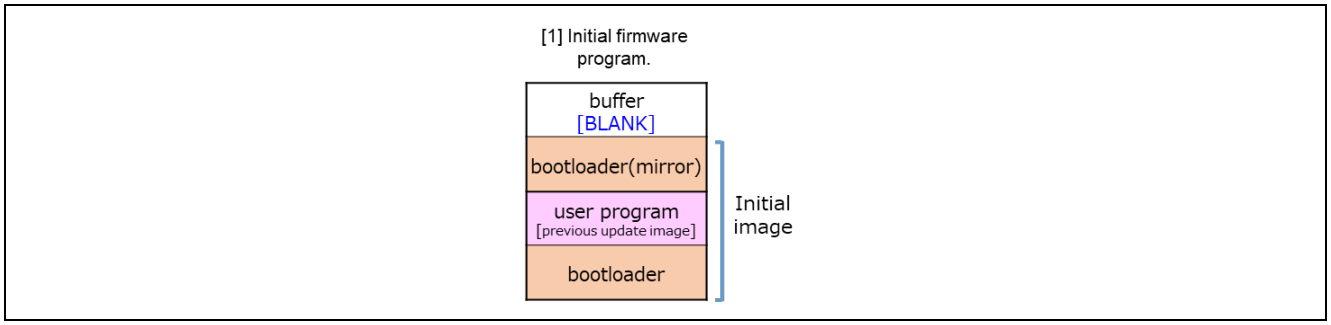


Figure 1.5 Initial Firmware Update Settings Utilizing Renesas Image Generator (Example of Dual-Bank Method)

[1] Program the initial image

The initial image is programmed to the on-chip flash memory using a tool such as Flash Writer.

1.4.2 Initial State of Linear Mode Partial Update Method Settings Utilizing Renesas Image Generator

The following figure shows the construction of the initial state of the partial update method using the Renesas Image Generator.

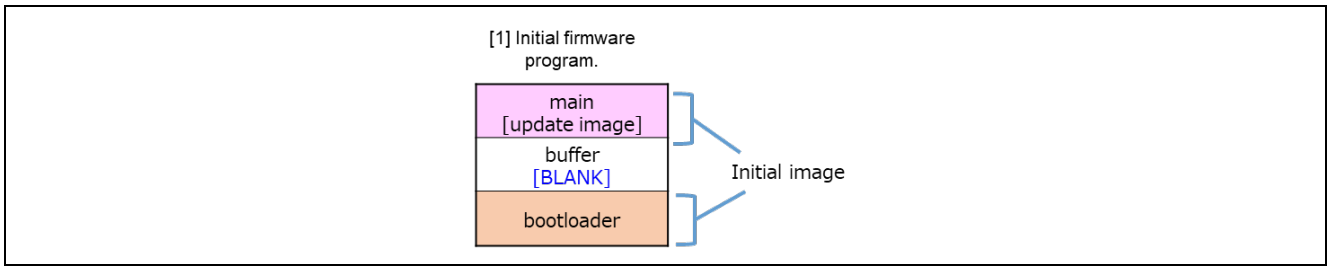


Figure 1.6 Initial Firmware Update Settings Utilizing Renesas Image Generator (Example of Partial Update Method)

[1] Program the initial image

The initial image is programmed to the on-chip flash memory using a tool such as Flash Writer.

1.4.3 Initial State of Linear Mode Full Update Method Settings Utilizing Renesas Image Generator

The following figure shows the construction of the initial state of the full update method using the Renesas Image Generator.

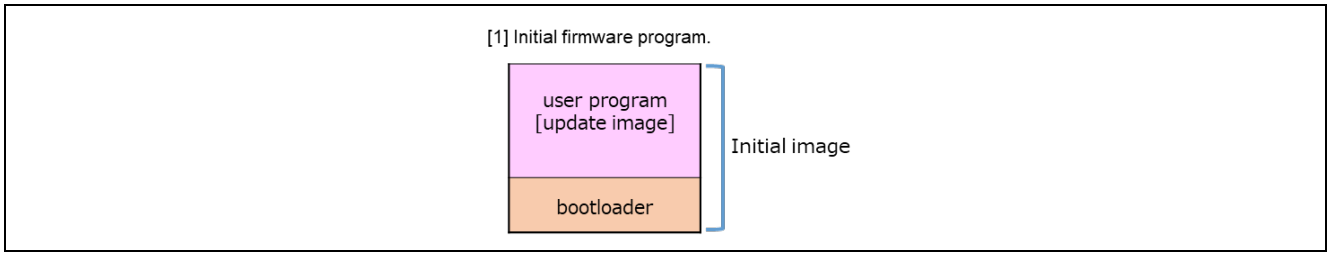


Figure 1.7 Initial Firmware Update Settings Utilizing Renesas Image Generator (Example of Full Update Method)

[1] Program the initial image
The initial image is programmed to the on-chip flash memory using a tool such as Flash Writer.

1.4.4 Initial State of Dual-Bank Method Settings Utilizing Bootloader

The following figure shows the construction of the initial state of the dual-bank method using the bootloader.

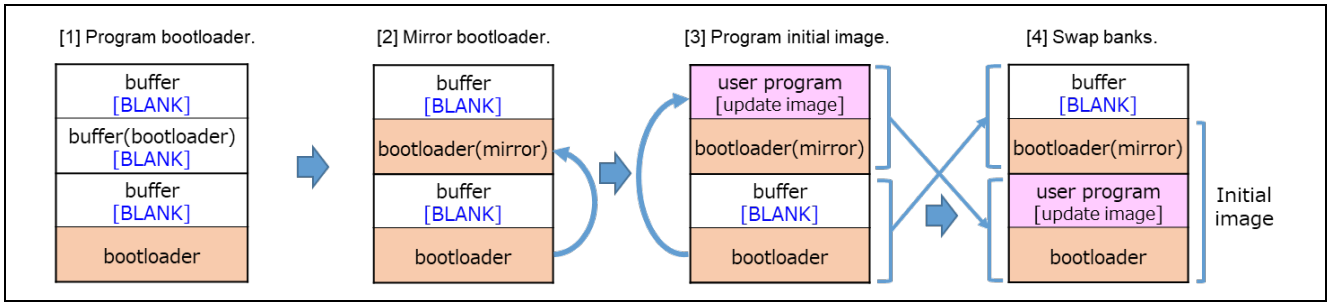


Figure 1.8 Initial Firmware Update Settings Utilizing Bootloader (Example of Dual-Bank Method)

[1] Program bootloader.
The bootloader is programmed to the on-chip flash memory using a tool such as Flash Writer.

[2] Mirror bootloader.
The bootloader is mirrored to bank 1 by the bootloader.

[3] Program initial image.
The initial image is downloaded from an external source and programmed to the buffer plane using the functionality of the bootloader. The programmed firmware is verified.

[4] Swap banks.
If verification is successful, the banks are swapped and processing ends.

1.4.5 Initial State of Linear Mode Partial Update Method Settings Utilizing Bootloader

The following figure shows the construction of the initial state of the partial update method using the bootloader.

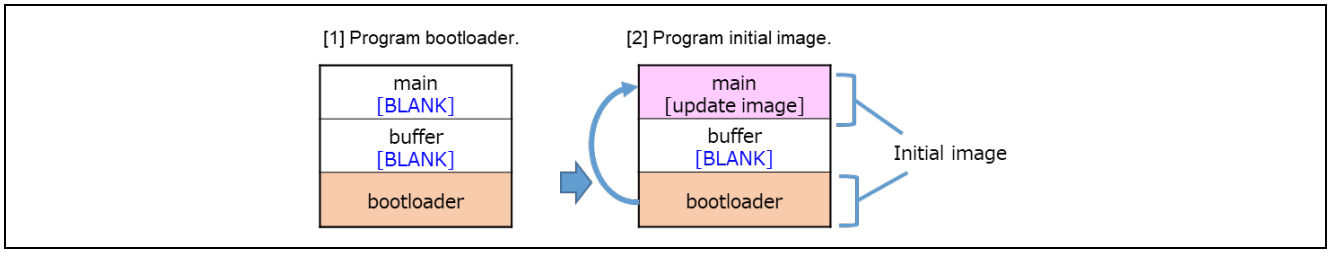


Figure 1.9 Initial Firmware Update Settings Utilizing Bootloader (Example of Partial Update Method)

[1] Program bootloader.
The bootloader is programmed to the on-chip flash memory using a tool such as Flash Writer.

[2] Program initial image.
The initial image is downloaded from an external source and programmed to the main plane using the functionality of the bootloader. The programmed firmware is verified, and if verification is successful, processing ends.

1.4.6 Initial State of Linear Mode Full Update Method Settings Utilizing Bootloader

The following figure shows the construction of the initial state of the full update method using the bootloader.

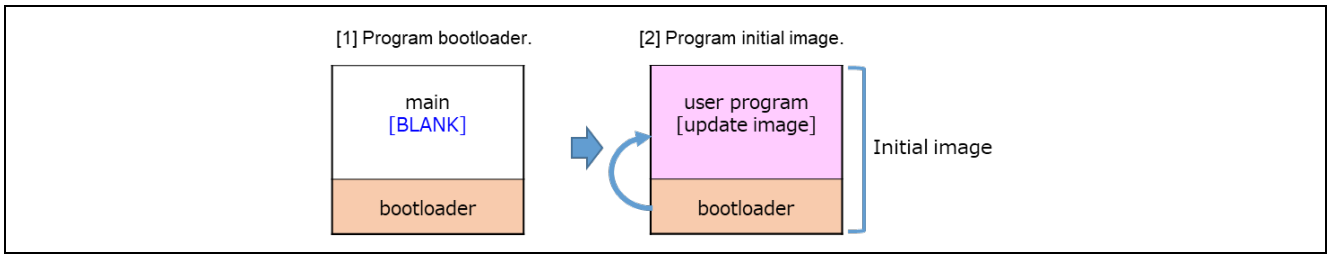


Figure 1.10 Initial Firmware Update Settings Utilizing Bootloader (Example of Full Update Method)

[1] Program bootloader.
The bootloader is programmed to the on-chip flash memory using a tool such as Flash Writer.

[2] Program initial image.
The initial image is downloaded from an external source and programmed to the main plane using the functionality of the bootloader. The programmed firmware is verified, and if verification is successful, processing ends.

1.5 Package Contents

The firmware update module package contains several files, including software and tools. These are listed in the table below.

Table 1.3 Folder Structure of Firmware Update Module Package

Folder Name	Description
r01an6850xx0203-rx-fwupdate.zip\	
├──FITDemos	Sample projects
│ └──rx	
│ ├──modules	Sample program
│ │ ├──boot_loader.c	boot_loader.c
│ │ ├──fwup_main.c	fwup_main.c
│ │ └──my_flash.c	my_flash.c
│ ├──rx*YYY*	YYY: Non-dual bank products (rx23ea-rssk/rx23eb-rssk/rx24t-rsk/rx66t-rsk/rx130-rsk/rx140-rask/rx231-rsk/rk660-rsk)
│ │ ├──w_buffer	Linear Mode Partial Update Method
│ │ │ ├──e2_ccrx	CC-RX version
│ │ │ │ ├──boot_loader	Bootloader
│ │ │ │ ├──fwup_leddemo	LED illumination application
│ │ │ │ └──fwup_main	User applications including firmware update
│ │ │ ├──e2_gcc	GCC version
│ │ │ │ ├──boot_loader	Bootloader
│ │ │ │ ├──fwup_leddemo	LED illumination application
│ │ │ │ └──fwup_main	User applications including firmware update
│ │ │ └──iar	IAR version
│ │ │ ├──boot_loader	Bootloader
│ │ │ ├──fwup_leddemo	LED illumination application
│ │ │ └──fwup_main	User applications including firmware update
│ │ └──wo_buffer	Linear Mode Full Update Method
│ │ ├──e2_ccrx	CC-RX version
│ │ │ ├──boot_loader	Bootloader
│ │ │ └──fwup_leddemo	User applications including firmware update
│ │ ├──e2_gcc	GCC version
│ │ │ ├──boot_loader	Bootloader
│ │ │ └──fwup_leddemo	User applications including firmware update
│ │ └──iar	IAR version
│ │ ├──boot_loader	Bootloader
│ │ └──fwup_leddemo	User applications including firmware update
│ └──rxZZZ	ZZZ: Dual Bank Products (rx65n-rsk/rx26t-mck/rx24t-rsk/rx72n-rsk/rx671-rsk)
│ ├──dualbank	Dual-Bank Method
│ │ ├──e2_ccrx	CC-RX version
│ │ │ ├──boot_loader	Bootloader
│ │ │ ├──fwup_leddemo	LED illumination application
│ │ │ └──fwup_main	User applications including firmware update
│ │ ├──e2_gcc	GCC version
│ │ │ ├──boot_loader	Bootloader
│ │ │ ├──fwup_leddemo	LED illumination application
│ │ │ └──fwup_main	User applications including firmware update

Folder Name	Description
│ │ └─iar	IAR version
│ │ ├─boot_loader	Bootloader
│ │ ├─fwup_leddemo	LED illumination application
│ │ └─fwup_main	User applications including firmware update
│ ├─w_buffer	Linear Mode Partial Update Method
│ │ ├─e2_ccrx	CC-RX version
│ │ │ ├─boot_loader	Bootloader
│ │ │ ├─fwup_leddemo	LED illumination application
│ │ │ └─fwup_main	User applications including firmware update
│ │ ├─e2_gcc	GCC version
│ │ │ ├─boot_loader	Bootloader
│ │ │ ├─fwup_leddemo	LED illumination application
│ │ │ └─fwup_main	User applications including firmware update
│ │ └─iar	IAR version
│ │ ├─boot_loader	Bootloader
│ │ ├─fwup_leddemo	LED illumination application
│ │ └─fwup_main	User applications including firmware update
│ └─wo_buffer	Linear Mode Full Update Method
│ ├─e2_ccrx	CC-RX version
│ │ ├─boot_loader	Bootloader
│ │ └─fwup_leddemo	User applications including firmware update
│ ├─e2_gcc	GCC version
│ │ ├─boot_loader	Bootloader
│ │ └─fwup_leddemo	User applications including firmware update
│ └─iar	IAR version
│ ├─boot_loader	Bootloader
│ └─fwup_leddemo	User applications including firmware update
├─FITModules	FIT module
│ ├─r_config	Configuration file
│ └─r_fwup	Firmware update module
│ ├─doc	Documentation
│ │ ├─en	English Application Notes
│ │ └─jp	Japanese Application Notes
│ ├─ref	Reference Files
│ └─src	Source code
│ ├─base64	Base64 decode
│ └─tinycrypt	Crypto library
│ ├─include	Include File
│ └─source	Source code
└─RenesasImageGenerator	Renesas Image Generator (Python program and parameter files)
├─image-gen.py	Python program for Renesas Image Generator
└─RX*xx_xxxx*_ImageGenerator_PRM.csv	Parameter file for demo project

1.6 API Overview

Table 1.3 lists the API functions included in the firmware update module.

Table 1.4 API Functions

Function	Function Description
R_FWUP_Open	Opens the module.
R_FWUP_Close	Performs processing to close the module.
R_FWUP_IsExistImage	Confirms the existence of an image in the specified area.
R_FWUP_EraseArea	Erases the specified area.
R_FWUP_GetImageSize	Obtains the size of the image.
R_FWUP_WriteImage	Writes the image (header portion + program portion).
R_FWUP_VerifyImage	Validates the image.
R_FWUP_ActivateImage	Activates a new image.
R_FWUP_ExecImage	Launches a new image.
R_FWUP_SoftwareReset	Applies a software reset.
R_FWUP_SoftwareDelay	Applies a software delay.
R_FWUP_GetVersion	Returns the version number of the module.
R_FWUP_WriteImageHeader	Writes the header portion of the image. (For Special Purpose)
R_FWUP_WriteImageProgram	Writes the program portion of the image. (For Special Purpose)

Note: Special purpose refers to the use of FWUP FIT Rev1.0x bootloader or OTA of FreeRTOS. If you are considering to use Rev2.0x firmware update module on bare metal without FreeRTOS, please skip this section.

2. API Information

2.1 Hardware Requirements

The MCU used must support the following functions:

- Flash memory

2.2 Software Requirements

The module is dependent upon the following drivers:

- Board support package (r_bsp)
- Flash module (r_flash)
- Serial communications interface (SCI: asynchronous/clock synchronous) (r_sci)
- Byte queue buffer module (r_byteq)

2.3 Supported Toolchains

The module has been confirmed to work with the toolchains listed in 6.1, Confirmed Operation Environments.

2.4 Header Files

All API calls and their supporting interface definitions are located in r_fwup_if.h.

2.5 Integer Types

The driver uses ANSI C99. These types are defined in stdint.h.

2.6 Compile Settings

The configuration option settings of the module are contained in r_fwup_config.h.

The names of the options and descriptions of their setting values are listed in Table 2.1.

Table 2.1 Configuration Settings

Configuration options in r_fwup _config.h	
FWUP_CFG_UPDATE_MODE	Update method 0: Dual-Bank Method 1: Linea Mode Partial Update Method 2: Linea Mode Full Update Method 3: Not available for RX
FWUP_CFG_FUNCTION_MODE	Specifies how the module is used. 0: Bootloader 1: Application program
FWUP_CFG_MAIN_AREA_ADDR_L	Specifies the start address of the main plane.
FWUP_CFG_BUF_AREA_ADDR_L	Specifies the start address of the buffer plane (in on-chip flash memory).
FWUP_CFG_AREA_SIZE	Specifies the size of the main plane and buffer plane.
FWUP_CFG_CF_BLK_SIZE	Specifies the block size of the on-chip code flash.
FWUP_CFG_CF_W_UNIT_SIZE	Specifies the writing unit for the on-chip code flash.
FWUP_CFG_EXT_BUF_AREA_ADDR_L	Specifies the start address of the buffer plane in external flash memory. (Not subject to change in RX)
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE	Specifies the block size or sector size of the external flash memory. (Not subject to change in RX)
FWUP_CFG_DF_ADDR_L	Start address of data flash.
FWUP_CFG_DF_BLK_SIZE	Block size of data flash.
FWUP_CFG_DF_NUM_BLKS	Block count of data flash. Specify 0 if there is no data flash.
FWUP_CFG_FWUPV1_COMPATIBLE	FWUP V1 Compatibility Setting (For Special Purpose) 0: Disable 1: Enable (For Special Purpose)
FWUP_CFG_SIGNATURE_VERIFICATION	Verification method 0: ECDSA + SHA256 1: SHA256
FWUP_CFG_PRINTF_DISABLE	Log display setting 0: Enable 1: Disable

Configuration options in r_fwup _config.h	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	User-defined interrupt disable function setting 0: FWUP function 1: User function
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	User-defined function name for disabling interrupts
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	User-defined interrupt enable function setting 0: FWUP function 1: User function
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	User-defined interrupt enable function name
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	User-defined software delay function setting 0: FWUP function 1: User function
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	User-defined software delay function name
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	User-defined software reset function setting 0: FWUP function 1: User function
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	User-defined software reset function name
FWUP_CFG_USER_SHA256_INIT_ENABLED	User-defined SHA256 Init function setting 0: FWUP function 1: User function
FWUP_CFG_USER_SHA256_INIT_FUNCTION	User-defined SHA256 Init function name
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	User-defined SHA256 Update function setting 0: FWUP function 1: User function
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	User-defined SHA256 Update function name
FWUP_CFG_USER_SHA256_FINAL_ENABLED	User-defined SHA256 Final function setting 0: FWUP function 1: User function
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	User-defined SHA256 Final function name
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	User-defined ECDSA Verify function setting 0: FWUP function 1: User function
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	User-defined ECDSA Verify function name
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	User-defined encryption context acquisition function setting 0: FWUP function 1: User function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	User-defined cryptographic operation context acquisition function name

Configuration options in r_fwup _config.h	
FWUP_CFG_USER_FLASH_OPEN_ENABLED	User-defined flash open function setting 0: FWUP function 1: User function
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	User-defined flash open function name
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	User-defined flash close function setting 0: FWUP function 1: User function
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	User-defined flash close function name
FWUP_CFG_USER_FLASH_ERASE_ENABLED	User-defined flash-erase function setting 0: FWUP function 1: User function
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	User-defined flash erase function name
FWUP_CFG_USER_FLASH_WRITE_ENABLED	User-defined flash write function setting 0: FWUP function 1: User function
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	User-defined flash write function name
FWUP_CFG_USER_FLASH_READ_ENABLED	User-defined flash read function setting 0: FWUP function 1: User function
FWUP_CFG_USER_FLASH_READ_FUNCTION	User-defined flash read function name
FWUP_CFG_USER_BANK_SWAP_ENABLED	User-defined bank swap function setting 0: FWUP function 1: User function
FWUP_CFG_USER_BANK_SWAP_FUNCTION	User-defined bank swap function name

2.7 Sample Project Code Sizes

The tables below show the ROM, RAM, and maximum stack sizes for the sample projects included in the package associated with this application note. The values in the table below have been confirmed under the following conditions:

Module revision: Firmware update module for RX, v2.0.3

Compiler version: Renesas Electronics C/C++ Compiler for RX Family V3.05.00
GCC for Renesas RX 8.3.0.202305
IAR C/C++ Compiler for Renesas RX 5.10.1

CC-RX

- Optimization level: Size and execution speed (-Odefault)
- Delete variables/functions that have never been referenced (optimize=symbol_delete)
- Generate reduced function I/O functions (Yes: maximum reduced version)

GCC

- Optimization level: Size (-Os)
- Use newlib-nano (--specs=nano.specs)

IAR

- Optimization level: High (balanced)

Table 2.2 ROM, RAM, and Maximum Stack Sizes for Sample Projects

ROM, RAM, and Stack Code Sizes					
Device	Category	Memory Used (byte)			Remarks
		CC-RX	GCC	IAR	
RX130	ROM	22280	23124	17266	boot_loader*1
		12801	12480	9927	fwup_leddemo
		21787	23044	19455	fwup_main
	RAM	5579	8212	5433	boot_loader
		7443	8572	4968	fwup_leddemo
		5851	8468	6729	fwup_main
	Stack	552	576	2568	boot_loader
		188	68	840	fwup_leddemo
		552	576	2556	fwup_main
RX140	ROM	21314	21380	18457	boot_loader*1
		11925	10812	9231	fwup_leddemo
		20777	21220	17898	fwup_main
	RAM	5502	8084	5104	boot_loader
		7443	8572	4968	fwup_leddemo
		5774	8340	6400	fwup_main
	Stack	552	576	2600	boot_loader
		188	68	868	fwup_leddemo
		552	576	2588	fwup_main
RX231	ROM	25048	21352	18852	boot_loader*1
		11784	10776	9127	fwup_leddemo
		20733	21288	18471	fwup_main
	RAM	5715	8340	5397	boot_loader
		7543	8700	5068	fwup_leddemo
		5962	8468	6693	fwup_main
	Stack	552	576	2656	boot_loader
		188	68	916	fwup_leddemo
		552	576	2644	fwup_main
RX23E-A	ROM	20926	21012	18512	boot_loader*1
		11382	10248	8619	fwup_leddemo
		20330	20776	17954	fwup_main
	RAM	5596	8212	5309	boot_loader
		7455	8572	4980	fwup_leddemo
		5868	8468	6605	fwup_main
	Stack	552	576	2656	boot_loader
		188	68	916	fwup_leddemo
		552	576	2644	fwup_main

ROM, RAM, and Stack Code Sizes					
Device	Category	Memory Used (byte)			Remarks
		CC-RX	GCC	IAR	
RX23E-B	ROM	20981	21052	18583	boot_loader*1
		11601	10528	8859	fwup_leddemo
		20540	21040	18203	fwup_main
	RAM	5620	8340	5333	boot_loader
		7479	8700	5004	fwup_leddemo
		5892	8468	6629	fwup_main
	Stack	552	576	2656	boot_loader
		188	68	916	fwup_leddemo
		552	576	2644	fwup_main
RX24T	ROM	21469	21720	18575	boot_loader*1
		11434	10376	8699	fwup_leddemo
		20844	20696	17966	fwup_main
	RAM	5118	7956	3801	boot_loader
		3411	5116	2472	fwup_leddemo
		5246	8084	3929	fwup_main
	Stack	552	576	2656	boot_loader
		188	68	916	fwup_leddemo
		552	576	2644	fwup_main
RX26T	ROM	24064	24020	22570	boot_loader*1
		13146	13028	11719	fwup_leddemo
		23650	23592	22125	fwup_main
	RAM	4077	5912	5615	boot_loader
		3285	5500	5417	fwup_leddemo
		4302	6552	5743	fwup_main
	Stack	552	576	2880	boot_loader
		188	68	1148	fwup_leddemo
		552	576	2868	fwup_main
RX65N	ROM	24389	24716	23034	boot_loader*2
		14392	14608	12700	fwup_leddemo
		24188	24984	22927	fwup_main
	RAM	5284	6552	4722	boot_loader
		8017	9212	5542	fwup_leddemo
		5556	6808	6018	fwup_main
	Stack	552	576	3008	boot_loader
		188	68	1268	fwup_leddemo
		552	576	2996	fwup_main

ROM, RAM, and Stack Code Sizes					
Device	Category	Memory Used (byte)			Remarks
		CC-RX	GCC	IAR	
RX66T	ROM	23526	23840	21659	boot_loader*2
		13261	13060	11267	fwup_leddemo
		22858	23560	20982	fwup_main
	RAM	6786	8600	6499	boot_loader
		7853	9084	5377	fwup_leddemo
		7186	8984	7923	fwup_main
	Stack	552	576	2820	boot_loader
		188	68	1088	fwup_leddemo
		552	576	2808	fwup_main
RX660	ROM	23870	24180	22324	boot_loader*2
		14052	13968	12095	fwup_leddemo
		23431	24212	21965	fwup_main
	RAM	6641	8472	6315	boot_loader
		7869	9084	5393	fwup_leddemo
		7041	8728	7739	fwup_main
	Stack	552	576	2816	boot_loader
		188	68	1084	fwup_leddemo
		552	576	2804	fwup_main
RX671	ROM	24977	25284	23738	boot_loader*2
		14781	15016	13212	fwup_leddemo
		24719	25544	23654	fwup_main
	RAM	5300	6552	4738	boot_loader
		8033	9212	5557	fwup_leddemo
		5572	6808	6034	fwup_main
	Stack	552	576	3008	boot_loader
		192	68	1266	fwup_leddemo
		552	576	2996	fwup_main
RX72N	ROM	25136	25612	23965	boot_loader*2
		15030	15416	13452	fwup_leddemo
		24946	25896	23878	fwup_main
	RAM	5404	6680	4842	boot_loader
		8137	9840	5661	fwup_leddemo
		5676	6936	6138	fwup_main
	Stack	552	576	3076	boot_loader
		192	68	1344	fwup_leddemo
		552	576	3064	fwup_main

*1) Allocate 32KB of space to place the bootloader.

*2) Allocate 64KB of space to place the bootloader.

2.8 Arguments

The return values of the API functions are shown below. This enumeration is located in r_fwup_if.h, as are the prototype declarations of the API functions.

```
typedef enum fwup_area
{
    FWUP_AREA_MAIN = 0,
    FWUP_AREA_BUFFER,
    FWUP_AREA_DATA_FLASH
} e_fwup_area_t;

typedef enum e_fwup_delay_units
{
    FWUP_DELAY_MICROSECS = 0,
    FWUP_DELAY_MILLISECS,
    FWUP_DELAY_SECS
} e_fwup_delay_units_t;
```

2.9 Return Values

The return values of the API functions are shown below. This enumeration is located in r_fwup_if.h, as are the prototype declarations of the API functions.

```
typedef enum fwup_err
{
    FWUP_SUCCESS = 0,               // Normally terminated.
    FWUP_PROGRESS,                  // Firmware update is in progress.
    FWUP_ERR_FLASH,                 // Detect error of flash module.
    FWUP_ERR_VERIFY,                // Verify error.
    FWUP_ERR_FAILURE,               // General error.
} e_fwup_err_t;
```

2.10 Adding the FIT Module to Your Project

The module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using the Smart Configurator in e2 studio

By using the Smart Configurator in e2 studio, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: e2 studio (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e2 studio

By using the FIT Configurator in e2 studio, the FIT module is automatically added to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the FIT Configurator in the IAR Embedded Workbench for Renesas RX environment

If you want to add a FIT module in the IAR Embedded Workbench for Renesas RX environment, use the RX Smart Configurator to add the FIT module to your project. Refer to “RX Smart Configurator User’s Guide:

2.11 “for”, “while” and “do while” statements

In this module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for the register to be reflected and so on. For this loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if the user incorporates fail-safe processing into loop processing, user can search the corresponding processing using “WAIT_LOOP”.

The following shows an example of description.

```
while statement example :
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
/* The delay period needed is to make sure that the PLL has stabilized. */
}
for statement example :
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
g_protect_counters[i] = 0;
}
do while statement example :
/* Reset completion waiting */
do
{
reg = phy_read(ether_channel, PHY_REG_CONTROL);
count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET));
/* WAIT_LOOP */
```

2.12 Implementation Examples of APIs

The following is an example implementation of a bootloader and application program for each firmware update method.
For details, please refer to the source code of the demo project included in this application note package.

2.12.1 Example of Bootloader Implementation of Dual-Bank Method

The following flowchart is an example of implementation of a bootloader in dual-bank method. The access part of the flash is accessed in blocking mode.

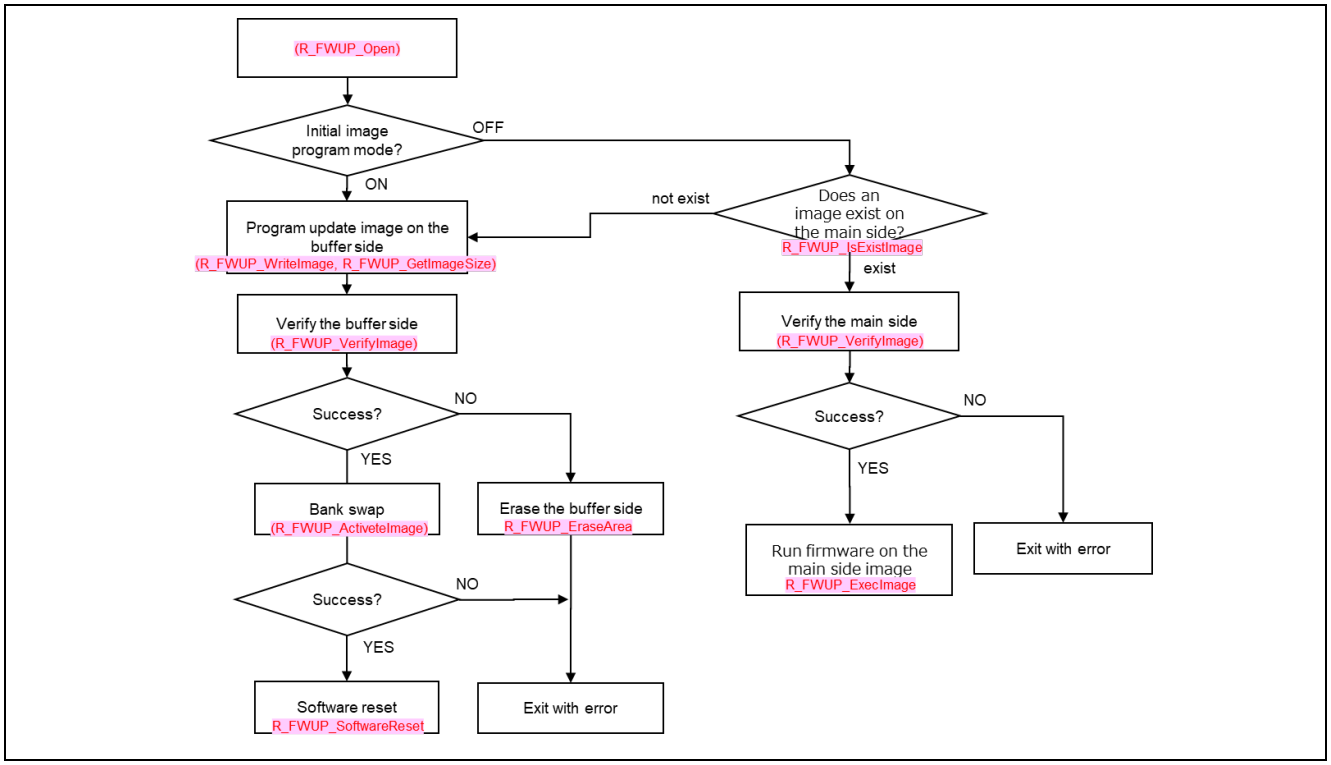


Figure 2.1 Example of Bootloader Implementation of Dual-Bank Method

2.12.2 Example of Application Program Implementation of Dual-Bank Method

The following flowchart is an example of implementation of an application program in dual-bank method. The access part of the flash is accessed in blocking mode.

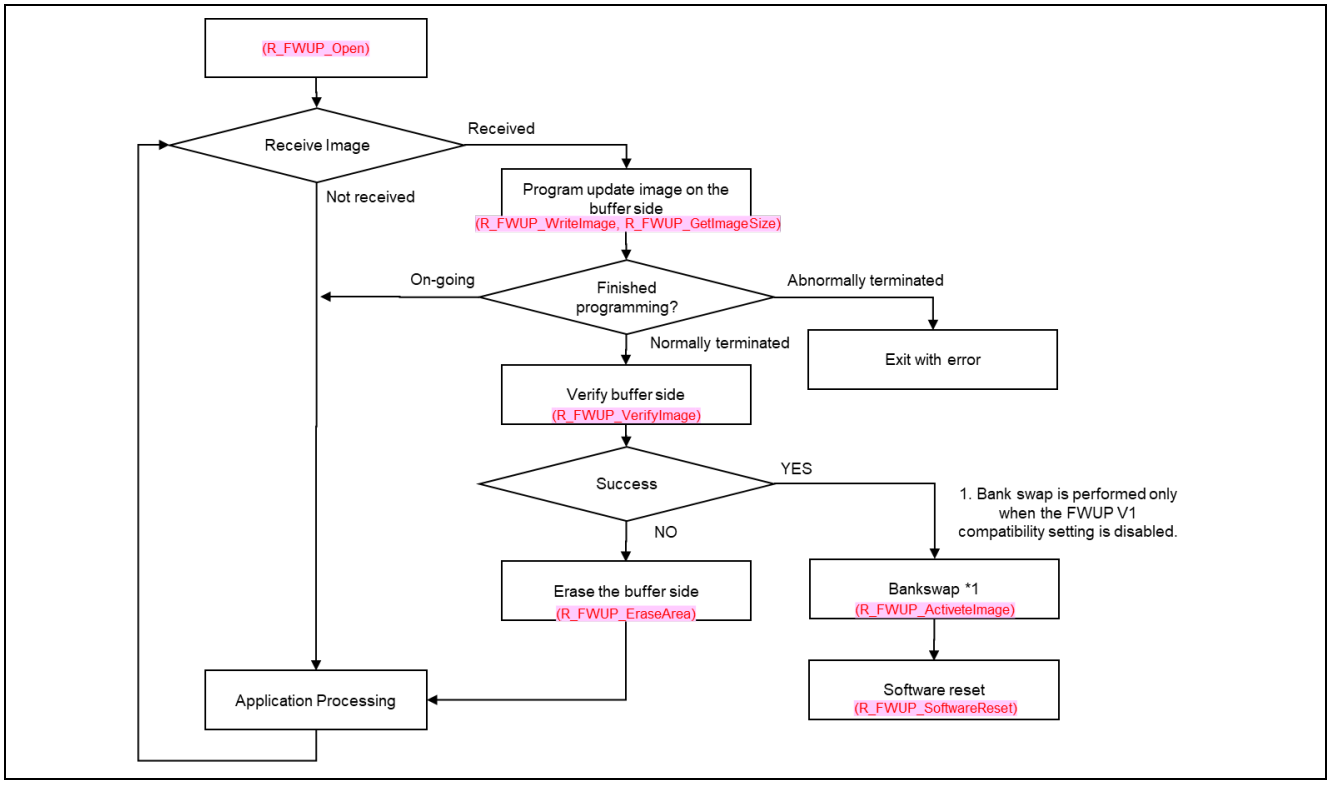


Figure 2.2 Example of Application Program Implementation of Dual-Bank Method

2.12.3 Example of Bootloader Implementation of Linear Mode Partial Update Method

The following flowchart is an example of implementation of a bootloader in linear mode partial update method.

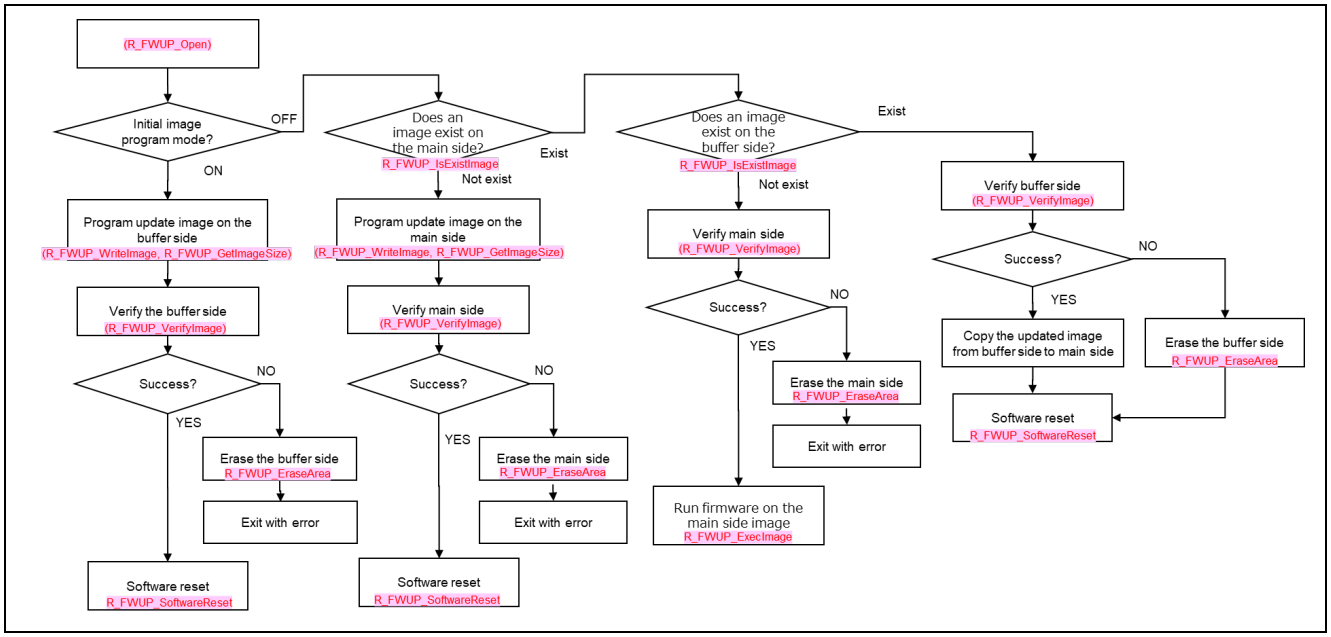


Figure 2.3 Example of Bootloader Implementation of Linear Mode Partial Update Method

2.12.4 Example of Application Program Implementation of Linear Mode Partial Update Method

The following flowchart is an example of implementation of an application program in linear mode partial update method.

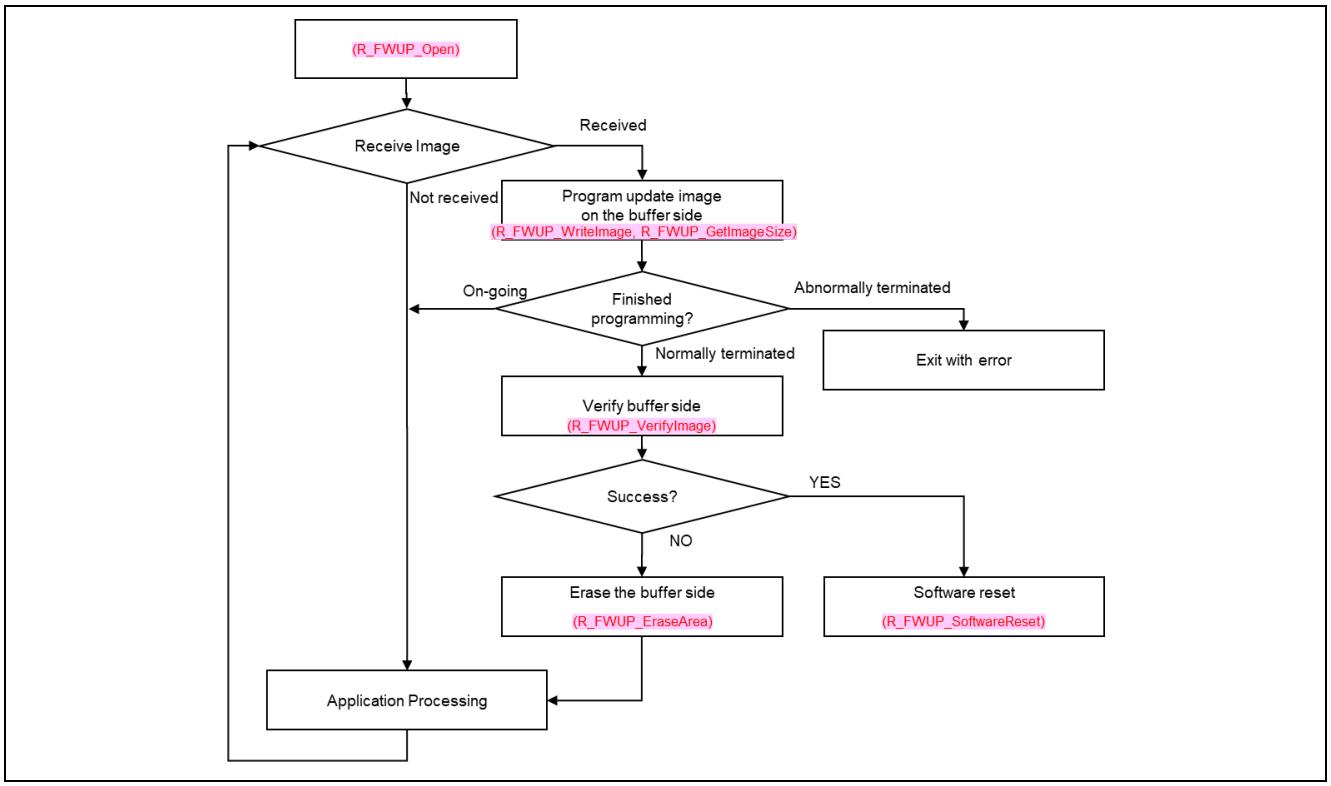


Figure 2.4 Example of application program implementation of linear mode partial update method

2.12.5 Example of Bootloader Implementation of Linear Mode Full Update Method

The following flowchart is an example of implementation of a bootloader in linear mode full update method.

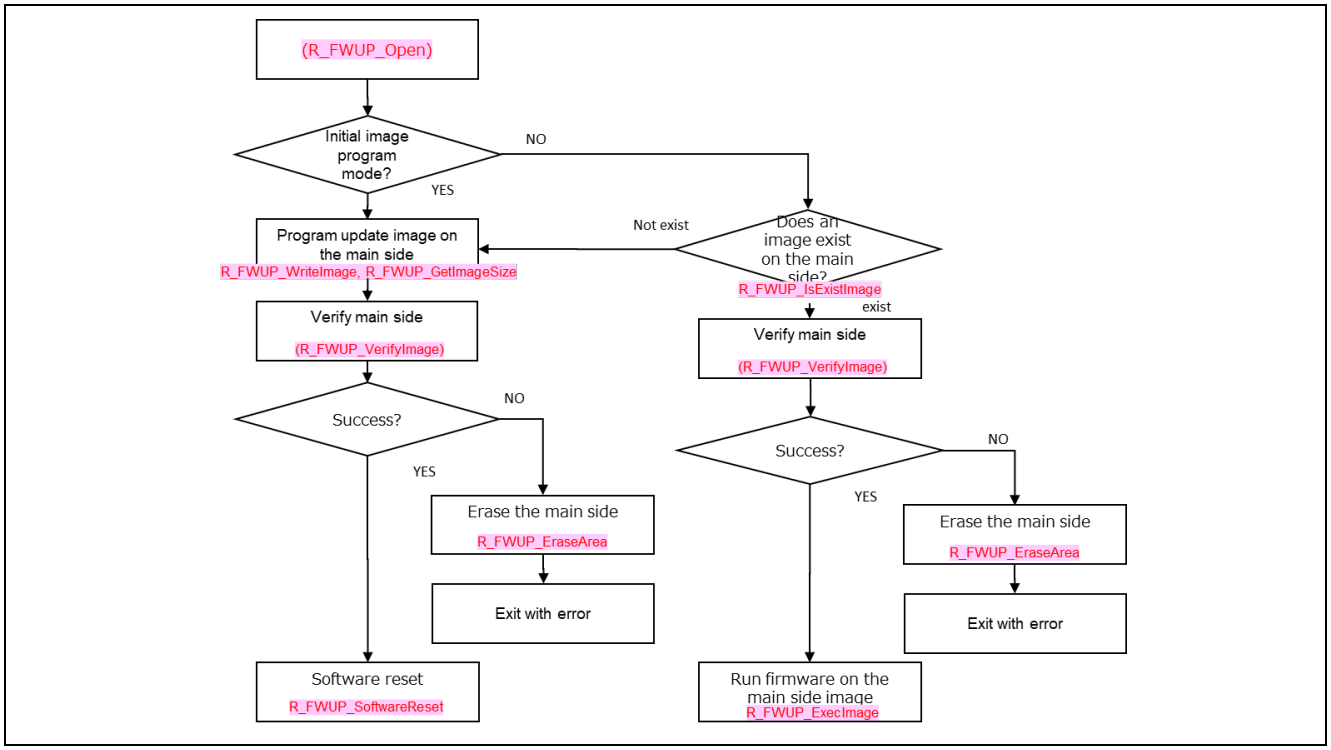


Figure 2.5 Example of Bootloader Implementation of Linear Mode Full Update Method

2.12.6 Example Implementation of API when Used in Non-blocking Mode

The flowchart below shows an example implementation when accessing flash in non-blocking mode. Non-blocking mode can only operate in a dual-mode environment.

Note that in non-blocking mode via this module, access to the flash module may occur multiple times in response to a firmware update module API call from the user, so the user cannot define a callback function for the flash module . Therefore, the wait time between the start of access to the flash module and the return of the callback function is provided as a process that can be implemented by the user.

In the dual bank method demonstration project, the wrapper function described in 3.15.3 is implemented in my_flash.c as a user-defined function. Figure 2.6 below shows the processing flow related to the non-blocking mode in my_flash.c.

To implement user processing while waiting for the callback function, change the user processing (R_BSP_NOP();) part described in red in the processing flow in Figure 2.6 to user processing..

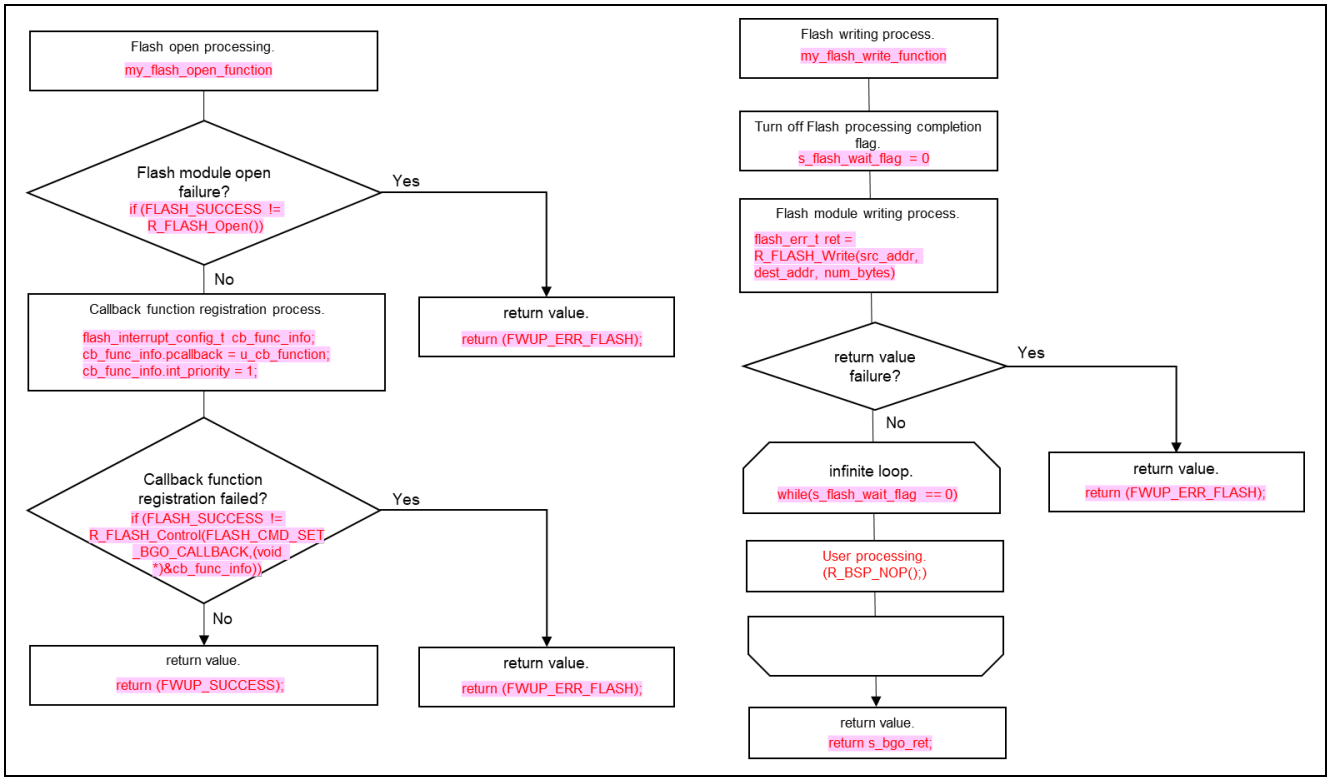


Figure 2.6 Processing flow in the case of non-blocking mode(1/2)

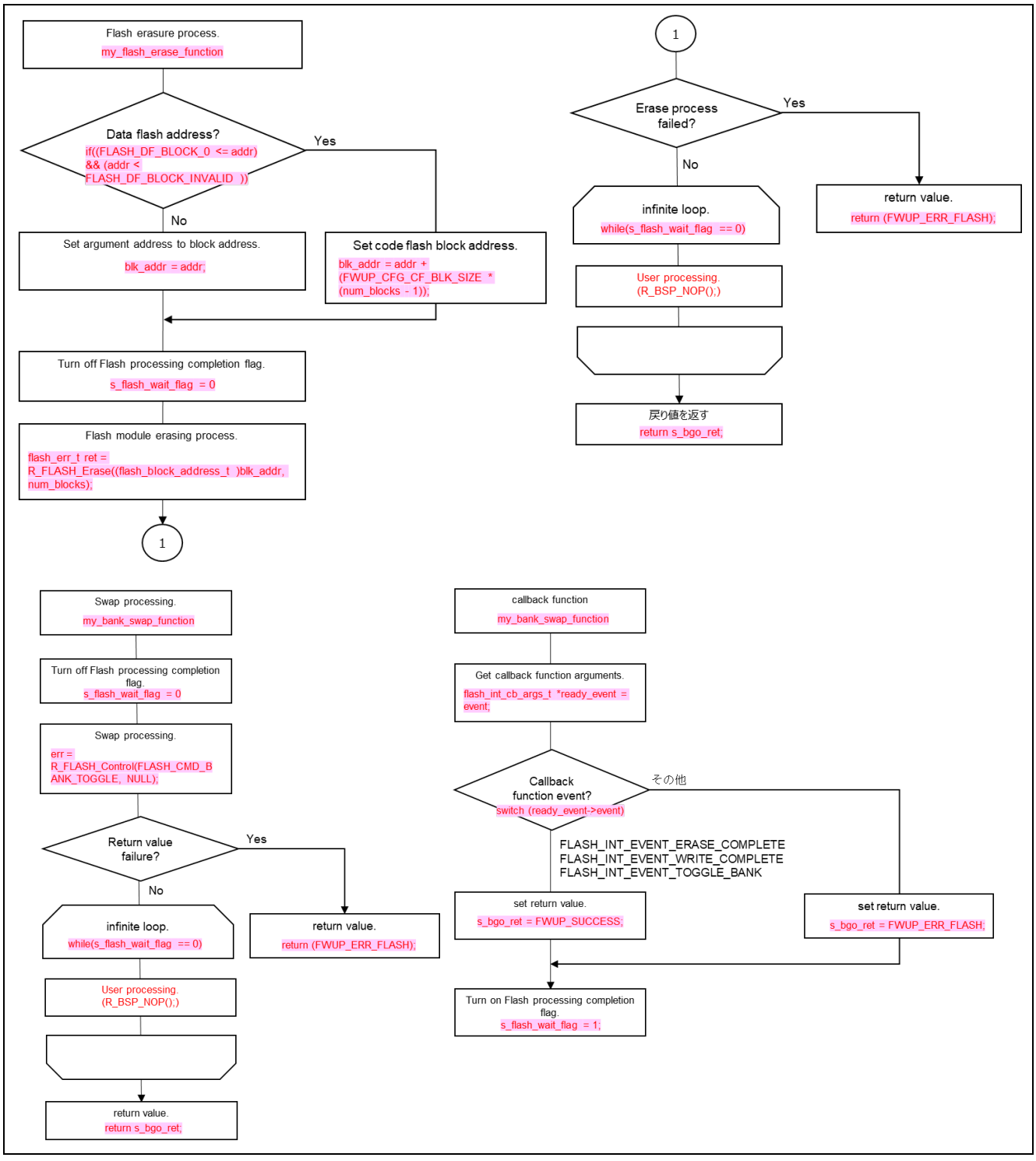


Figure 2.7 Processing flow in the case of non-blocking mode(2/2)

3. API Functions

3.1 R_FWUP_Open Function

Table 3.1 R_FWUP_Open Function Specifications

Format	e_fwup_err_t R_FWUP_Open (void)	
Description	Performs processing to open the firmware update module. Implements processing to open the flash module.	
Parameters	None	
Return Values	FWUP_SUCCESS	Normal end
	FWUP_ERR_FLASH	Flash module error
Special Notes	—	

3.2 R_FWUP_Close Function

Table 3.2 R_FWUP_Close Function Specifications

Format	void R_FWUP_Close (void)
Description	Performs processing to close the firmware update module. Implements processing to close the flash module.
Parameters	None
Return Values	None
Special Notes	—

3.3 R_FWUP_IsExistImage Function

Table 3.3 R_FWUP_IsExistImage Function Specifications

Format	bool R_FWUP_IsExistImage(e_fwup_area_t area)	
Description	Confirms the existence of an image in the specified area.	
Parameters	area: Main plane (FWUP_AREA_MAIN) or buffer plane (FWUP_AREA_BUFFER)	
Return Values	true	Image exists.
	false	Image does not exist.
Special Notes	Verify that the magic code is written correctly.	

3.4 R_FWUP_EraseArea Function

Table 3.4 R_FWUP_EraseArea Function Specificationss

Format	e_fwup_err_t R_FWUP_EraseArea(e_fwup_area_t area)	
Description	Erases the specified area.	
Parameters	area: Main plane (FWUP_AREA_MAIN) or buffer plane (FWUP_AREA_BUFFER), Data Flash (FWUP_AREA_DATA_FLASH)	
Return Values	FWUP_SUCCES	Normal end
	FWUP_ERR_FLASH	Flash module error
Special Notes	Erasure of the main plane can only be performed by the bootloader.	

3.5 R_FWUP_GetImageSize Function

Table 3.5 R_FWUP_GetImageSize Function Specificationss

Format	uint32_t R_FWUP_GetImageSize(void)	
Description	Returns the size of the image in bytes. This function obtains the byte size of the image based on the RSU header address information shown in Figure 4.1. Therefore, first write the RSU header address information to code flash using the R_FWUP_WriteImage function or the R_FWUP_WriteImageProgram function.	
Parameters	None	
Return Values	0	Acquisition in progress
	1 or more	Image size
Special Notes	—	

3.6 R_FWUP_WriteImage Function

Table 3.6 R_FWUP_WriteImage Function Specifications

Format	e_fwup_err_t R_FWUP_WriteImage(e_fwup_area_t area, uint8_t *p_buf, uint32_t buf_size)	
Description	Writes an image (header portion + program portion) to the specified area. Continue calling this function until the total size of the image is reached. The image size is obtained by R_FWUP_GetImageSize().	
Parameters	area: Main plane (FWUP_AREA_MAIN) or buffer plane (FWUP_AREA_BUFFER) p_buf: Image (header + program) buffer buf_size: Buffer size*1	
Return Values	FWUP_SUCCES	Writing of all images is completed.
	FWUP_PROGRESS	Writing of all images not completed (Writing of the specified number of images completed)
	FWUP_ERR_FLASH	Flash module error
	FWUP_ERR_FAILURE	Illegal parameter
Special Notes	1. Specify a multiple of the code flash write unit (for example, 64, 128, or 256). This size also applies to data flash. When FWUP_CFG_FWUPV1_COMPATIBLE is enabled, the magic code in the RSU header area used for processing is "Renesas" for FWUP V1.	

3.7 R_FWUP_VerifyImage Function

Table 3.7 R_FWUP_VerifyImage Function Specifications

Format	e_fwup_err_t R_FWUP_VerifyImage(e_fwup_area_t area)	
Description	Verifies an image using the cryptographic library embedded in the module.	
Parameters	area: Main plane (FWUP_AREA_MAIN) or buffer plane (FWUP_AREA_BUFFER)	
Return Values	FWUP_SUCCES	Verification successful
	FWUP_ERR_VERIFY	Verification failed
	FWUP_ERR_FAILURE	Illegal parameter
Special Notes	—	

3.8 R_FWUP_ActivateImage Function

Table 3.8 R_FWUP_ActivateImage Function Specifications

Format	e_fwup_err_t R_FWUP_ActivateImage(void)	
Description	Activates a new image. • Dual-bank update method — FWUP_CFG_FWUPV1_COMPATIBLE is enabled: Bank swap. — FWUP_CFG_FWUPV1_COMPATIBLE is disabled: Returns FWUP_SUCCESS without doing anything. • Linea mode partial update method — Bootloader: Copies the buffer plane image to the main plane. — User program: Returns FWUP_SUCCESS without doing anything. • Linea mode full update method — Returns FWUP_SUCCESS without doing anything.	
Parameters	None	
Return Values	FWUP_SUCCESS	Normal end
	FWUP_ERR_FLASH	Flash module error
Special Notes	—	

3.9 R_FWUP_ExecImage Function

Table 3.9 R_FWUP_ExecImage Function Specifications

Format	void R_FWUP_ExecImage(void)
Description	Runs the program in a valid image.
Parameters	None
Return Values	None
Special Notes	Set the interrupt allow flag of PSW to 0 to disable interrupts.

3.10 R_FWUP_SoftwareReset Function

Table 3.10 R_FWUP_SoftwareReset Function Specifications

Format	void R_FWUP_SoftwareReset(void)
Description	Execute software reset processing.
Parameters	None
Return Values	None
Special Notes	—

3.11 R_FWUP_SoftwareDelay Function

Table 3.11 R_FWUP_SoftwareDelay Function Specifications

Format	uint32_t R_FWUP_SoftwareDelay(uint32_t delay, e_fwup_delay_units_t units)	
Description	Execute software delay processing.	
Parameters	delay: Delay time units: Unit (µs, ms, or sec.)	
Return Values	0	Normal end
	Other	Abnormal end
Special Notes	—	

3.12 R_FWUP_GetVersion Function

Table 3.12 R_FWUP_GetVersion Function Specifications

Format	uint32_t R_FWUP_GetVersion(void)
Description	Returns the version number of the module.
Parameters	None
Return Values	Version number
Special Notes	—

3.13 R_FWUP_WriteImageHeader Function

This function is an API for special use where header information and program information must be written separately. Normally, use the R_FWUP_WriteImage function.

Table 3.13 R_FWUP_WriteImageHeader Function Specifications

Format	e_fwup_err_t R_FWUP_WriteImageHeader (e_fwup_area_t area, uint8_t FWUP_FAR *p_sig_type, uint8_t FWUP_FAR *p_sig, uint32_t sig_size)	
Description	Writes a signature that the bootloader uses for verification to the header of the image in the designated area.	
Parameters	area: Main plane (FWUP_AREA_MAIN) or buffer plane (FWUP_AREA_BUFFER) p_sig_type: Signature type character string “hash-sha256” or “sig-sha256-ecdsa” p_sig: Signature sig_size: Length of signature (Should be set to 64.)	
Return Values	FWUP_SUCCES	Write completed
	FWUP_ERR_FLASH	Flash module error
	FWUP_ERR_FAILURE	Illegal parameter
Special Notes	When FWUP_CFG_FWUPV1_COMPATIBLE is enabled, the magic code in the RSU header area used for processing is "Renesas" for FWUP V1.	

3.14 R_FWUP_WriteImageProgram Function

This function is an API for special use where header information and program information must be written separately. Normally, use the R_FWUP_WriteImage function.

Table 3.14 R_FWUP_WriteImageProgram Function Specifications

Format	e_fwup_err_t R_FWUP_WriteImageProgram (e_fwup_area_t area, uint8_t *p_buf, uint32_t offset, uint32_t buf_size)	
Description	Writes the program portion of the image to the specified area. Continue calling this function until the total size of the image is reached. The image size is obtained by R_FWUP_GetImageSize(). This function writes a program by offset based on the address information in the RSU header shown in Figure 4.1. Therefore, be sure to set 0x100 bytes of data from the offset (0x200) in Table 4.3 in the first call to this function. (Specify 0x200 for the offset argument and 0x100 or more for the buf_size argument.)	
Parameters	area: Main plane (FWUP_AREA_MAIN) or buffer plane (FWUP_AREA_BUFFER) p_buf: Buffer for program portion of image offset: Offset*1 buf_size: Buffer size*2	
Return Values	FWUP_SUCCES	Writing of all images is completed.
	FWUP_PROGRESS	Writing of all images not completed (Writing of the specified number of images completed)
	FWUP_ERR_FLASH	Flash module error
	FWUP_ERR_FAILURE	Illegal parameter
Special Notes	1. The offset must be 0x200 or greater. 2. Specify a multiple of the code flash write unit (for example, 64, 128, or 256). This size also applies to data flash.	

3.15 Wrapper Functions

The demo project provided in this package uses Tinycrypt for the cryptographic library and the FIT module for other wrapper functions. If you wish to use other libraries, applications, or drivers, please set the user-defined function setting for wrapper functions described in section 2.6 to 1 (User function) and implement the functions defined by the user.

3.15.1 Wrapper Functions (r_fwup_wrap_verify.c, h)

The demo project provided in this package uses Tinycrypt as the cryptographic library.

3.15.1.1 r_fwup_wrap_sha256_init Function

Table 3.15 r_fwup_wrap_sha256_init Function Specifications

Format	int32_t r_fwup_wrap_sha256_init (void *vp_ctx)	
Description	Start hash value calculation.	
Parameters	vp_ctx: Pointer to the context of the cryptographic library	
Return Values	0	Normal end
	Other	Abnormal end
Special Notes	—	

3.15.1.2 r_fwup_wrap_sha256_update Function

Table 3.16 r_fwup_wrap_sha256_update Function Specifications

Format	int32_t r_fwup_wrap_sha256_update (void *vp_ctx, const uint8_t *p_data, uint32_t datalen)	
Description	Calculates hash values for the specified range.	
Parameters	vp_ctx: Pointer to the context of the cryptographic library p_data: Starting address datalen: Data length (bytes)	
Return Values	0	Normal end
	Other	Abnormal end
Special Notes	—	

3.15.1.3 r_fwup_wrap_sha256_final Function

Table 3.17 r_fwup_wrap_sha256_final Function Specifications

Format	int32_t r_fwup_wrap_sha256_final (uint8_t *p_hash, void *vp_ctx)	
Description	Finishes computing the hash value and returns the hash value.	
Parameters	P_hash: Pointer to buffer to store the calculated hash value vp_ctx: Pointer to the context of the cryptographic library	
Return Values	0	Normal end
	Other	Abnormal end
Special Notes	—	

3.15.1.4 r_fwup_wrap_verify_ecdsa Function

Table 3.18 r_fwup_wrap_verify_ecdsa Function Specifications

Format	int32_t r_fwup_wrap_verify_ecdsa (uint8_t *p_hash, uint8_t *p_sig_type, uint8_t *p_sig, uint32_t sig_size)	
Description	ECDSA performs the verification.	
Parameters	p_hash: Pointer to the buffer where the hash value is stored p_sig_type: Signature type p_sig: Signature sig_size: Signature size	
Return Values	0	Normal end
	Other	Abnormal end
Special Notes	—	

3.15.1.5 r_fwup_wrap_get_crypt_context Function

Table 3.19 r_fwup_wrap_get_crypt_context Function Specifications

Format	void * r_fwup_wrap_get_crypt_context (void);	
Description	Returns a pointer to the context of the cryptographic library.	
Parameters	None	
Return Values	Void *	Pointer to the context of the cryptographic library
Special Notes	—	

3.15.2 Wrapper Functions (r_fwup_wrap_com.c, h)

The demo project provided in this package uses the FIT module.

3.15.2.1 r_fwup_wrap_disable_interrupt Function

Table 3.20 r_fwup_wrap_disable_interrupt Function Specifications

Format	void r_fwup_wrap_disable_interrupt (void)
Description	Disable Interrupt
Parameters	None
Return Values	None
Special Notes	This function calls the R_BSP_InterruptsDisable function.

3.15.2.2 r_fwup_wrap_enable_interrupt Function

Table 3.21 r_fwup_wrap_enable_interrupt Function Specifications

Format	void r_fwup_wrap_enable_interrupt (void)
Description	Enable Interrupt
Parameters	None
Return Values	None
Special Notes	This function calls the R_BSP_InterruptsEnable function.

3.15.2.3 r_fwup_wrap_software_delay Function

Table 3.22 r_fwup_wrap_software_delay Function Specifications

Format	uint32_t r_fwup_wrap_software_delay (uint32_t delay, e_fwup_delay_units_t units)	
Description	Software delay	
Parameters	delay：Delay time uinits: unit (us,ms,sec)	
Return Values	0	: normal end
	Other	: Abnormal end
Special Notes	This function calls the R_BSP_SoftwareDelay function.	

3.15.2.4 r_fwup_wrap_software_reset Function

Table 3.23 r_fwup_wrap_software_reset Function Specifications

Format	void r_fwup_wrap_software_reset (void)	
Description	Software reset	
Parameters	None	
Return Values	FWUP_SUCCES	: Normal end
	FWUP_ERR_FLASH	: Flash module error
Special Notes	This function calls the R_BSP_SoftwareReset function.	

3.15.3 Wrapper Functions (r_fwup_wrap_flash.c, h)

The demo project provided in this package uses the FIT module. Flash memory access is in blocking mode. If you want to run the dual bank method demo in non-blocking mode, please refer to 4.4.1.2

3.15.3.1 r_fwup_wrap_flash_open Function

Table 3.24 r_fwup_wrap_flash_open Function Specifications

Format	e_fwup_err_t r_fwup_wrap_flash_open (void)	
Description	Open the internal flash.	
Parameters	None	
Return Values	FWUP_SUCCES	: Normal end
	FWUP_ERR_FLASH	: Flash module error
Special Notes	This function calls the R_FLASH_Open function.	

3.15.3.2 r_fwup_wrap_flash_close Function

Table 3.25 r_fwup_wrap_flash_close Function Specifications

Format	void r_fwup_wrap_flash_close (void)
Description	Close the internal flash.
Parameters	None
Return Values	None
Special Notes	This function calls the R_FLASH_Close function.

3.15.3.3 r_fwup_wrap_flash_erase Function

Table 3.26 r_fwup_wrap_flash_erase Function Specifications

Format	e_fwup_err_t r_fwup_wrap_flash_erase (uint32_t addr, uint32_t num_blocks)	
Description	Erase the internal flash in block units.	
Parameters	addr : erase address num_blocks : erase block	
Return Values	FWUP_SUCCES	: Normal end
	FWUP_ERR_FLASH	: Flash module error
Special Notes	This function calls the R_FLASH_Erase function.	

3.15.3.4 r_fwup_wrap_flash_write Function

Table 3.27 r_fwup_wrap_flash_write Function Specifications

Format	e_fwup_err_t r_fwup_wrap_flash_write(uint32_t src_addr, uint32_t dest_addr, uint32_t num_bytes)	
Description	Erase the internal flash in block units.	
Parameters	src_addr: Pointer to write data dest_addr: Write address num_bytes: Write size (bytes)	
Return Values	FWUP_SUCCES	: Normal end
	FWUP_ERR_FLASH	: Flash module error
Special Notes	This function calls the R_FLASH_Write function.	

3.15.3.5 r_fwup_wrap_flash_read Function

Table 3.28 r_fwup_wrap_flash_read Function Specifications

Format	e_fwup_err_t r_fwup_wrap_flash_read (uint32_t buf_addr, uint32_t src_add, uint32_t size)	
Description	Reads the internal flash.	
Parameters	buf_addr: Address of the buffer to store the read data src_addr: Read address size: Read size	
Return Values	FWUP_SUCCES	: Normal end
	FWUP_ERR_FLASH	: Flash module error
Special Notes	Data is read out by the memcpy function.	

3.15.3.6 r_fwup_wrap_bank_swap Function

Table 3.29 r_fwup_wrap_bank_swap Function Specifications

Format	e_fwup_err_t r_fwup_wrap_bank_swap (void)	
Description	Execute a bank swap.	
Parameters	None	
Return Values	FWUP_SUCCES	: Normal end
	FWUP_ERR_FLASH	: Flash module error
Special Notes	This function calls the R_FLASH_Control function. Only models with dual banks.	

4. Demo Project

The demo project is a sample program that shows how to implement firmware update functionality using the serial communications interface (SCI).

4.1 Demo project Structure

The demo project comprises the FIT module, modules dependent on it, and a main() function that implements the firmware update demonstration. Versions of the demo project for the devices and compilers listed in 1.5 are provided.

The firmware update demo consists of the following projects.

Dual-bank method folder structure：Under □□\dualbank\△△\

Linear mode half-surface update method folder structure：Under □□\ w_buffer \△△\

Linear mode full update method folder structure: Under □□\ wo_buffer \△△\

□□: Device name

△△: Compiler (ccrx/gcc/iar)

- boot_loader: Bootloader
  This program runs first after a reset. It verifies that the user program has not been tampered with and then, if verification is successful, launches the user program.
- fwup_main: Application program
  An application program (initial firmware) that downloads updated firmware and performs signature verification.
- fwup_leddemo: Application program (for update)
  This is an application program (for updating) that blinks an LED.

4.2 Operating environment preparation

To run the firmware update demo project, you need to install the tools (see 4.2.1 to 4.2.4) on your Windows PC. Also, use a USB serial conversion board (see 4.2.5) that connects the Windows PC and the target board.

4.2.1 Installing TeraTerm

Used to transfer the firmware update image via serial communication from a Windows PC to the target board. In the demo project, we have checked the operation with TeraTerm 4.105.

After installation, set the serial port communication settings as shown in Table Table 4.1

Table 4.1 Communication Specifications

Item	Description
Communication system	Asynchronous communication
Bit rate	115,200 bps
Data length	8 bits
Parity	None
Stop bit	1 bit
Flow control	CTS/RTS

4.2.2 Installing the Python execution environment

Used by Renesas Image Generator (image-gen.py) to create initial and update images.

Renesas Image Generator uses ECDSA to generate signature data. In the demo project, environment operation is confirmed with Python 3.9.0.

Install Python 3.9.0 or higher.

In addition, since the Python encryption library (pycryptodome) is used, after installing Python, execute the following pip command from the command prompt to install the library.

```
pip install pycryptodome
```

4.2.3 Installing the OpenSSL execution environment

OpenSSL is used to generate the keys needed to generate and verify ECDSA signature data for initial and update images.

Download the OpenSSL installer from the following URL and install it. There is no problem with the Light version.

https://slproweb.com/products/Win32OpenSSL.html

4.2.4 Installing the Flash Writer

A flash writer is required to write the initial image.

The demo project uses Renesas Flash Programmer v3.11.01.

Renesas Flash Programmer (Programming GUI) | Renesas

4.2.5 USB serial conversion board

Used to transfer the firmware update image via serial communication from a Windows PC to the target board.

For details on how to connect with the target board, refer to the operation confirmation environment (6.2) of the relevant target board.

Use Pmod USBUART (manufactured by DIGILENT).

https://reference.digilentinc.com/reference/pmod/pmodusbuart/start

4.3 Execution environment preparation

4.3.1 Generating Keys for Signature Generation and Verification

Use OpenSSL for key generation. Refer to 4.2.3 in advance and install OpenSSL.

Execute the following OpenSSL commands to generate an elliptic curve cryptography (secp256r1) key pair to be used to generate and verify image signatures, and to extract the private and public keys:

```
>openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
using curve name prime256v1 instead of secp256r1

>openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey
read EC key
writing EC key

> openssl ec -in secp256r1.keypair -outform PEM -pubout -out
secp256r1.publickey
read EC key
writing EC key
```

4.3.2 Preparing the execution environment for Renesas Image Generator

Unzip ImageGenerator.zip included in the package to any folder on your Windows PC. Make sure the folder name does not contain double-byte characters.

Renesas Image Generator requires a Python execution environment, so refer to 4.2.2 and install Python in advance.

4.4 Demo Project Execution Procedure

The execution procedure of the demo project differs depending on the firmware update method.

This chapter describes the procedure for executing the demo project using the RX65N (2MB) as an example.

The demo project execution procedure is the same for other MCU products, but only the operation check environment differs for each MCU, so check the operation check environment (6.2) for the applicable MCU product.

The execution procedure of this demo project does not assume a debugger connection. Refer to 6.3 for information on how to debug the application through a debugger connection.

4.4.1 Dual Bank Method

4.4.1.1 Execution Environment

Prepare the RX65N operation check environment (6.2.8). For MCU products other than RX65N, please refer to the operation confirmation environment of the applicable MCU product.

4.4.1.2 Building The Demo Project

Follow the steps below to build three demo projects for the dual-bank method in linear mode.

The flash memory access in this demo project operates in blocking mode. If you want to operate in non-blocking mode, you can set the flash module configuration (FLASH_CFG_CODE_FLASH_BGO and FLASH_CFG_DATA_FLASH_BGO1) to 1 to operate in a non-blocking mode environment (FLASH_CFG_DATA_FLASH_BGO1). (See section 2.12.6 for details on operating in non-blocking mode with the firmware update module.)

The following procedure is described for the e2 studio environment; when using the IAR environment, please read and follow the procedure for IAR's Integrated Development Environment.

1. Import the boot_loader, fwup_leddemo, and fwup_main demo projects into the integrated development environment.
2. To operate in non-blocking mode, change the configuration settings of the flash module (r_flash).
   FLASH_CFG_CODE_FLASH_BGO : 1
   FLASH_CFG_DATA_FLASH_BGO : 1
3. Add the public key to be used to verify the image to the demo project.
   Paste the contents of secp256r1.publickey generated as described in 4.3.1 into code_signer_public_key.h in the boot_loader and fwup_main projects.

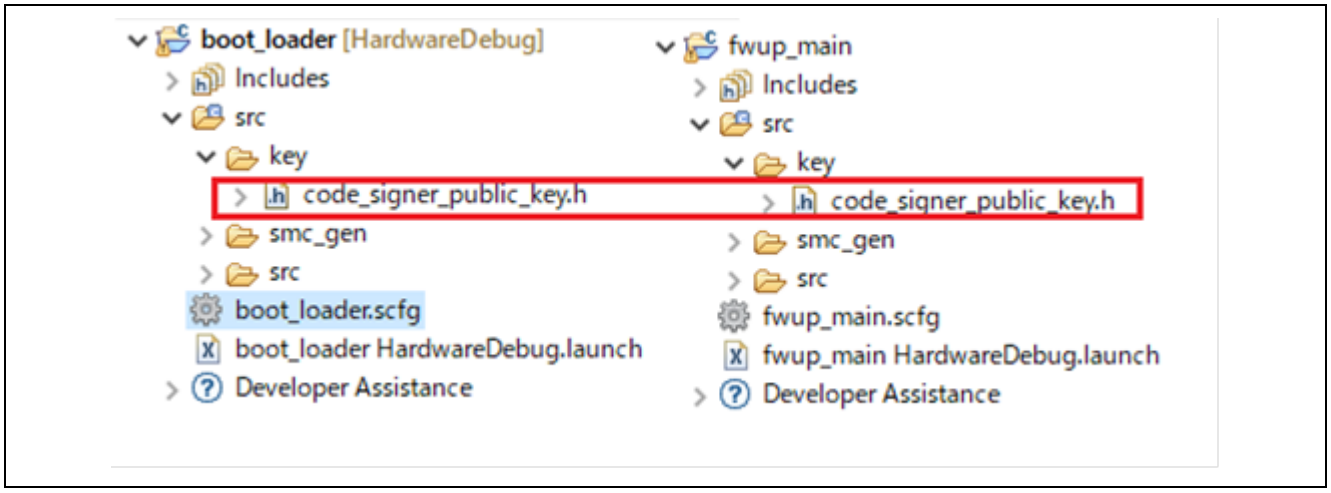


Figure 4.1 Storage Location of code_signer_public_key.h File in Demo Project

```
/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----"\
 * "...base64 data..."\
 * "-----END CERTIFICATE-----"
 */
#define CODE_SIGNER_PUBLIC_KEY_PEM \
"-----BEGIN PUBLIC KEY-----"\
Paste the contents of secp256r1.publickey here.
"-----END PUBLIC KEY-----"
#endif /* CODE_SIGNER_PUBLIC_KEY_H_ */
```

4. Build the demo projects.
   Build the three demo projects and confirm that the following mot files have been generated: boot_loader.mot, fwup_leddemo.mot, and fwup_main.mot.

4.4.1.3 Creating Initial and Update Images

The procedure for creating the initial and update images, using initial_firm.mot as the name of the initial image and fwup_leddemo.rsu as the name of the update image is, is described below.

1. Store the mot files created by building the demo projects in the same folder as Renesas Image Generator.

```
image-gen.py
RX65N_DualBank_ImageGenerator_PRM.csv
RX65N_Linear_Full_ImageGenerator_PRM.csv
RX65N_Linear_Half_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
```

2. Execute the following command to create the initial image:

```
> python image-gen.py -iup fwup_main.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o initial_firm -ibp
boot_loader.mot -vt ecdsa -key secp256r1.privatekey

Successfully generated the initial_firm.mot file.
```

3. Execute the following command to create the update image:

```
> python image-gen.py -iup fwup_leddemo.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o fwup_leddemo -vt ecdsa -key
secp256r1.privatekey

Successfully generated the fwup_leddemo.rsu file.
```

Confirm that the initial and update image have been created in the same folder as Renesas Image Generator.

```
image-gen.py
RX65N_DualBank_ImageGenerator_PRM.csv
RX65N_Linear_Full_ImageGenerator_PRM.csv
RX65N_ Linear_Half_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo.rsu
initial_firm.mot
```

4.4.1.4 Programming the Initial Image

Use Flash Writer to program the initial image (initial_firm.mot) to the MCU board. After programming, turn off the power to the board and disconnect the debugger (E2 Lite).

When writing with Renesas Flash Programmer, dual mode and linear mode are recognized as different microcontrollers, so an error will occur if the project is connected with a different bank mode from that of the microcontroller. For details on countermeasures, refer to chapter 2.1 of the following URL.

URL： https://www.renesas.com/jp/ja/document/apn/dual-mode-usage-guide?language=en&r=25424951

4.4.1.5 Executing a Firmware Update

Once the initial image firmware is activated, it waits for the transfer of the update image via the terminal emulator. The received update image is programmed to the flash memory, and after the transfer completes, the signature of the update image is verified and the firmware is activated.

Follow the steps below to execute a firmware update.

1. Refer to 6.2.8, Execution Environment, and connect the devices.
2. Launch the terminal emulator software on the PC, select the serial COM port, and configure the connection settings.
3. Power on the board. The following message is output:

```
==== RX65N : BootLoader [dual bank] ====
verify install area main [sig-sha256-ecdsa]...OK
execute new image ...

==== RX65N : Update from User [dual bank] ver 1.0.0 ====
send user program (*.rsu) via UART.
```

4. Send the updated image via the terminal emulator.
   Send file... > check Binary > fwup_leddemo.rsu
   The following message is output during the transfer of the update image, and a software reset is applied after installation and signature verification complete.

```
W 0xFFE00000, 256 ... OK
W 0xFFE00100, 256 ... OK
...
W 0xFFE03B00, 128 ... OK
W 0xFFEEFF80, 128 ... OK
```

5. After installing the updated firmware and verifying the signature, it jumps to the updated firmware and executes the program after processing such as bank swap.

```
verify install area buffer [sig-sha256-ecdsa]...OK
bank swap ...
software reset...
```

6. When the bootloader completes signature verification, the update image firmware is activated. When the process completes successfully, the following message is output and the LED flashes.

```
==== RX65N : BootLoader [dual bank] ====
verify install area main [sig-sha256-ecdsa]...OK
execute image ...

---------------------------------------
FWUP demo (ver 1.0.0)
---------------------------------------
Check the LEDs on the board.
```

Note: The demo program does not erase the buffer side. If you need to erase the image before updating for rollback measures, please add a process to erase the buffer side image.

4.4.2 Operation of Linear Mode Partial Update Method

4.4.2.1 Execution Environment

Prepare the RX65N operation check environment (6.2.8). For MCU products other than RX65N, please refer to the operation confirmation environment of the applicable MCU product.

4.4.2.2 Building The Demo Project

Follow the steps below to build three demo projects for the partial update method in linear mode.

The following procedure is described for the e2 studio environment; when using the IAR environment, please read and follow the procedure for IAR's Integrated Development Environment.

1. Import the boot_loader, fwup_leddemo, and fwup_main demo projects into the integrated development environment.
2. Add the public key to be used to verify the image to the demo project.
   Paste the contents of secp256r1.publickey generated as described in 4.3.1 into code_signer_public_key.h in the boot_loader and fwup_main projects.

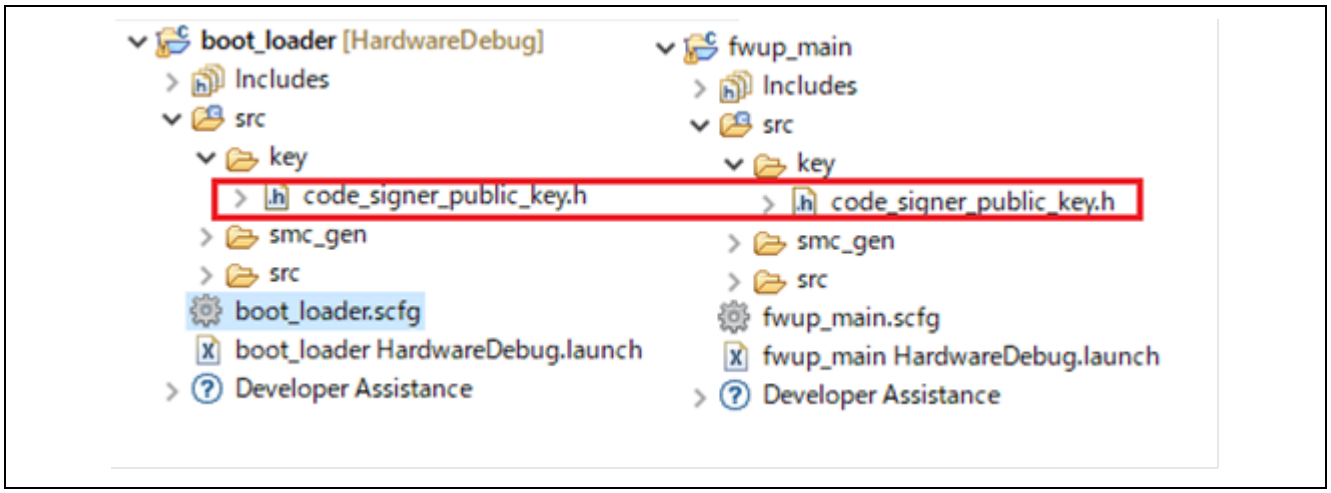


Figure 4.2 Storage Location of code_signer_public_key.h File in Demo Project

```
/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----"\
 * "...base64 data..."\
 * "-----END CERTIFICATE-----"
 */
#define CODE_SIGNER_PUBLIC_KEY_PEM \
"-----BEGIN PUBLIC KEY-----"\
Paste the contents of secp256r1.publickey here.
"-----END PUBLIC KEY-----"
#endif /* CODE_SIGNER_PUBLIC_KEY_H_ */
```

3. Build the demo projects.
   Build the three demo projects and confirm that the following mot files have been generated: boot_loader.mot, fwup_leddemo.mot, and fwup_main.mot.

4.4.2.3 Creating Initial and Update Images

The procedure for creating the initial and update images, using initial_firm.mot as the name of the initial image and fwup_leddemo.rsu as the name of the update image is, is described below.

1. Store the mot files created by building the demo projects in the same folder as Renesas Image Generator.

```
image-gen.py
RX65N_DualBank_ImageGenerator_PRM.csv
RX65N_Linear_Full_ImageGenerator_PRM.csv
RX65N_Linear_Half_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
```

2. Execute the following command to create the initial image:

```
> python image-gen.py -iup fwup_main.mot -ip
RX65N_Linear_Half_ImageGenerator_PRM.csv -o initial_firm -ibp
boot_loader.mot -vt ecdsa -key secp256r1.privatekey

Successfully generated the initial_firm.mot file.
```

3. Execute the following command to create the update image:

```
> python image-gen.py -iup fwup_leddemo.mot -ip
RX65N_Linear_Half_ImageGenerator_PRM.csv -o fwup_leddemo -vt ecdsa
-key secp256r1.privatekey

Successfully generated the fwup_leddemo.rsu file.
```

   Confirm that the initial and update image have been created in the same folder as Renesas Image Generator.

```
image-gen.py
RX65N_DualBank_ImageGenerator_PRM.csv
RX65N_Linear_Full_ImageGenerator_PRM.csv
RX65N_ Linear_Half_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo.rsu
initial_firm.mot
```

4.4.2.4 Programming the Initial Image

Use Flash Writer to program the initial image (initial_firm.mot) to the MCU board. After programming, turn off the power to the board and disconnect the debugger (E2 Lite).

When writing with Renesas Flash Programmer, dual mode and linear mode are recognized as different microcontrollers, so an error will occur if the project is connected with a different bank mode from that of the microcontroller. For details on countermeasures, refer to chapter 2.1 of the following URL.

URL： https://www.renesas.com/jp/ja/document/apn/dual-mode-usage-guide?language=en&r=25424951

4.4.2.5 Executing a Firmware Update

Once the initial image firmware is activated, it waits for the transfer of the update image via the terminal emulator. The received update image is programmed to the flash memory, and after the transfer completes, the signature of the update image is verified and the firmware is activated.

Follow the steps below to execute a firmware update.

1. Refer to 6.2.8, Execution Environment, and connect the devices.
2. Launch the terminal emulator software on the PC, select the serial COM port, and configure the connection settings.
3. Power on the board. The following message is output:

```
==== RX65N : BootLoader [with buffer] ====
verify install area main [sig-sha256-ecdsa]...OK
execute image ...

==== RX65N : Update from User [with buffer]  ver 1.0.0 ====
send image(*.rsu) via UART.
```

4. Send the updated image via the terminal emulator.
   Send file… > check Binary > fwup_leddemo.rsu
   The following message is output during the transfer of the update image, and a software reset is applied after installation and signature verification complete.

```
W 0xFFF00000, 256 ... OK
W 0xFFF00100, 256 ... OK
...
W 0xFFF03B00, 128 ... OK
W 0xFFFEFF80, 128 ... OK
verify install area buffer [sig-sha256-ecdsa]...OK
software reset...
```

5. Activation processing is performed by the bootloader and a second software reset is applied.

```
==== RX65N : BootLoader [with buffer] ====
verify install area buffer [sig-sha256-ecdsa]...OK
activating image ... OK
software reset...
```

6. When the bootloader completes signature verification, the update image firmware is activated. When the process completes successfully, the following message is output and the LED flashes.

```
==== RX65N : BootLoader [with buffer] ====
verify install area main [sig-sha256-ecdsa]...OK
execute image ...

---------------------------------------
FWUP demo (ver 1.0.0)
---------------------------------------
Check the LEDs on the board.
```

Note: The demo program does not erase the buffer side. If you need to erase the image before updating for rollback measures, please add a process to erase the buffer side image.

4.4.3 Operation of Linear Mode Full Update Method

4.4.3.1 Execution Environment

Prepare the RX65N operation check environment (6.2.8). For MCU products other than RX65N, please refer to the operation confirmation environment of the applicable MCU product.

4.4.3.2 Building The Demo Project

Follow the steps below to build two demo projects for the full update method in linear mode.

The following procedures are described for the e2 studio environment; when using the IAR environment, please read and follow the procedures for the IAR integrated development environment.

1. Import the boot_loader, fwup_leddemo, and fwup_main demo projects into the integrated development environment.
2. Add the public key to be used to verify the image to the demo project.
   Paste the contents of secp256r1.publickey generated as described in 4.3.1 into code_signer_public_key.h in the boot_loader and fwup_main projects.

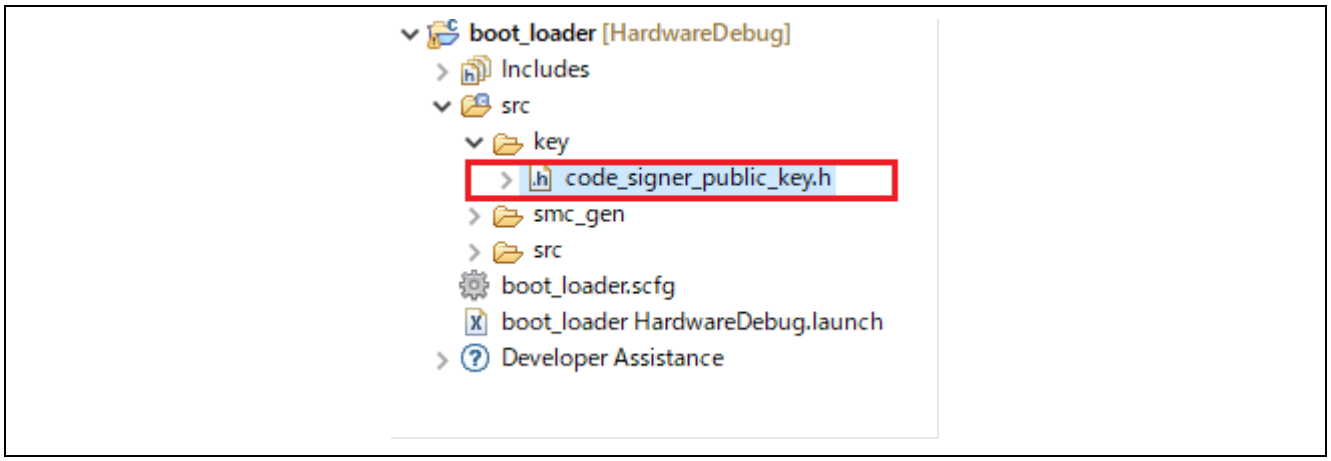


Figure 4.3 Storage Location of code_signer_public_key.h File in Demo Project

```
/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----"\
 * "...base64 data..."\
 * "-----END CERTIFICATE-----"
 */
#define CODE_SIGNER_PUBLIC_KEY_PEM \
"-----BEGIN PUBLIC KEY-----"\
Paste the contents of secp256r1.publickey here.
"-----END PUBLIC KEY-----"
#endif /* CODE_SIGNER_PUBLIC_KEY_H_ */
```

3. Build the demo projects.
   Build the first project (boot_loader) to generate boot_loader.mot.
   Build the second project (fwup_leddemo) to generate fwup_leddemo.mot.
   Rename fwup_leddemo.mot to fwup_leddemo_100.mot.
   Change the version of the second project (fwup_leddemo) as follows, then build it to generate fwup_leddemo.mot.

```
fwup_leddemo.c
---
#define FWUP_DEMO_VER_MAJOR        (1)
#define FWUP_DEMO_VER_MINOR        (0)
#define FWUP_DEMO_VER_BUILD        (0)★0->1
```

   Rename fwup_leddemo.mot to fwup_leddemo_101.mot.

4.4.3.3 Creating Initial and Update Images

The procedure for creating the initial and update images, using initial_firm.mot as the name of the initial image and fwup_leddemo_101.rsu as the name of the update image is, is described below.

1. Store the mot files created by building the demo projects in the same folder as Renesas Image Generator.

```
image-gen.py
RX65N_DualBank_ImageGenerator_PRM.csv
RX65N_Linear_Full_ImageGenerator_PRM.csv
RX65N_Linear_Half_ImageGenerator_PRM.csv
boot_loader.mot
fwup_leddemo_100.mot
fwup_leddemo_101.mot
secp256r1.privatekey
```

2. Execute the following command to create the initial image:

```
> python image-gen.py -iup fwup_leddemo_100.mot -ip
RX65N_Linear_Full_ImageGenerator_PRM.csv -o initial_firm -ibp
boot_loader.mot -vt ecdsa -key secp256r1.privatekey

Successfully generated the initial_firm.mot file.
```

3. Execute the following command to create the update image:

```
> python image-gen.py -iup fwup_leddemo_101.mot -ip
RX65N_Linear_Full_ImageGenerator_PRM.csv -o fwup_leddemo_101 -vt
ecdsa -key secp256r1.privatekey

Successfully generated the fwup_leddemo.rsu file.
```

Confirm that the initial and update image have been created in the same folder as Renesas Image Generator.

```
image-gen.py
RX65N_DualBank_ImageGenerator_PRM.csv
RX65N_Linear_Full_ImageGenerator_PRM.csv
RX65N_Linear_Half_ImageGenerator_PRM.csv
boot_loader.mot
fwup_main.mot
fwup_leddemo.mot
secp256r1.privatekey
fwup_leddemo_012.rsu
initial_firm.mot
```

4.4.3.4 Programming the Initial Image

Use Flash Writer to program the initial image (initial_firm.mot) to the MCU board. After programming, turn off the power to the board and disconnect the debugger (E2 Lite).

When writing with Renesas Flash Programmer, dual mode and linear mode are recognized as different microcontrollers, so an error will occur if the project is connected with a different bank mode from that of the microcontroller. For details on countermeasures, refer to chapter 2.1 of the following URL.

URL： https://www.renesas.com/jp/ja/document/apn/dual-mode-usage-guide?language=en&r=25424951

4.4.3.5 Executing a Firmware Update

Once the initial image firmware is activated, it waits for the transfer of the update image via the terminal emulator. The received update image is programmed to the flash memory, and after the transfer completes, the signature of the update image is verified and the firmware is activated.

Follow the steps below to execute a firmware update.

1. Refer to 6.2.8, Execution Environment, and connect the devices.
2. Launch the terminal emulator software on the PC, select the serial COM port, and configure the connection settings.
3. Power on the board. The following message is output:

```
==== RX65N : BootLoader [without buffer] ====
verify install area main [sig-sha256-ecdsa]...OK
execute image ...

----------------------------------------
FWUP demo (ver 1.0.0)
----------------------------------------

Check the LEDs on the board.
```

4. Turn ON->OFF RESET_SW while pressing USER_SW.

```
==== RX65N : Image updater [without buffer] ====
send image(*.rsu) via UART.
```

5. Send the updated image via the terminal emulator.
   Send file… > check Binary > fwup_leddemo_101.rsu

   The following message is output during the transfer of the update image, and a software reset is applied after installation and signature verification complete.

```
W 0xFFE00000, 128 ... OK
W 0xFFE00080, 128 ... OK
...
W 0xFFE03B00, 128 ... OK
W 0xFFFEFF80, 128 ... OK
verify install area main [sig-sha256-ecdsa]...OK
software reset...
```

6. When the bootloader completes signature verification, the update image firmware is activated. When the process completes successfully, the following message is output and the LED flashes.

```
==== RX65N : BootLoader [without buffer] ====
verify install area main [sig-sha256-ecdsa]...OK
execute image ...

---------------------------------------
FWUP demo (ver 1.0.1)
---------------------------------------
Check the LEDs on the board.
```

5. Renesas Image Generator

Renesas Image Generator is a utility tool that generates firmware images for use with firmware update modules. The Renesas Image Generator can generate the following images used by the firmware update module.

- Initial image: An image file containing the bootloader and application program that is programmed using Flash Writer at the time of initial system configuration (extension: mot).
- Update image: An image file containing the firmware update (extension: rsu).

See 5.1 for how to generate an image, and 5.2 to 5.3 for details on image configuration and parameter files.

Renesas Image Generator is a program that runs on Python.

5.1 Image Generation Methods

Describes the specifications of Renesas Image Generator (image-gen.py) and how to generate an image file (initial image or update image) using this tool.

See 5.1.1 for how to generate an initial image, and 5.1.2 for how to generate an update image.

The format of the image-gen.py command is as follows:

```
python image-gen.py < options >
```

Some image-gen.py command options are required and others are optional. Table 5.1 lists the required image-gen.py options, and Table 5.2 lists the optional image-gen.py options.

Table 5.1 Required Options of image-gen.py

Option	Description
-iup <file>	Specifies the application program. For the character string < file >, specify the mot file name (the full path including the file name) of the user application program.
-ip <file>	Specifies a parameter file. For the character string < file >, specify the name of the file (the full path including the file name) containing the parameters to be input.
-o <file>	Specifies the file name of the output image. For the character string < file >, specify the file name (the full path including the file name), excluding the extension, of the firmware update image file to be output. The file extension is .mot because the output image is determined to be the initial image when the bootloader is specified with the -ibp <file> option. If you omit the -ibp <file> specification, the output image is determined to be an update image and becomes .rsu.

Table 5.2 Optional Options of image-gen.py

Option	Description
-ibp <file>	Specifies the bootloader. For the character string < file >, specify mot file name (the full path including the file name) of the bootloader program. Specify this option when generating a mot file.
--key <file>	Specify the name of the key file to be used to sign the image using ECDSA. (This option does not need to be set if sha256 is specified for the -vt option.) Store the file **secp256r1.privatekey** in the command execution folder. If the file name has been changed, specify the full path or the relative path including the file name.
-vt <VerificationType>[sha256 / ecdsa]	Specifies the image verification method in the firmware update module. The following VerificationType can be specified. sha256: Append a hash of the image. If this option is omitted, "sha256" is specified. ecdsa: Adds an image signature. The key file specified by -key is used to generate signature data. An error will result if the key file is not specified with -key.
-ff <FileFormat>[BareMetal / RTOS]	Specifies the RSU format type. The following FileFormat can be specified BareMetal: Generates an image of the application program data with RSU header signature information. This is the RSU format used in the demo project. If this option is omitted, "BareMetal" is specified. RTOS: Generates an updated image for FreeRTOS OTA. Update images for FreeRTOS OTA do not add RSU header signature information. BareMetal_FWUP_V2_V1_DATA: For special purpose. RTOS_FWUP_V2_V1_DATA: For special purpose.
-h	Output a list of commands. Specify this option to display help information for the tool.

5.1.1 Initial Image Generation Method

Renesas Image Generator has the bootloader file name (.mot) generated by build, application program (.mot), parameter file name (.csv), output file name (no extension), image verification method in firmware update module. Specify (ecdsa/sha256) as a command line option to generate an initial image file (.mot).

Command input example

```
> python image-gen.py -iup fwup_main.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o initial_firm -ibp
boot_loader.mot -vt ecdsa -key secp256r1.privatekey
```

fwup_main.mot: The mot file name of the user application program
RX65N_DualBank_ImageGenerator_PRM.csv: The name of the file containing the parameters to be input
(Example of dual bank mode)
initial_firm: The file name of the initial image file to be output
boot_loader.mot: The mot file name of the bootloader program
ecdsa: Specifies that ECDSA is used to sign the image.
secp256r1.privatekey: Key file name for signing images with ECDSA.

5.1.2 Update Image Generation Method

The Renesas Image Generator uses the update application program (.mot) generated by the build, parameter file name (.csv), output file name (no extension), image verification method (ecdsa/sha256) for the firmware update module. Set the command line options to generate an update image file (.rsu).

Command input example

```
> python image-gen.py -iup fwup_leddemo.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o fwup_leddemo -vt ecdsa -key
secp256r1.privatekey
```

fwup_leddemo.mot: The mot file name of the user application program to be applied as an update
RX65N_DualBank_ImageGenerator_PRM.csv: The name of the file containing the parameters to be input
(Example of dual bank mode)
fwup_leddemo: The file name of the update image file to be output
ecdsa: Specifies that ECDSA is used to sign the image.
secp256r1.privatekey: Key file name for signing images with ECDSA.

5.2 Image File

5.2.1 Update Image File

Figure 5.1 shows the configuration diagram of the update image file generated by Renesas Image Generator.

For the format of the RSU header, see Table 43.

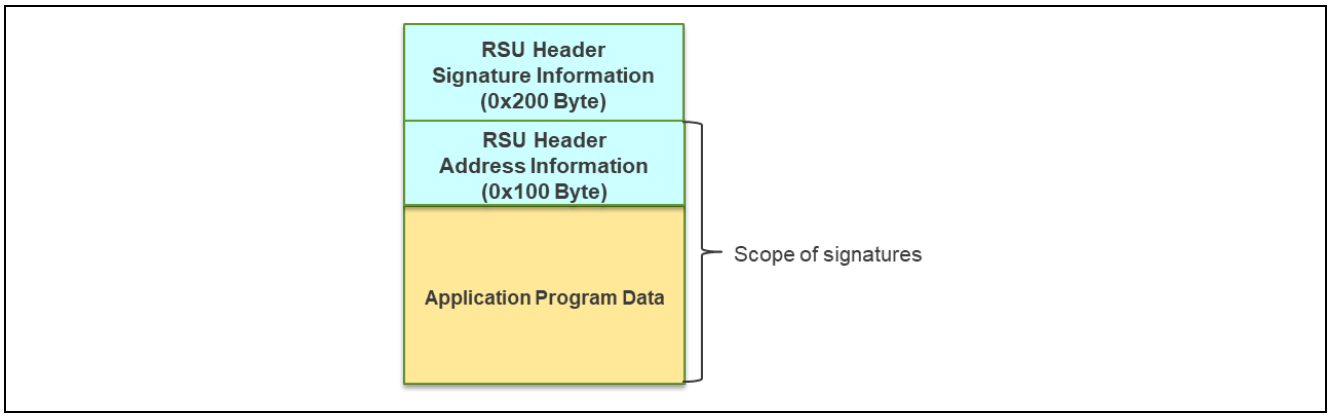


Figure 5.1 Configuring the update image file

The update image file consists of RSU header and application program data. The RSU header stores the application program location information required to verify the validity of the application program, as well as the signature value and hash value of the application program calculated based on the information. Following the RSU header, place the application program data corresponding to the program allocation information stored in the RSU header. The Renesas Image Generator arranges the application program data in the order of the data to be placed in the code flash and the data to be placed in the data flash. Valid code flash data and data flash data are extracted from the user-generated application program file (.mot), converted to binary data, and set.

The update image file has the same configuration for the dual bank method, linear mode half-updating method, and linear mode full-updating method.

Table 5.3 RSU Header Format (1/2)

Offset	Item	Length (Bytes)	Description
0x00000000	Magic Code	7	Magic code (“RELFWV2”)
0x00000007	Reserved	1	Reserved area
0x00000008	Firmware Verification Type	32	Image verification method Set sig-sha256-ecdsa to use ECDSA for image verification, and hash-sha256 to use hash.
0x00000028	Signature size	4	Data size of signature value or hash value stored in Signature Set 0x40 if Firmware Verification Type is sig-sha256-ecdsa, and 0x20 if hash-sha256.
0x0000002C	Signature	64	Signature value used for firmware verification For SHA-256 signature data, bytes 33 to 64 are set to 0x00.
0x0000006C	RSU File Size	4	File size of entire update image file
0x00000070	Reserved	400	Reserved area

Table 5.4 RSU Header Format (2/2)

Offset	Item	Length (Bytes)	Description
0x00000200	Program Data Num	4	Number of subsequent divided application programs or data flashes (maximum 31)
0x00000204	Start Address[0]	4	Start address of the first application program or data flash
0x00000208	Data Size[0]	4	Size of the first application program or data flash
0x0000020C	Start Address[1]	4	Start address of second application program or data flash
0x00000210	Data Size[1]	4	Second application program or data flash size
:	:		
0x000002F4	Start Address[30]	4	Start address of the 31st application program or data flash
0x000002F8	Data Size[30]	4	Size of the 31st application program or data flash
0x000002FC	Reserved	4	Reserved area

See Figure 5.2 for the mechanism of generating the update image file.

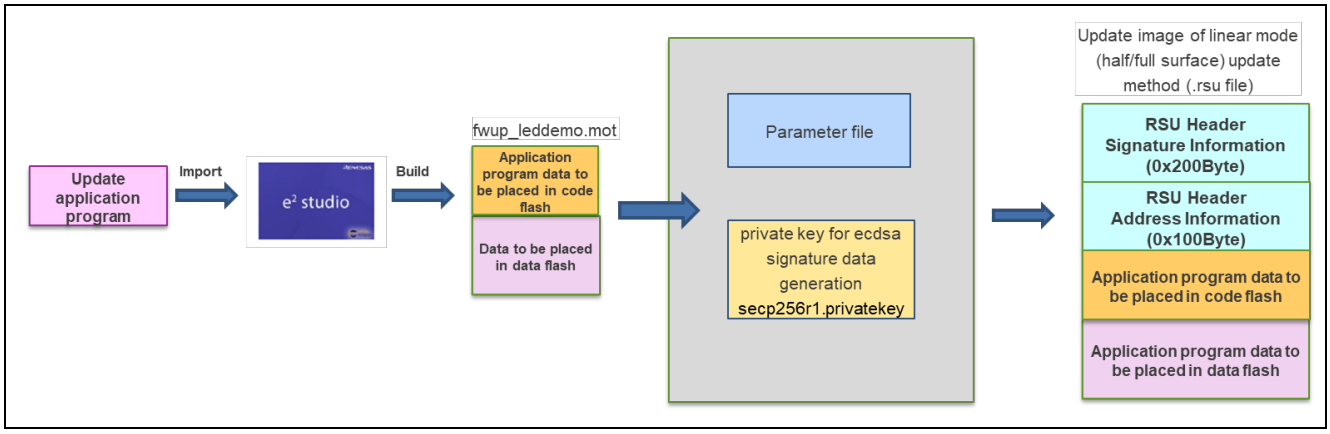


Figure 5.2 Updating image of dual bank method/linear mode (half side/whole side) updating method

- The parameter file is a CSV format file that contains the device address information required to generate the image file.
- The private key for generating the ecdsa signature value is used when ecdsa is specified as the image verification method in the firmware update module.

5.2.2 Initial Image File

The initial image file is the RSU header and application program data plus the bootloader program data.

Figure 5.3 and Figure 5.4 also show a diagram of the initial image file (dual bank method/linear mode (half/full surface) update method).

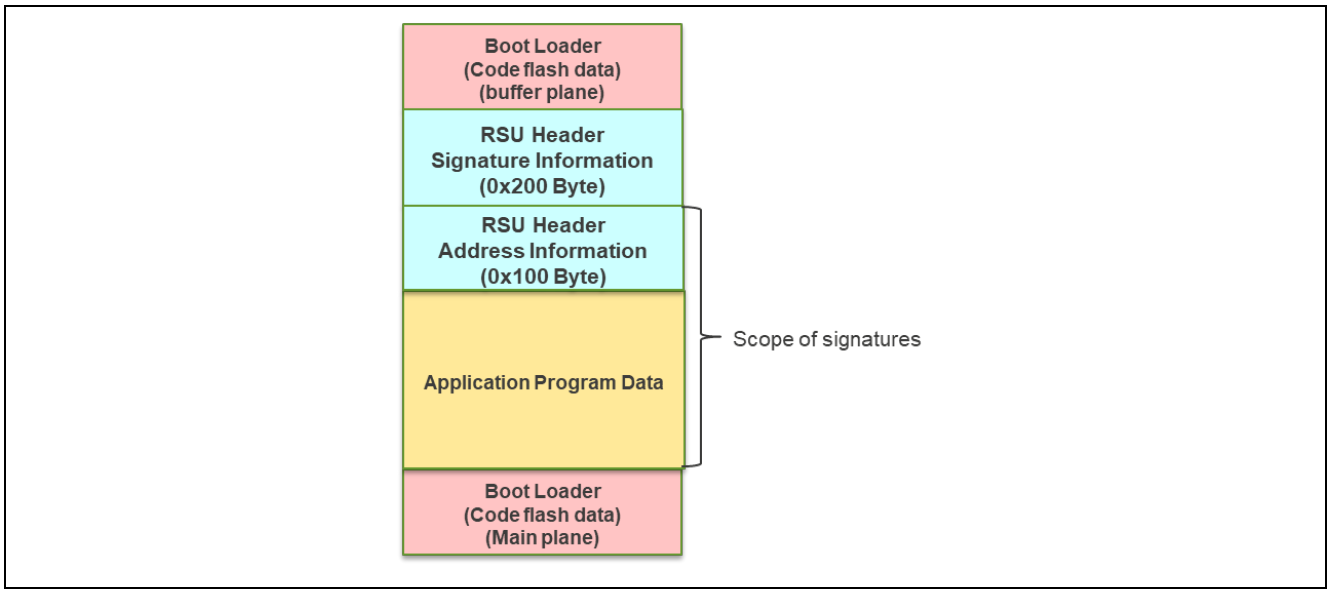


Figure 5.3 Initial image file (dual bank method) configuration

The initial image of the dual-bank method places the same bootloader code flash data on both the main and buffer sides of the code flash to support the bank switching function. The bootloader data to be placed on the main side of Code Flash uses the data in the user-generated bootloader file (boot_loader.mot) as is. The bootloader placed on the buffer side of the code flash is equivalent to the bootloader data on the main side.

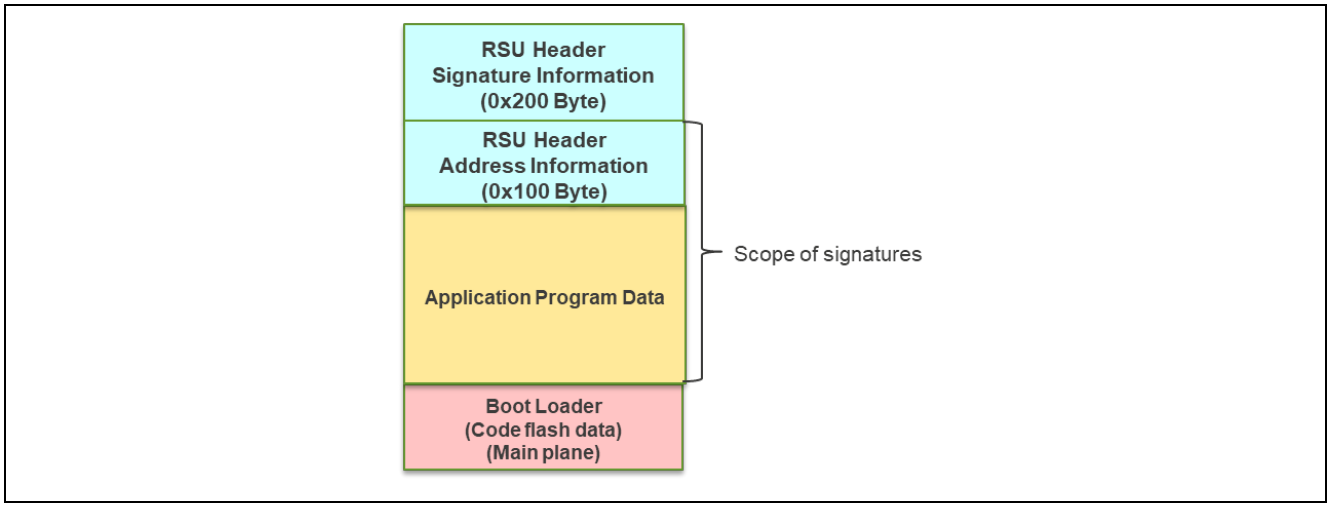


Figure 5.4 Composition of initial image file (linear mode (half/full surface) update method)

In the initial image file of the linear mode (half-face/full-face) update method, the bootloader data to be placed on the main side of the code flash uses the data in the user-generated bootloader file (boot_loader.mot) as is.

See Figure 5.5 and Figure 5.6 for the mechanism that generates the initial image file.

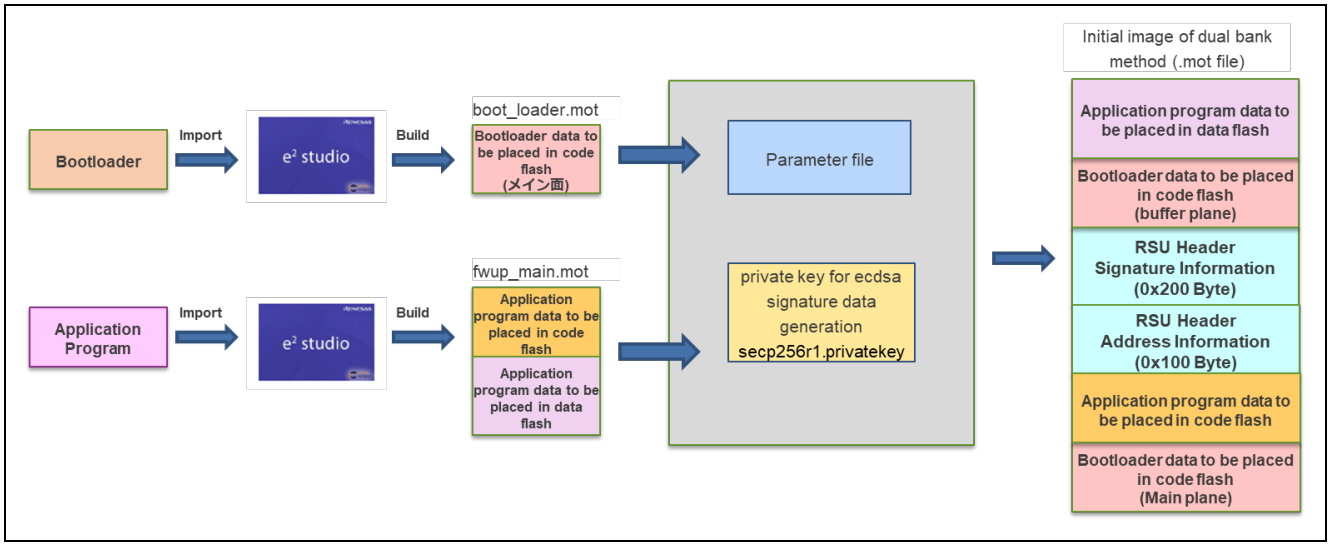


Figure 5.5 Initial image of the dual bank method

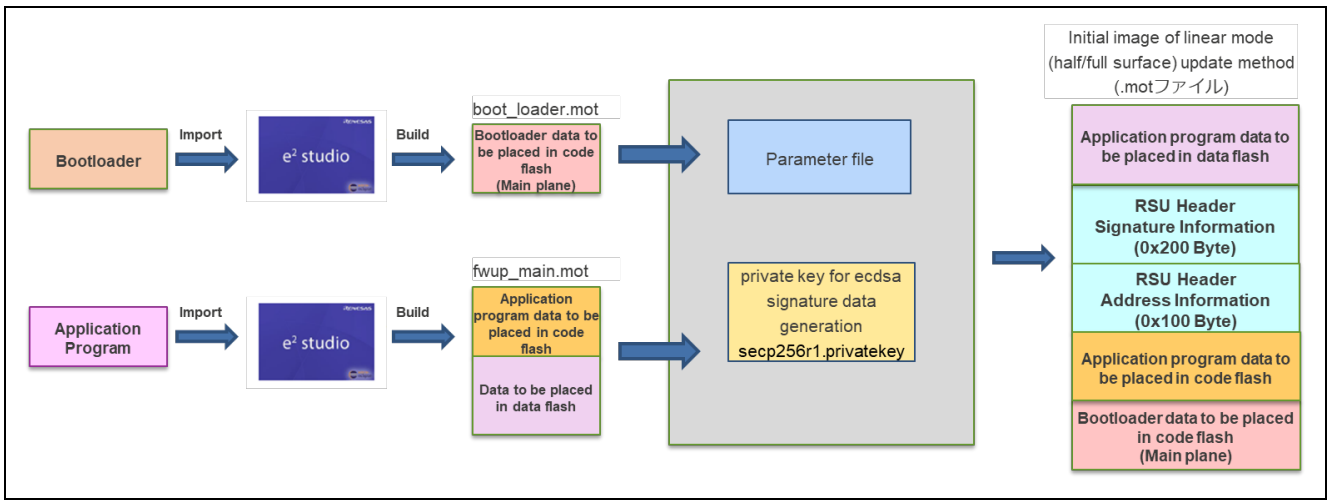


Figure 5.6 Initial image of linear mode (half/full) update method

- The parameter file is a CSV format file that contains the device address information required to generate the image file.
- The private key for generating the ecdsa signature value is used when ecdsa is specified as the image verification method in the firmware update module.

5.3 Parameter File

The parameter file is the information required for Renesas Image Generator to generate the initial and updated image files for the sample program, and is included in the release package as part of the Renesas Image Generator Python It is included in the release package as part of the Renesas Image Generator Python program set (see1.5). When a customer generates an initial or updated image for a demo project, there is no need to change the contents of the parameter file.

If you are using a product with a different flash size than the demo project (see 5.3.2) or if you do not want to include data from the data flash in the image (see 5.3.3), you can do so by editing the parameter file.

As an example, the contents of the parameter file for the RX65N (2MB) dual bank system are shown in 5.3.1.

5.3.1 Contents of Parameter File

The items listed in the parameter file are the same for all devices, but the settings differ for each device. Table 5.5 shows the contents of the parameter file for the RX65N (2MB) dual bank method demonstration project. Figure 5.7 shows the parameters referenced for image generation, and Figure 5.8 shows an example of parameters referenced for initial image generation for the RX65N (2MB) dual bank system.

Table 5.5 Contents of parameter file

Parameter name	Description	Example of setting contents RX65N(2MB)
device Type	Dual Mode：Generation of mot file for dual bank system Linear Mode：Linear mode (half/full surface) update method Mot file generation for	Dual Mode
Code Flash Size(Dual Mode Only)	Code Flash Size (Used to calculate the bootloader address of the buffer surface in the dual bank method)	0x00200000
Bootloader Start Address	Bootloader start address	0xFFFF0000
Bootloader End Address	Bootloader end address	0xFFFFFFFF
User Program Start Address	Starting address of the application program on the main face (In dual mode, application program area on main side)	0xFFF00000
User Program End Address	End address of the application program on the main side (in dual mode, application program area on main side)	0xFFFEFFFF
OFS Data Start Address	OFSM data start address (Set 'No Used.' for non-dual bank products)	0xFE7F5D00
OFS Data End Address	OFSM data end address (Set 'No Used.' for non-dual bank products)	0xFE7F5D7F
Data Flash Start Address	Data flush start address (Set 'No Used.' if data flush data is not to be generated)	0x00100000
Data Flash End Address	Data flash end address (Set 'No Used.' if data flash data is not to be generated)	0x00107FFF
Near Data Start Address(RL78 Only)	Near bootloader start address for RL78 (For RX, set 'No Used.)	No Used.
Near Data End Address(RL78 Only)	Near boot loader start address for RL78 (For RX, set 'No Used.')	No Used.
Flash Write Size	Flash write size (number of bytes required for one write to the flash in decimal)	128

The value specified for each parameter is specified in decimal for Flash Write Size and in hexadecimal (with 0x added at the beginning) for other parameters.

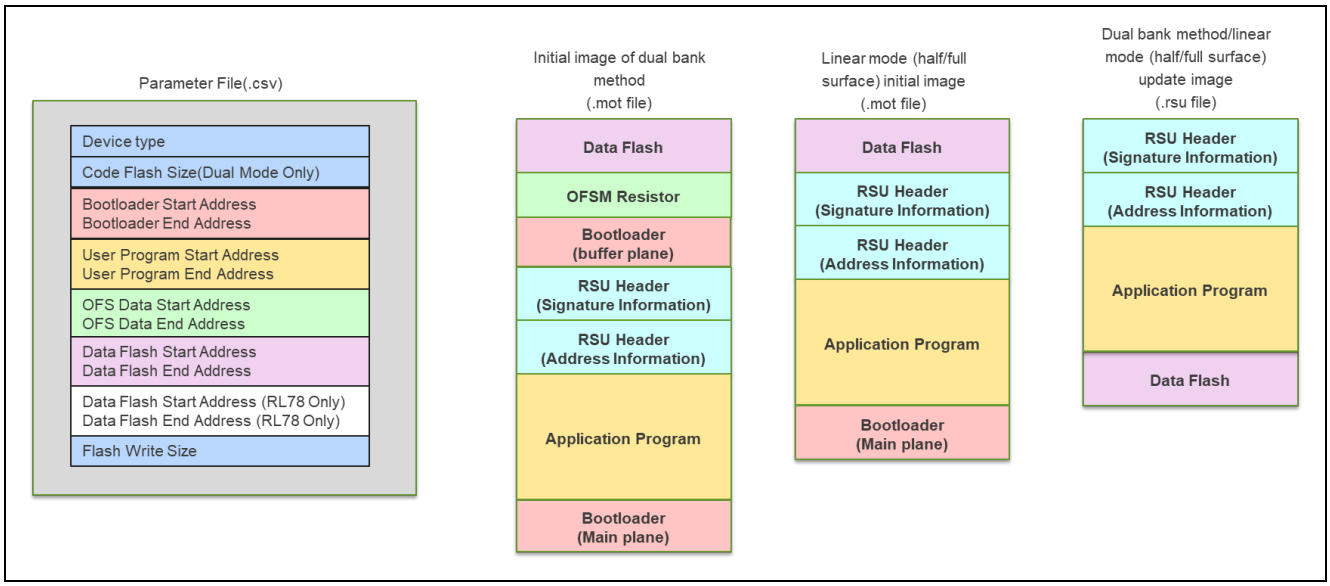


Figure 5.7 Parameters referenced when generating image files

- Device type is used to determine whether to generate a dual-banked initial image; if Device type is Dual Mode, a bootloader (main side) and a bootloader (buffer side) are generated; if Device type is Linear Mode, only the bootloader (Main plane) is generated only in the case of Linear Mode.
- Code Flash Size (Dual Mode Only) is used to determine the address where the bootloader (buffer plane) is placed.
- Using the bootloader file (boot_loader.mot) as input data, the range from Bootloader Start Address to Bootloader End Address is generated as a code flash for the bootloader (main plane).
- With the application program file (.mot) as input data, the range from User Program Start Address to User Program End Address is generated as an application program code flash.
- Using the bootloader file (boot_loader.mot) as input data, the range from OFS Data Start Address to OFS Data End Address is generated as OFS registers.
- Using the application program file (.mot) as input data, the range from Data Flash Start Address to Data Flash End Address is generated as a data flash.
- Flash Write Size is used to set the data size of the RSU header (address information) as the minimum unit when writing to the flash.

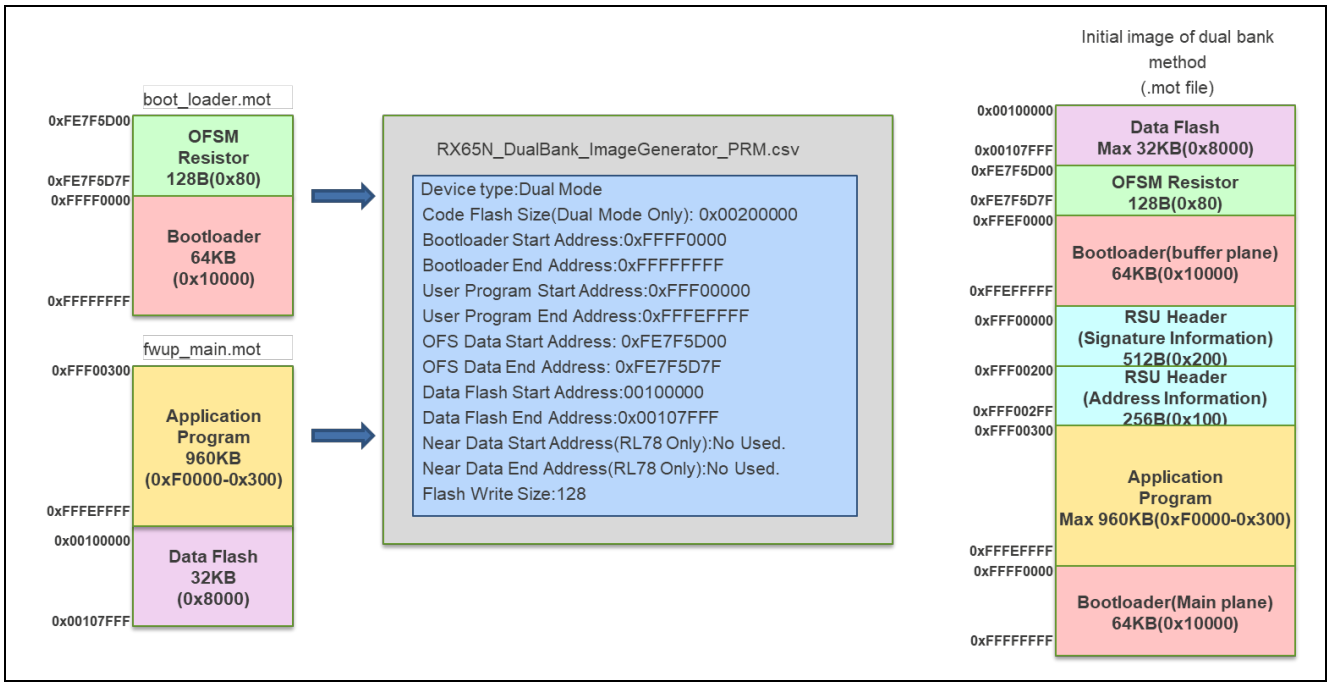


Figure 5.8 RX65N (2MB) Example of parameters referenced for initial image generation for dual bank method

5.3.2 How to generate an image with a flash size different from the demo project

If you want to perform firmware updates on products with different flash sizes that are compatible with the demo project, you can generate initial and updated images by editing the parameter file.

Figure 5.9 shows the contents of the parameter file using the RX65N dual bank method (1.5 MB) as an example. (The parameters in the red box show the differences from RX65N_DualBank_ImageGenerator_PRM.csv)

The parameter file for the RX65N dual bank method (1.5MB) is not included in the package, so customers must edit the parameters themselves.

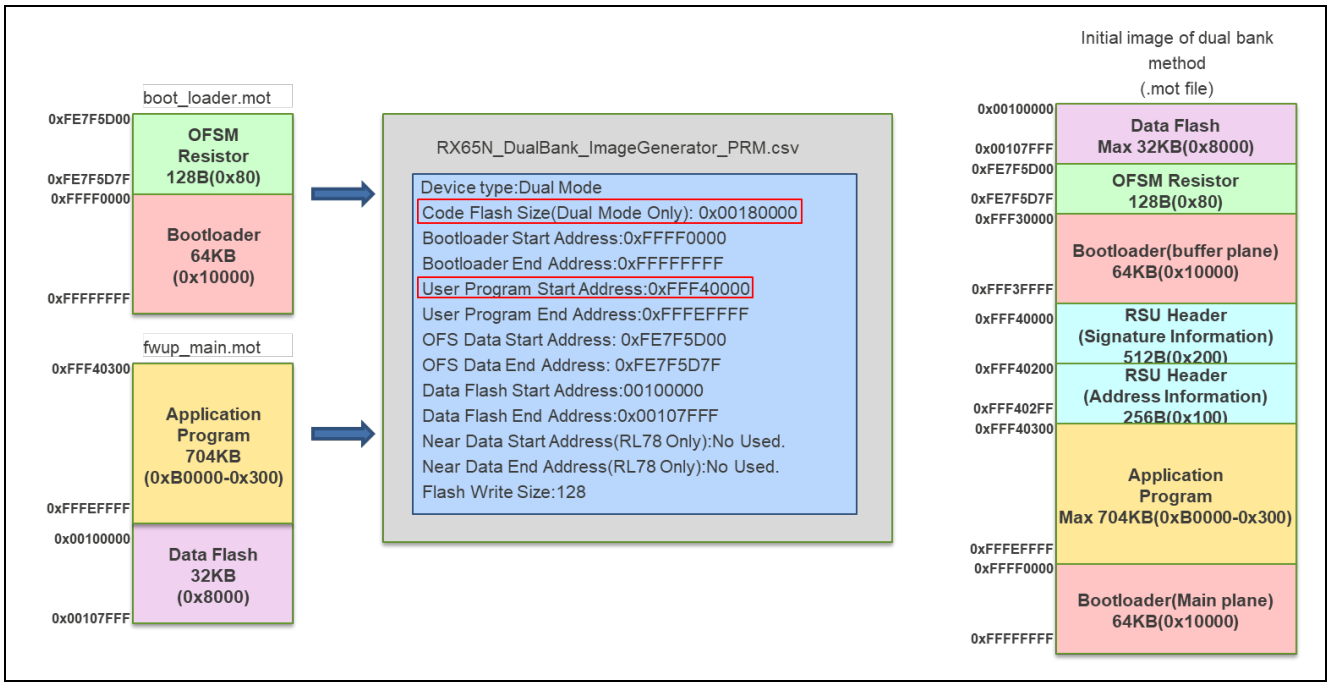


Figure 5.9 Example of parameter setting for RX65N (1.5MB) dual bank method

- Code Flash Size (Dual Mode Only) describes the code flash memory capacity of RX65N (1.5MB).
- The User Program Start Address is the address following the last address of the bootloader (buffer plane).

5.3.3 How to prevent data flash data from being included in the image

By setting the Data Flash Start Address and Data Flash End Address to "No Used." in the parameter file, the data flash data is not included in the initial image or update image. Data flash data is not included in the initial image or update image.

Figure 5.10 shows an example of parameter file settings for the RX65N (2MB) dual-bank system with the data flash not included in the update image.

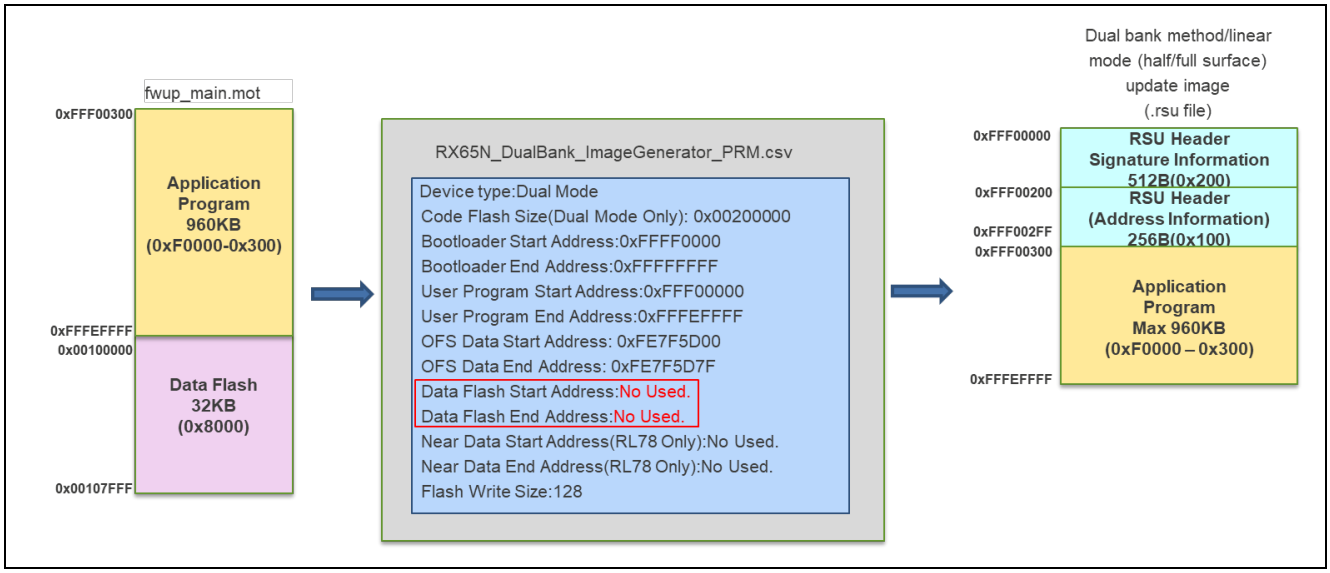


Figure 5.10 Parameter settings when data flash is not included in the update image (example for RX65N)

6. Appendices

6.1 Confirmed Operation Environments

This section describes environments on which the operation of the FIT module has been confirmed.

Table 6.1 Confirmed Operation Environment (CC-RX)

Item	Description
Integrated development environment	Renesas Electronics e^2 studio 2024-01
C compiler	Renesas Electronics C/C++ Compiler for RX Family V3.04.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.2.03
Board used	Renesas Starter Kit for RX130-512KB (product No.:RTK5051308SxxxxxBE) Renesas Starter Kit for RX140-256KB (product No.:RTK551406BxxxxxBJ) Renesas Starter Kit for RX231 (product No.:R0K505231SxxxBE) Renesas Solution Starter Kit for RX23E-A (product No.:RTK0ESXB10C00001BJ) Renesas Solution Starter Kit for RX23E-B (product No.:RTK0ES1001C00001BJ) Renesas Starter Kit+ for RX24T（product No.：RTK500524TS00000BE） Renesas Flexible Motor Control Kit for RX26T (product No.: RTK0EMXE70S00020BJ) Renesas Starter Kit+ for RX65N (product No.：RTK50565N2SxxxxxBE） Renesas Starter Kit for RX66T (product No.:RTK50566T0S00000BE) Renesas Starter Kit for RX660 (product No.:RTK556609HCxxxxxBJ) Renesas Starter Kit+ for RX671 (product No.:RTK55671EHS10000BE) Renesas Starter Kit+ for RX72N (product No.:RTK5572NNxxxxxxxBE)

Table 6.2 Confirmed Operation Environment (GCC)

Item	Description
Integrated development environment	Renesas Electronics e^2 studio 2024-01
C compiler	GCC for Renesas RX 8.3.0.202305 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
Endian order	Little endian
Revision of the module	Rev.2.03
Board used	Renesas Starter Kit for RX130-512KB (product No.:RTK5051308SxxxxxBE) Renesas Starter Kit for RX140-256KB (product No.:RTK551406BxxxxxBJ) Renesas Starter Kit for RX231 (product No.:R0K505231SxxxBE) Renesas Solution Starter Kit for RX23E-A (product No.:RTK0ESXB10C00001BJ) Renesas Solution Starter Kit for RX23E-B (product No.:RTK0ES1001C00001BJ) Renesas Starter Kit+ for RX24T（product No.：RTK500524TS00000BE） Renesas Flexible Motor Control Kit for RX26T (product No.: RTK0EMXE70S00020BJ) Renesas Starter Kit+ for RX65N（product No.：RTK50565N2SxxxxxBE） Renesas Starter Kit for RX66T (product No.:RTK50566T0S00000BE) Renesas Starter Kit for RX660 (product No.:RTK556609HCxxxxxBJ) Renesas Starter Kit+ for RX671 (product No.:RTK55671EHS10000BE) Renesas Starter Kit+ for RX72N (product No.:RTK5572NNxxxxxxxBE)

Table 6.3 Confirmed Operation Environment (IAR)

Item	Description
Integrated development environment	IAR Embedded Workbench for Renesas RX 5.10.1 RX Smart Configurator V2.14.0
C compiler	IAR C/C++ Compiler for Renesas RX 5.10.1 Compiler option: The default settings of the integrated development environment
Endian order	Little endian
Revision of the module	Rev.2.03
Board used	Renesas Starter Kit for RX130-512KB (product No.:RTK5051308SxxxxxBE) Renesas Starter Kit for RX140-256KB (product No.:RTK551406BxxxxxBJ) Renesas Starter Kit for RX231 (product No.:R0K505231SxxxBE) Renesas Solution Starter Kit for RX23E-A (product No.:RTK0ESXB10C00001BJ) Renesas Solution Starter Kit for RX23E-B (product No.:RTK0ES1001C00001BJ) Renesas Starter Kit+ for RX24T（product No.：RTK500524TS00000BE） Renesas Flexible Motor Control Kit for RX26T (product No.: RTK0EMXE70S00020BJ) Renesas Starter Kit+ for RX65N（product No.：RTK50565N2SxxxxxBE） Renesas Starter Kit for RX66T (product No.:RTK50566T0S00000BE) Renesas Starter Kit for RX660 (product No.:RTK556609HCxxxxxBJ) Renesas Starter Kit+ for RX671 (product No.:RTK55671EHS10000BE) Renesas Starter Kit+ for RX72N (product No.:RTK5572NNxxxxxxxBE)

The versions of the FIT modules used by the demo project to confirm firmware update operation are listed below.

Table 6.4 FIT Module Versions (CC-RX)

Device	Project	r_bsp	r_byteq	r_flash	r_sci	r_fwup
RX130	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX140	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX231	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX23E-A	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX24E-B	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX24T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX26T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX65N	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX65T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX660	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX671	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX72N	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-

Table 6.5 FIT Module Versions (GCC)

Device	Project	r_bsp	r_byteq	r_flash	r_sci	r_fwup
RX130	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX140	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX231	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX23E-A	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX24E-B	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX24T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX26T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX65N	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX65T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX660	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX671	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX72N	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-

Table 6.6 FIT Module Versions (IAR)

Device	Project	r_bsp	r_byteq	r_flash	r_sci	r_fwup
RX130	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX140	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX231	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX23E-A	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX24E-B	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX24T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX26T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX65N	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX65T	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX660	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX671	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-
RX72N	boot_loader	7.42	2.10	5.10	4.90	2.03
	fwup_main	7.42	2.10	5.10	4.90	2.03
	fwup_leddemo	7.42	2.10	-	4.90	-

6.2 Operating Environment for Demo Project

This module supports multiple compilers. When using this module, the different settings for each compiler are shown below.

6.2.1 Operation Confirmation Environment for RX130

The execution environment and connection diagram are shown below.

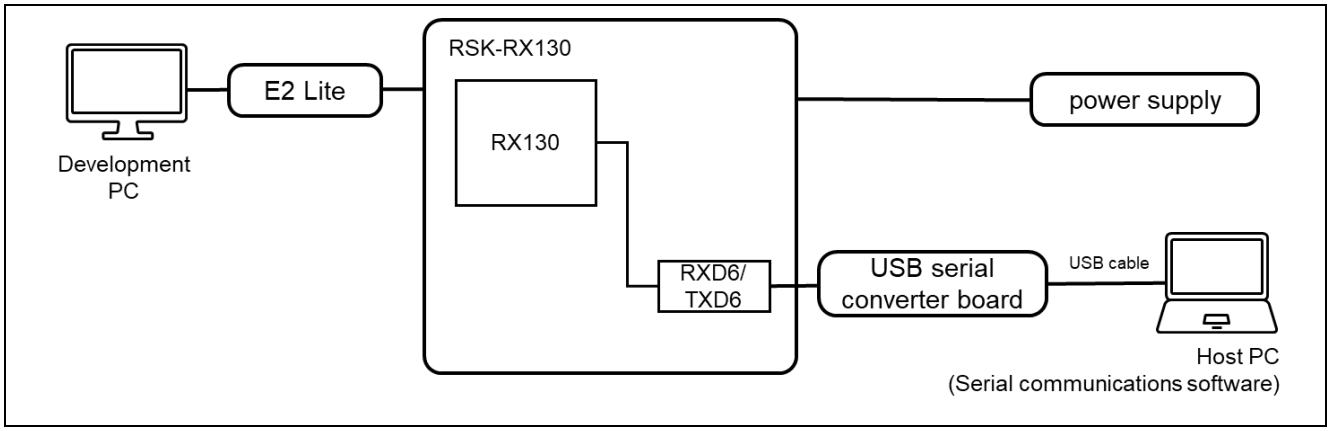


Figure 6.1 RSK-RX130 Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD6	RX
3	RXD6	TX
4	PB3(RTS)	CTS

■ USER_SW (②)

SW1	Note
P31	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RESn	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
PD3	LED0

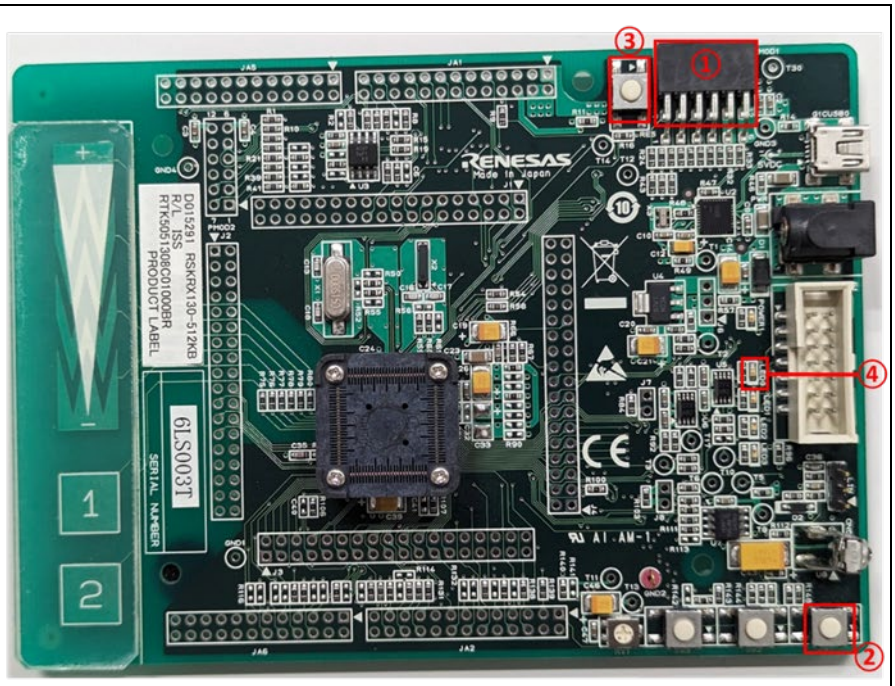


Figure 6.2 RSK-RX130 Pin Information

6.2.1.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX130 linear mode half-surface update method demo project and the memory map of the configuration settings.

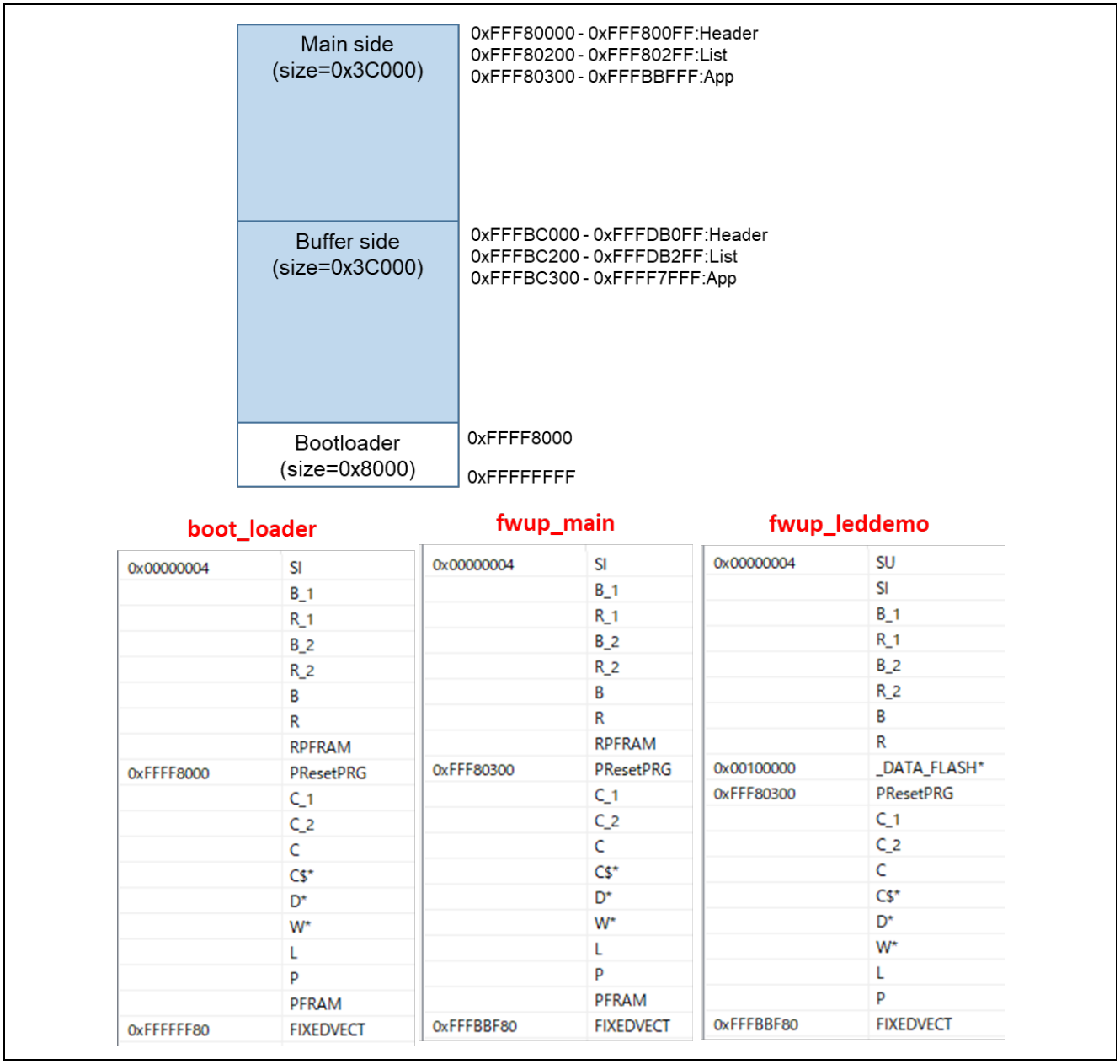


Figure 6.3 RX130 linear mode half-surface update method demo project memory map

Table 6.7 RX130 linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFBC000	
FWUP_CFG_AREA_SIZE	0x3C000	
FWUP_CFG_CF_BLK_SIZE	0x400	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x100000	
FWUP_CFG_DF_BLK_SIZE	1024	
FWUP_CFG_DF_NUM_BLKS	8	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.1.2 Memory map of demo project for full update method in linear mode

The memory map of the RX130 linear mode full update method demo project and the memory map of the configuration settings are shown below.

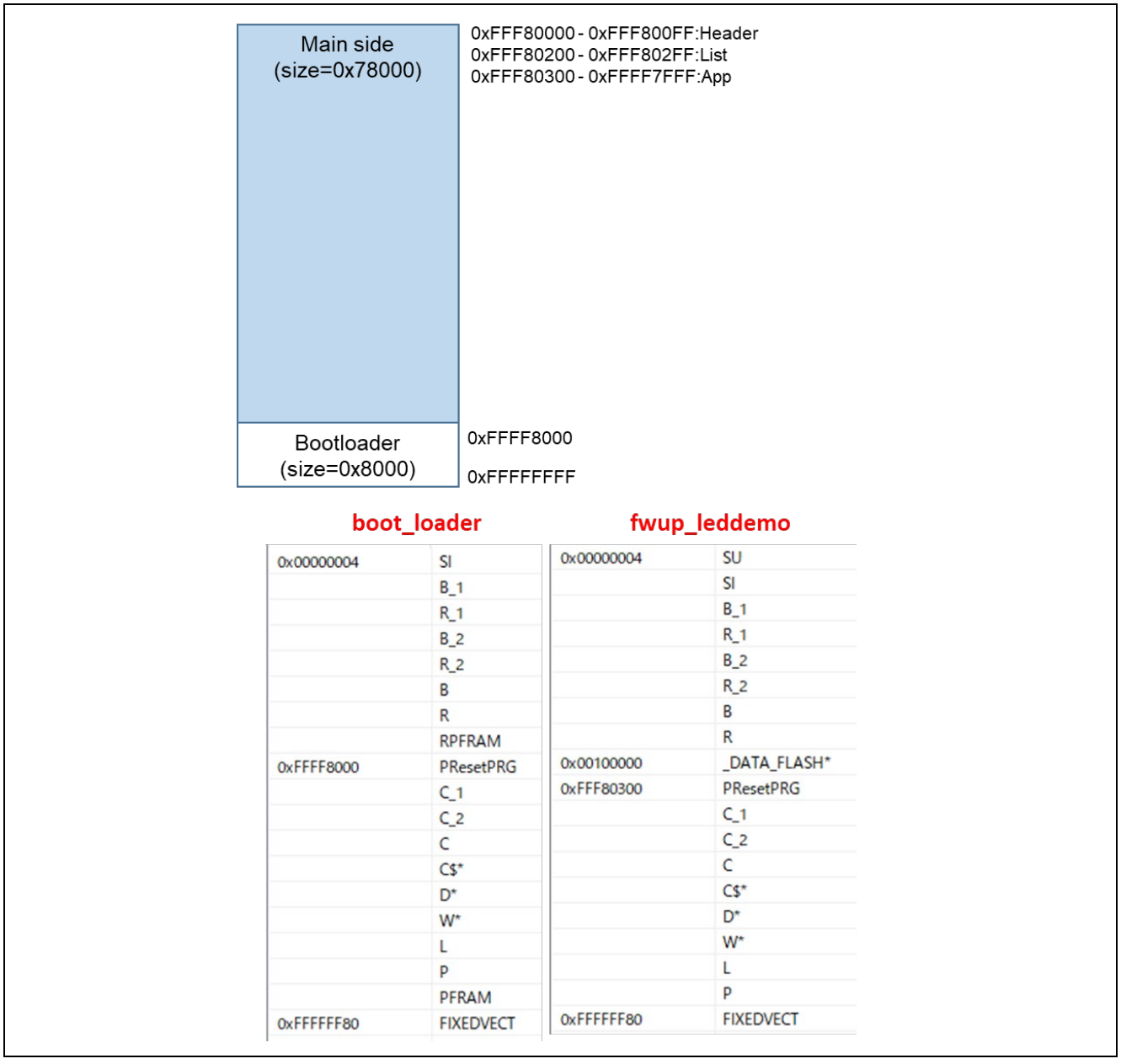


Figure 6.4 RX130 linear mode full update method demo project memory map

Table 6.8 RX130 linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF80000
FWUP_CFG_AREA_SIZE	0x78000
FWUP_CFG_CF_BLK_SIZE	0x400
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x100000
FWUP_CFG_DF_BLK_SIZE	1024
FWUP_CFG_DF_NUM_BLKS	8
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.2 Operation Confirmation Environment for RX140

The execution environment and connection diagram are shown below.

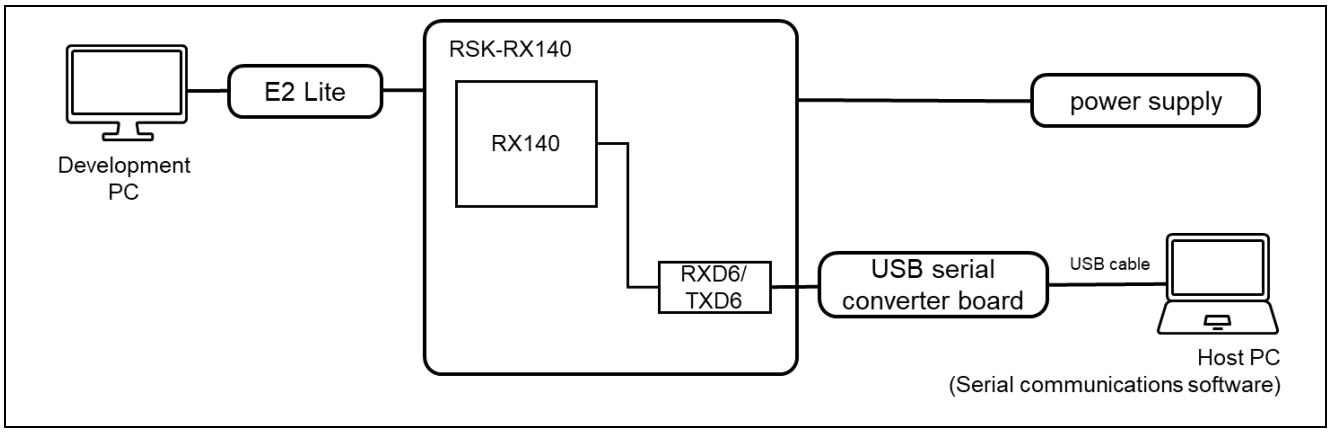


Figure 6.5 RSK-RX140 Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD6	RX
3	RXD6	TX
4	PB3(RTS)	CTS

■ USER_SW (②)

SW1	Note
P31	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P21	LED0

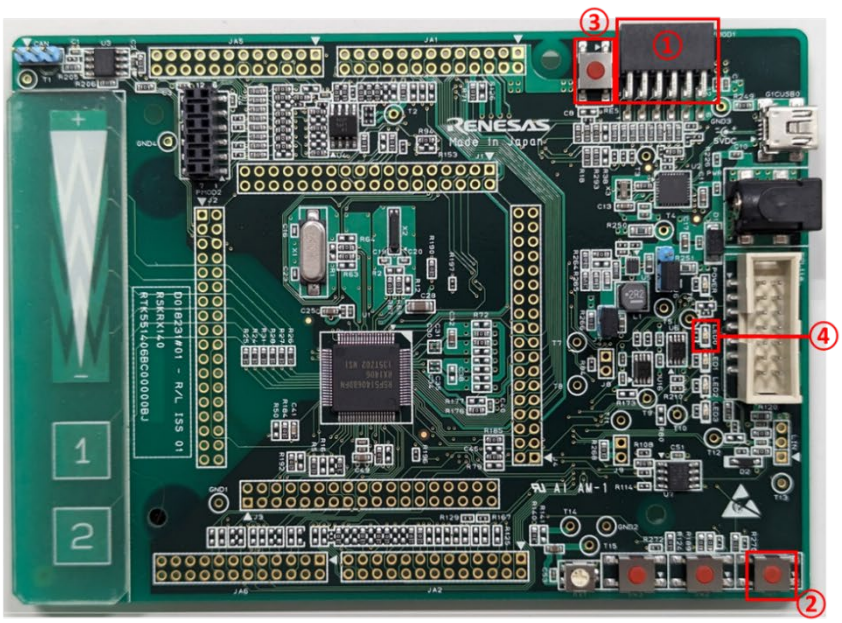


Figure 6.6 RSK-RX140 Pin Information

6.2.2.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX140 linear mode half-surface update method demo project and the memory map of the configuration settings.

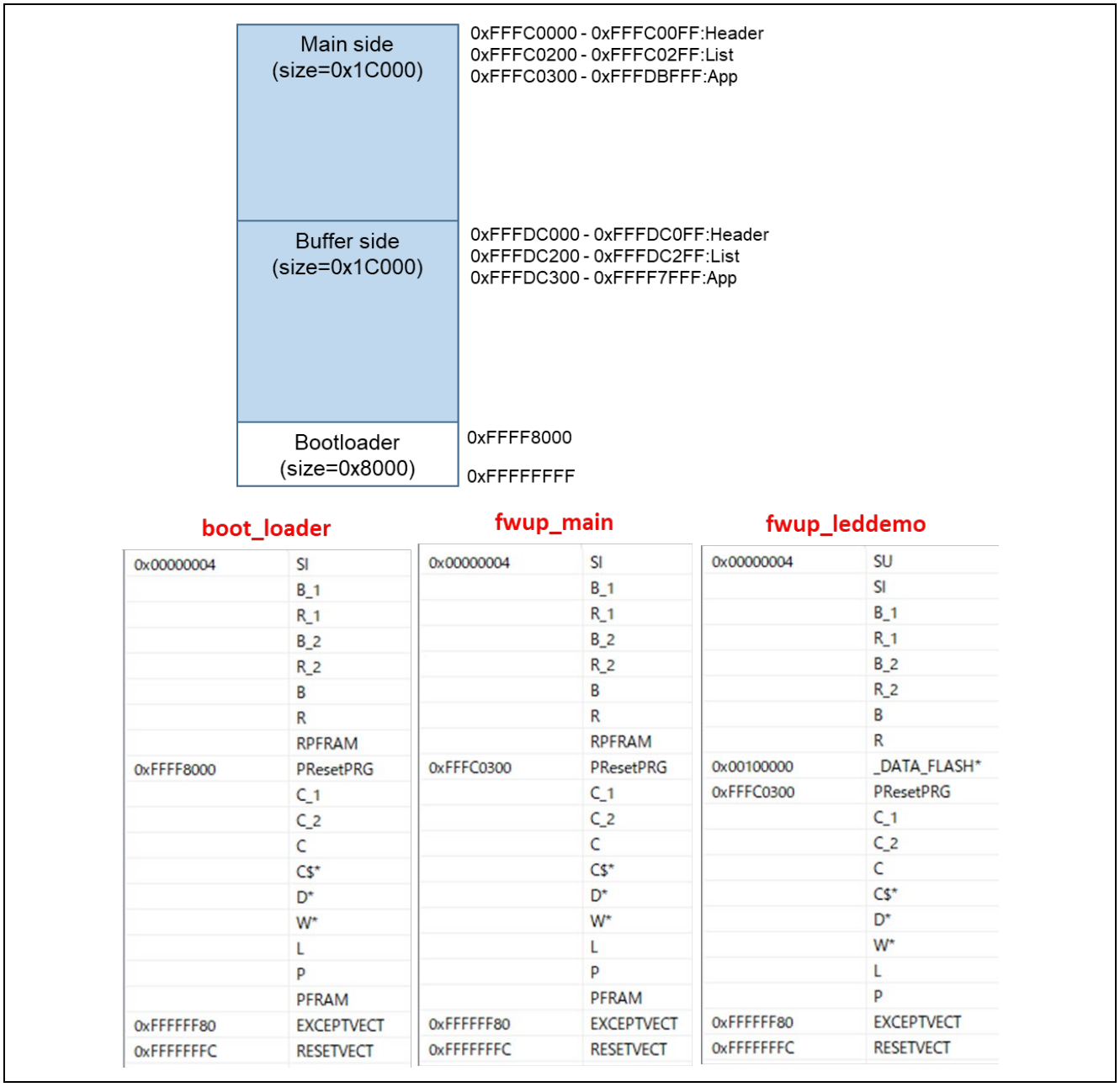


Figure 6.7 RX140 linear mode half-surface update method demo project memory map

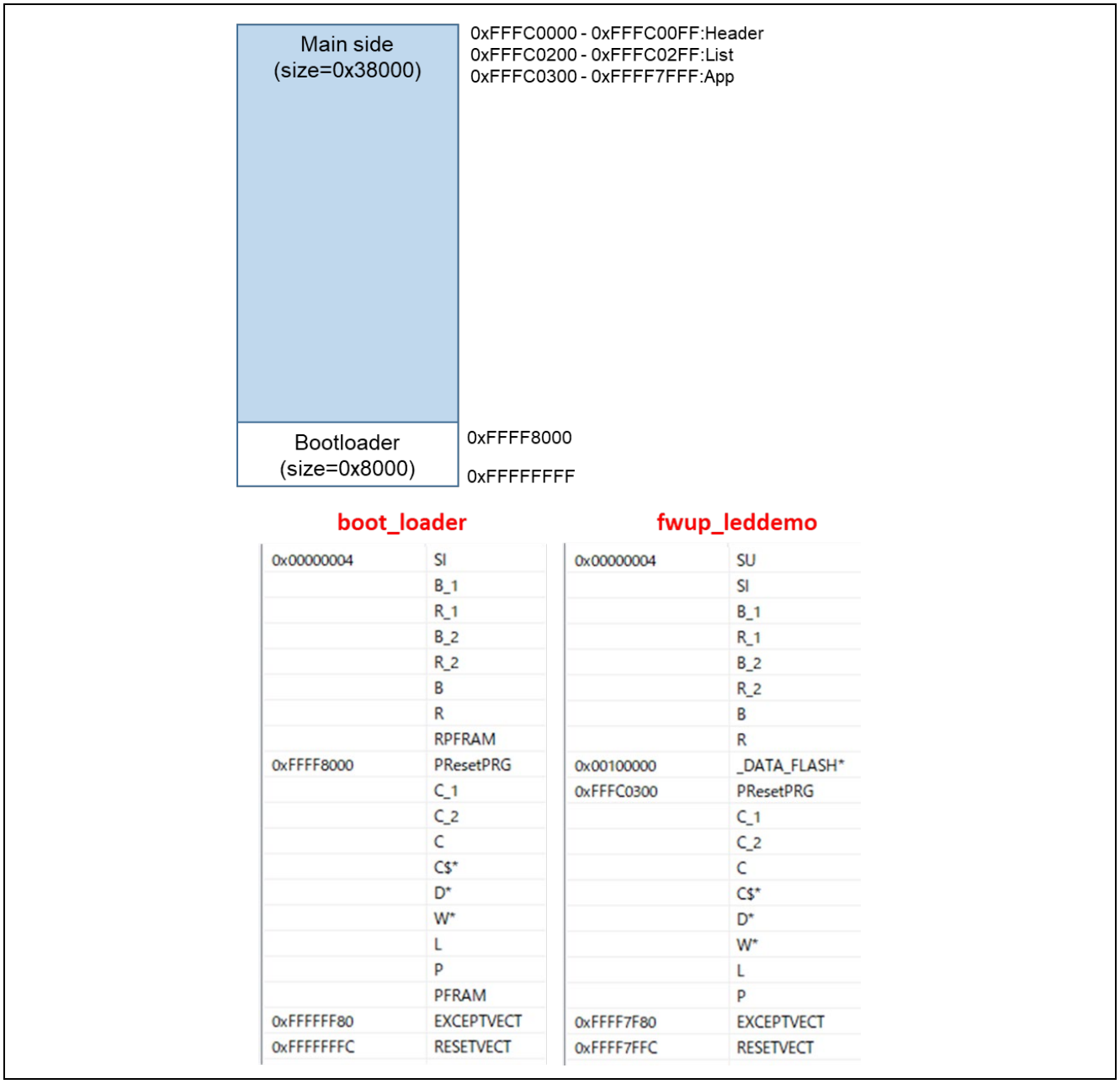
Table 6.9 RX140 linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFDC000	
FWUP_CFG_AREA_SIZE	0x1C000	
FWUP_CFG_CF_BLK_SIZE	0x800	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x100000	
FWUP_CFG_DF_BLK_SIZE	256	
FWUP_CFG_DF_NUM_BLKS	8	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.2.2 Memory map of demo project for full update method in linear mode

The memory map of the RX140 linear mode full update method demo project and the memory map of the configuration settings are shown below.



0x00000004	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
	RPFRAM
0xFFFF8000	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	P
	PFRAM
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
0x00100000	_DATA_FLASH*
0xFFFC0300	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	P
0xFFFF7F80	EXCEPTVECT
0xFFFF7FFC	RESETVECT

Figure 6.8 RX140 linear mode full update method demo project memory map

Table 6.10 RX140 linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF80000
FWUP_CFG_AREA_SIZE	0x78000
FWUP_CFG_CF_BLK_SIZE	0x400
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x100000
FWUP_CFG_DF_BLK_SIZE	1024
FWUP_CFG_DF_NUM_BLKS	8
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.3 Operation Confirmation Environment for RX231

The execution environment and connection diagram are shown below.

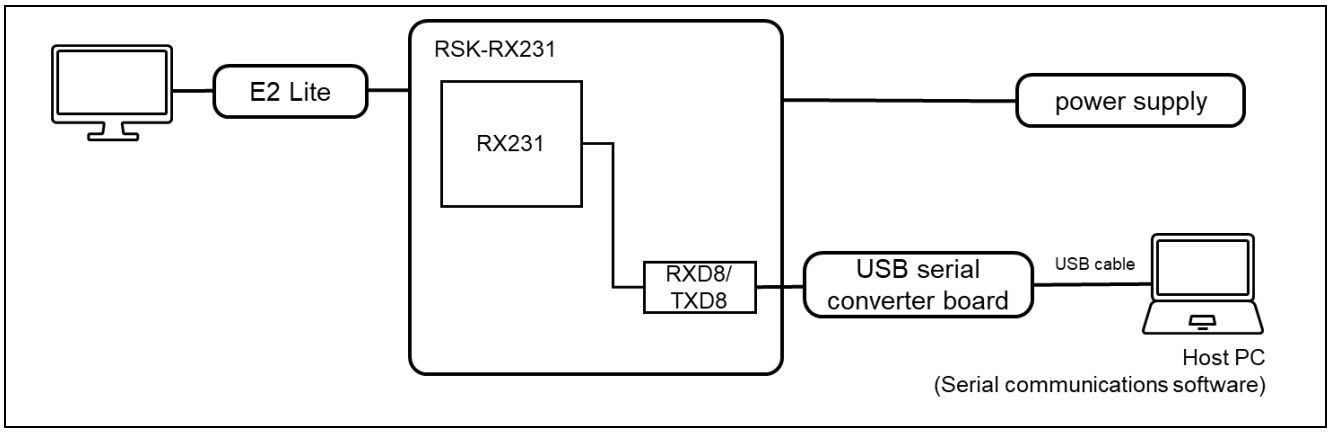


Figure 6.9 RSK-RX231 Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD8	RX
3	RXD8	TX
4	PC5(RTS)	CTS

■ USER_SW (②)

SW1	Note
P31	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P17	LED0

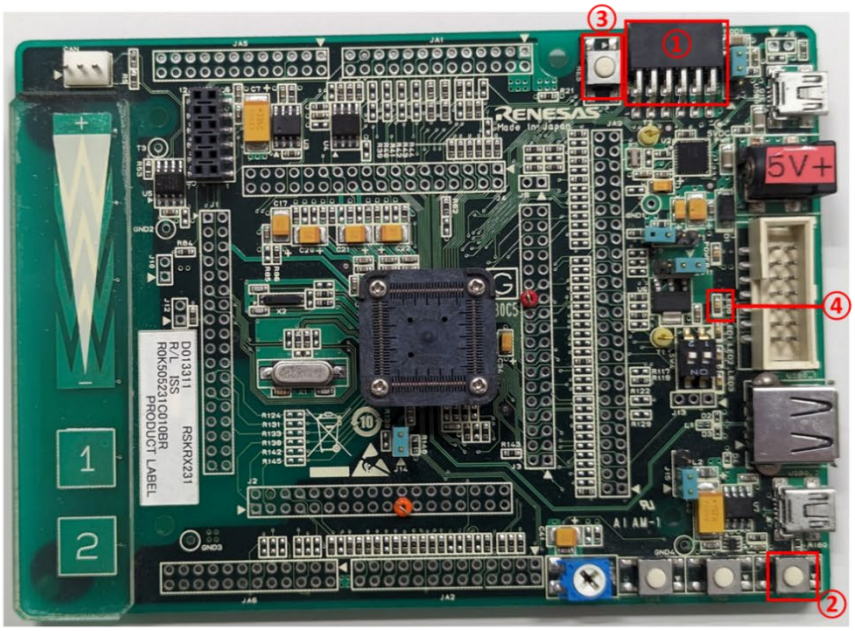


Figure 6.10 RSK-RX231 Pin Information

6.2.3.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX231 linear mode half-surface update method demo project and the memory map of the configuration settings.

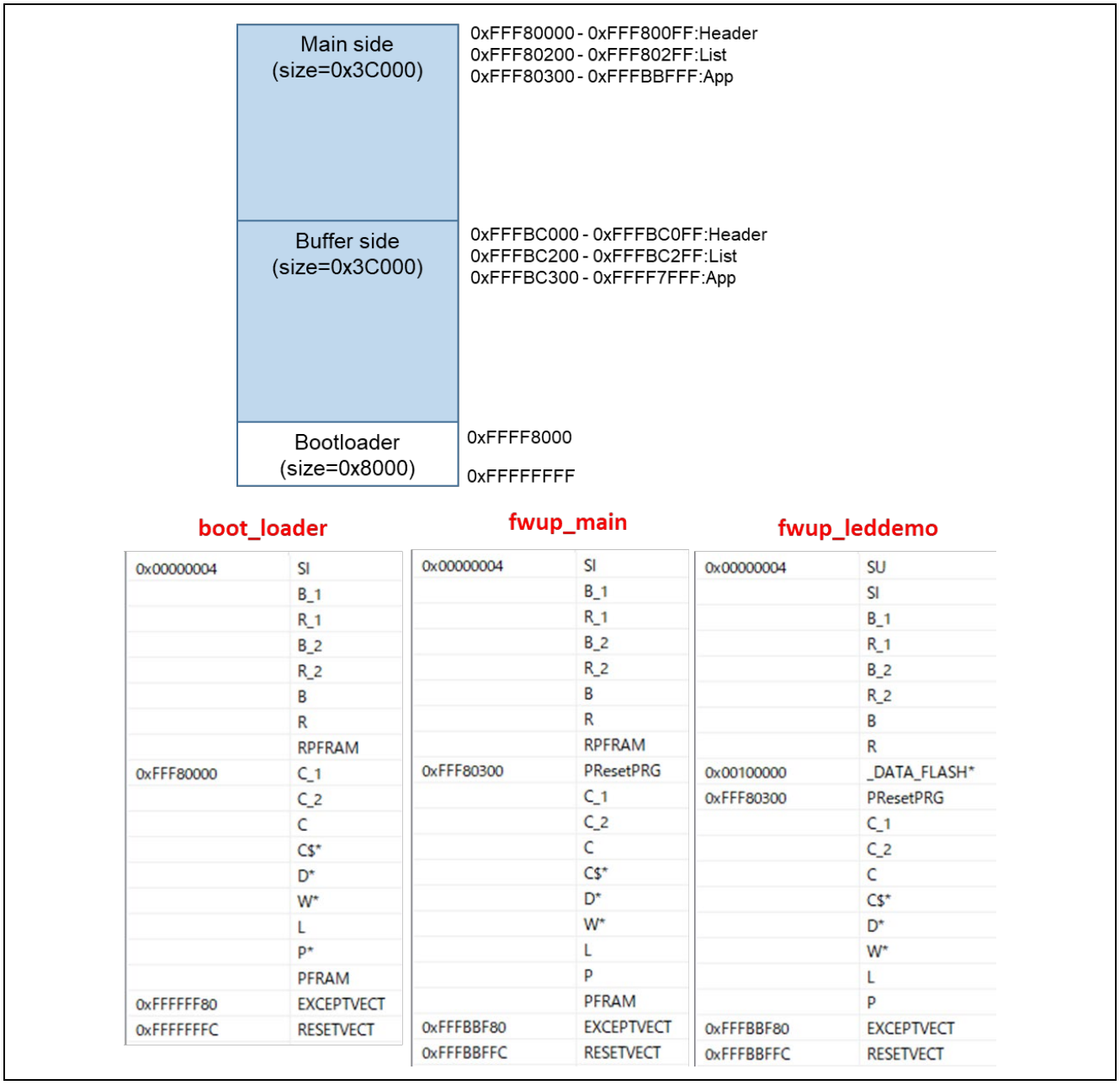


Figure 6.11 RX231 linear mode half-surface update method demo project memory map

Table 6.11 RX231 linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFBC000	
FWUP_CFG_AREA_SIZE	0x3C000	
FWUP_CFG_CF_BLK_SIZE	0x800	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x100000	
FWUP_CFG_DF_BLK_SIZE	1024	
FWUP_CFG_DF_NUM_BLKS	8	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.3.2 Memory map of demo project for full update method in linear mode

The memory map of the RX231 linear mode full update method demo project and the memory map of the configuration settings are shown below.

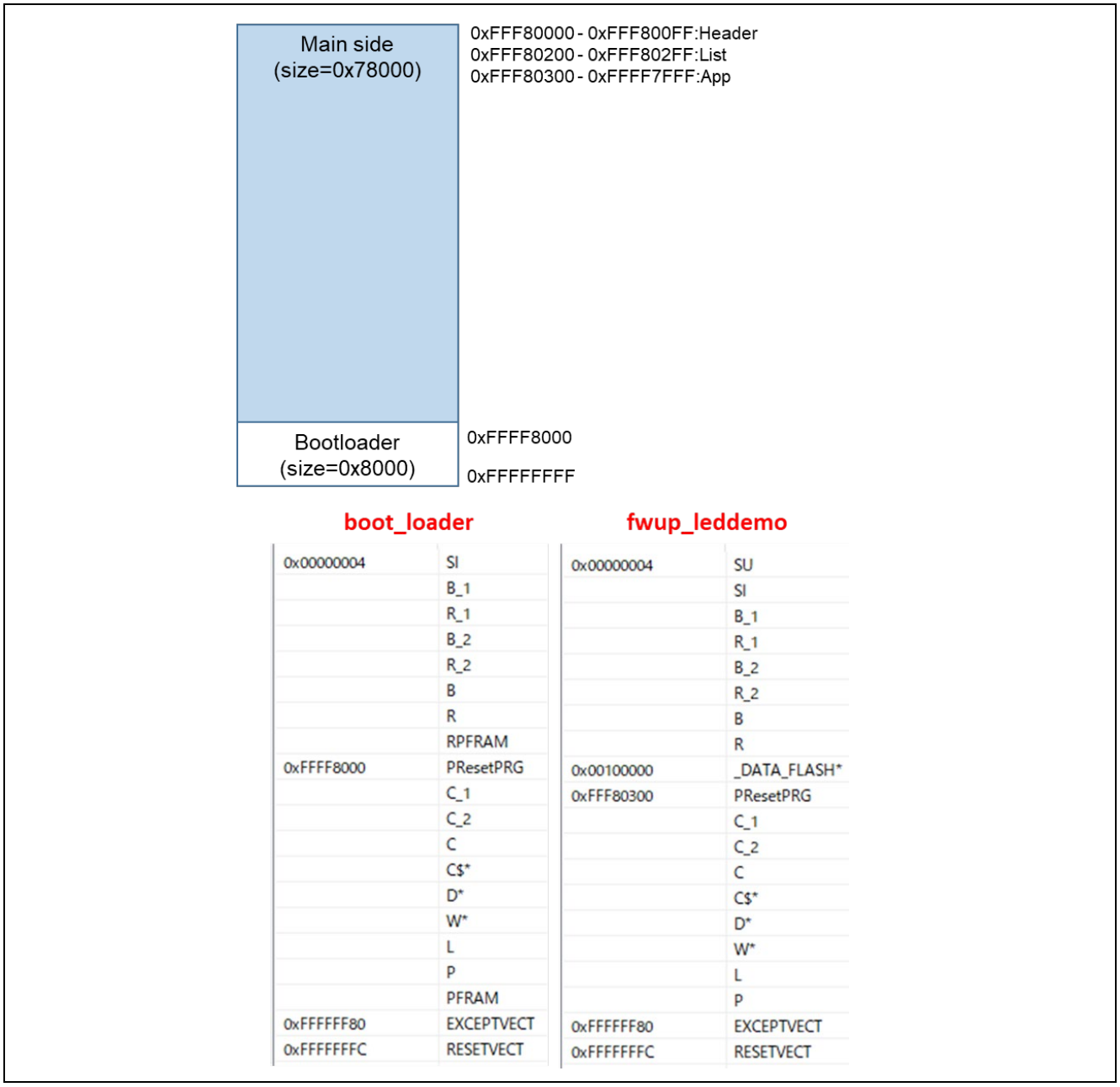


Figure 6.12 RX231 linear mode full update method demo project memory map

Table 6.12 RX231 linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF80000
FWUP_CFG_AREA_SIZE	0x78000
FWUP_CFG_CF_BLK_SIZE	0x800
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x100000
FWUP_CFG_DF_BLK_SIZE	1024
FWUP_CFG_DF_NUM_BLKS	8
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.4 Operation Confirmation Environment for RX23E-A

The execution environment and connection diagram are shown below.

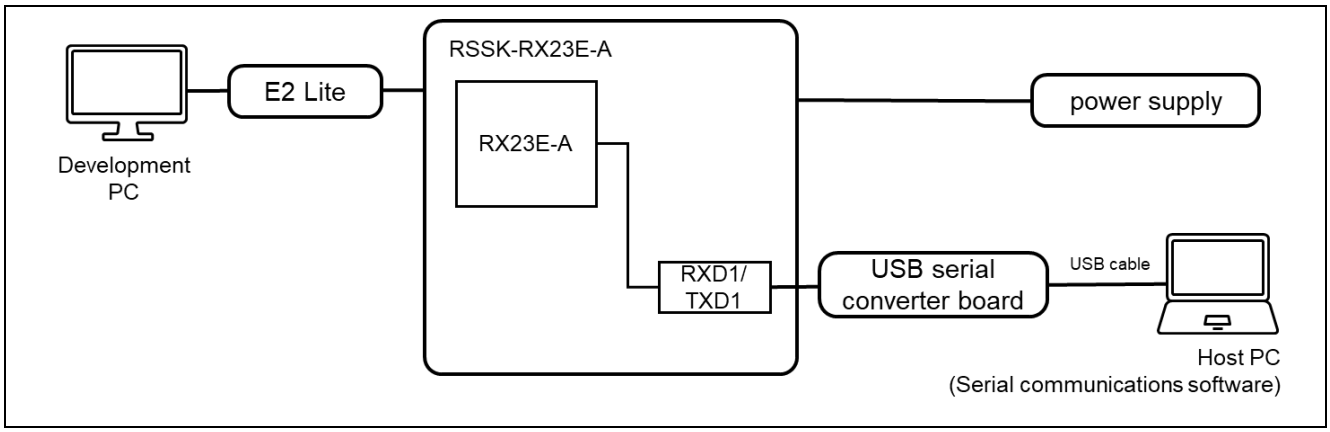


Figure 6.13 RSSK-RX23E-A Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

J4 SCI1 I/F		USB-UART
2	TXD1	RX
4	RXD1	TX
6	CTS1#	CTS

■ USER_SW (②)

SW1	Note
P27	LOW : USER_SW is ON

■ Reset Switch (Blue)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (Yellow)

LED0	Note
PH2	LED

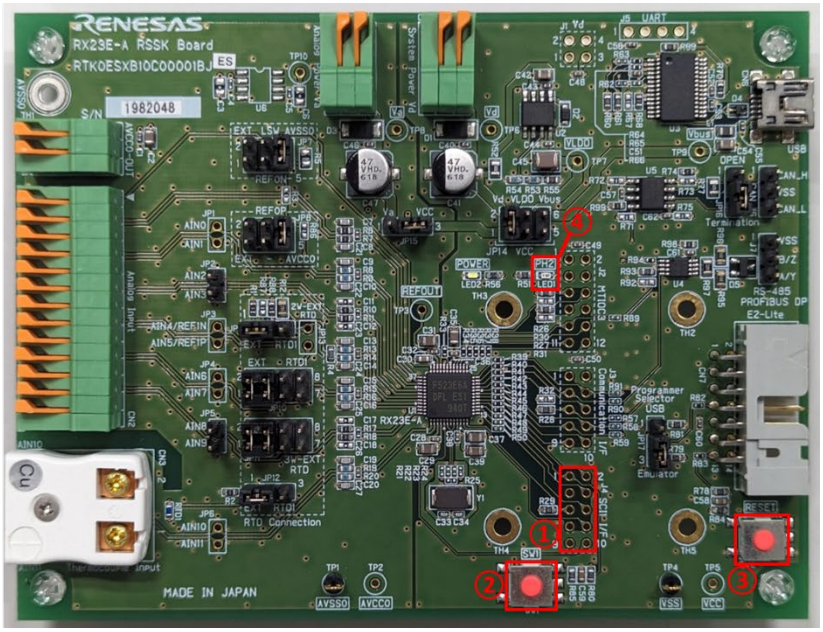


Figure 6.14 RSSK-RX23E-A Pin Information

6.2.4.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX23E-A linear mode half-surface update method demo project and the memory map of the configuration settings.

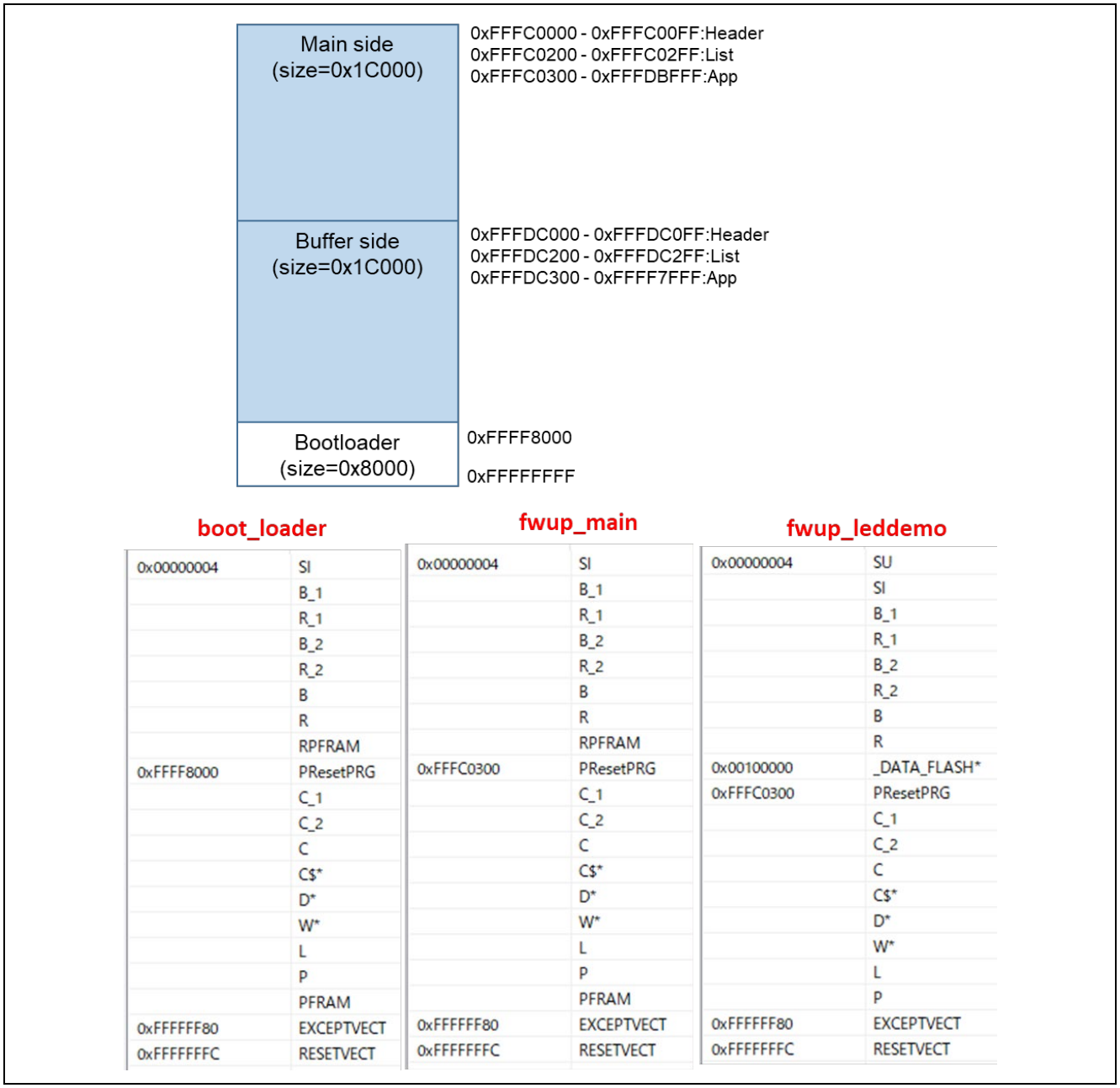


Figure 6.15 RX23E-A linear mode half-surface update method demo project memory map

Table 6.13 RX23E-A linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFDC000	
FWUP_CFG_AREA_SIZE	0x1C000	
FWUP_CFG_CF_BLK_SIZE	0x800	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x100000	
FWUP_CFG_DF_BLK_SIZE	1024	
FWUP_CFG_DF_NUM_BLKS	8	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.4.2 Memory map of demo project for full update method in linear mode

The memory map of the RX23E-A linear mode full update method demo project and the memory map of the configuration settings are shown below.

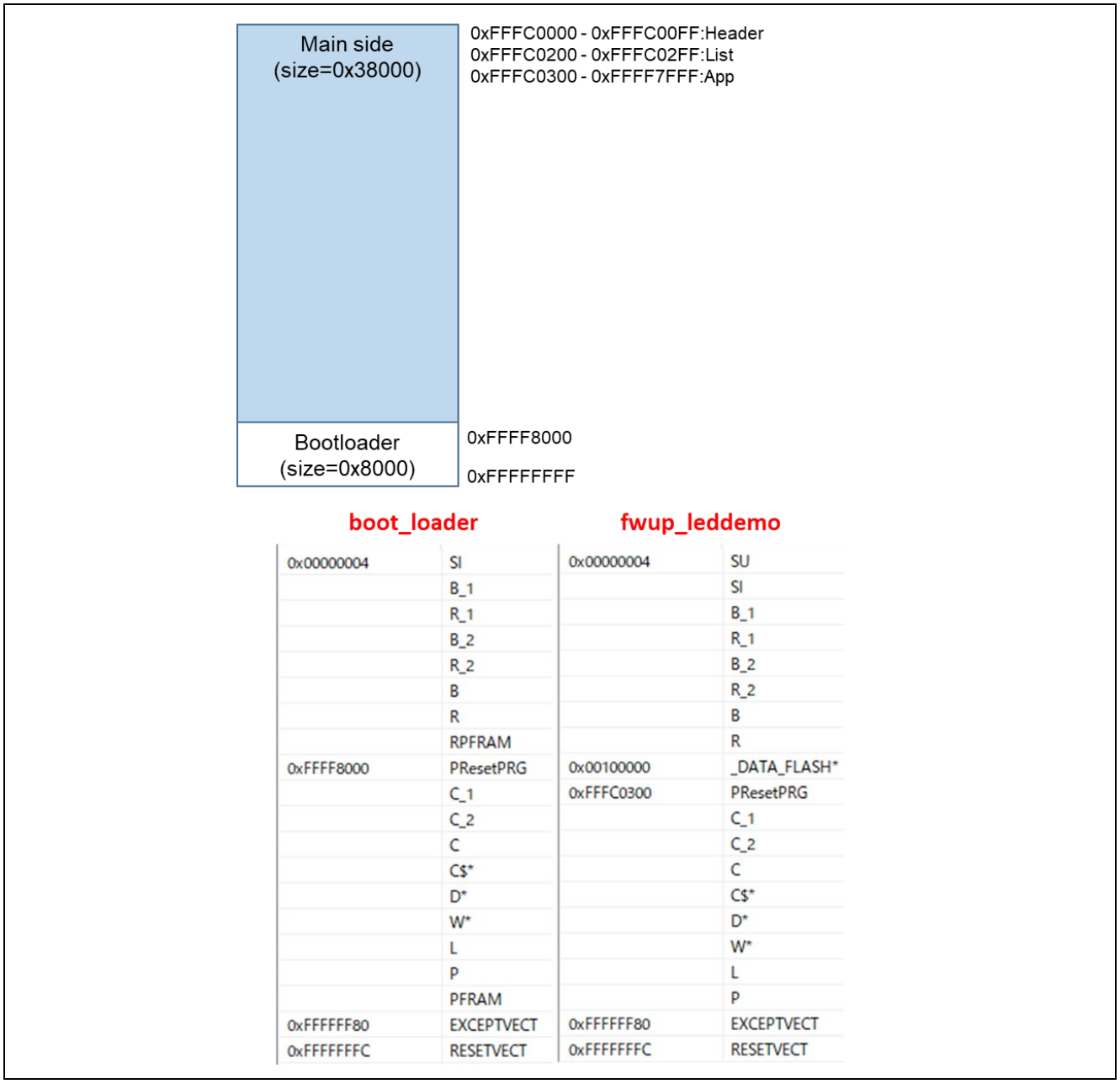


Figure 6.16 RX23E-A linear mode full update method demo project memory map

Table 6.14 RX23E-A linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFC0000
FWUP_CFG_AREA_SIZE	0x38000
FWUP_CFG_CF_BLK_SIZE	0x800
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x100000
FWUP_CFG_DF_BLK_SIZE	1024
FWUP_CFG_DF_NUM_BLKS	8
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.5 Operation Confirmation Environment for RX23E-B

The execution environment and connection diagram are shown below.

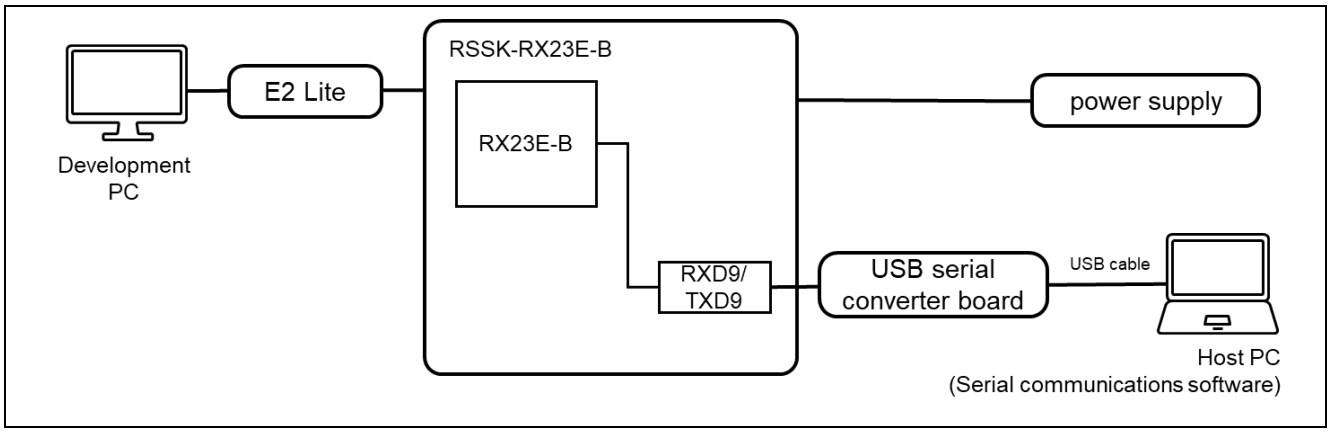


Figure 6.17 RSSK-RX23E-B Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD9	RX
3	RXD9	TX
4	P23(RTS)	CTS

■ USER_SW (②)

SW1	Note
PE1	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P70	LED0

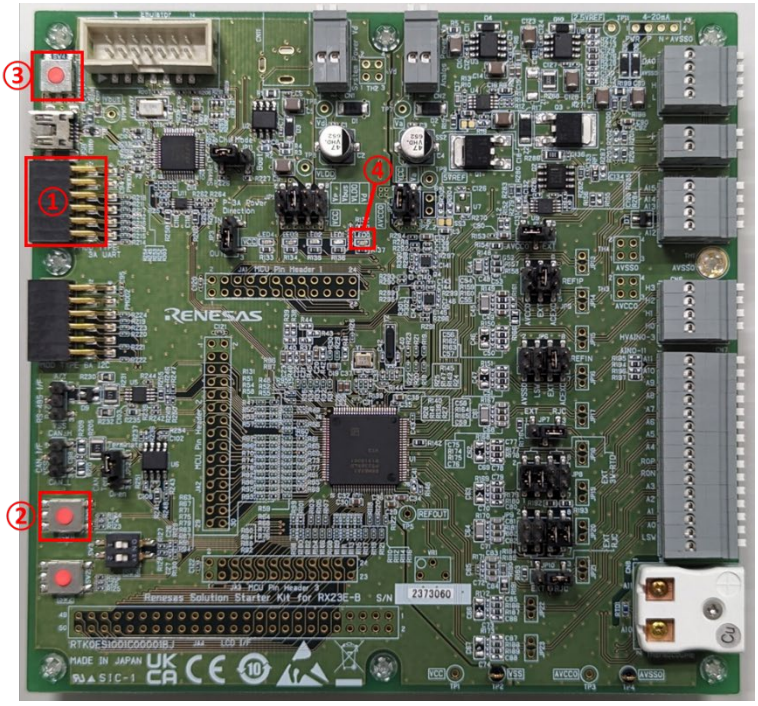


Figure 6.18 RSSK-RX23E-B Pin Information

6.2.5.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX23E-B linear mode half-surface update method demo project and the memory map of the configuration settings.

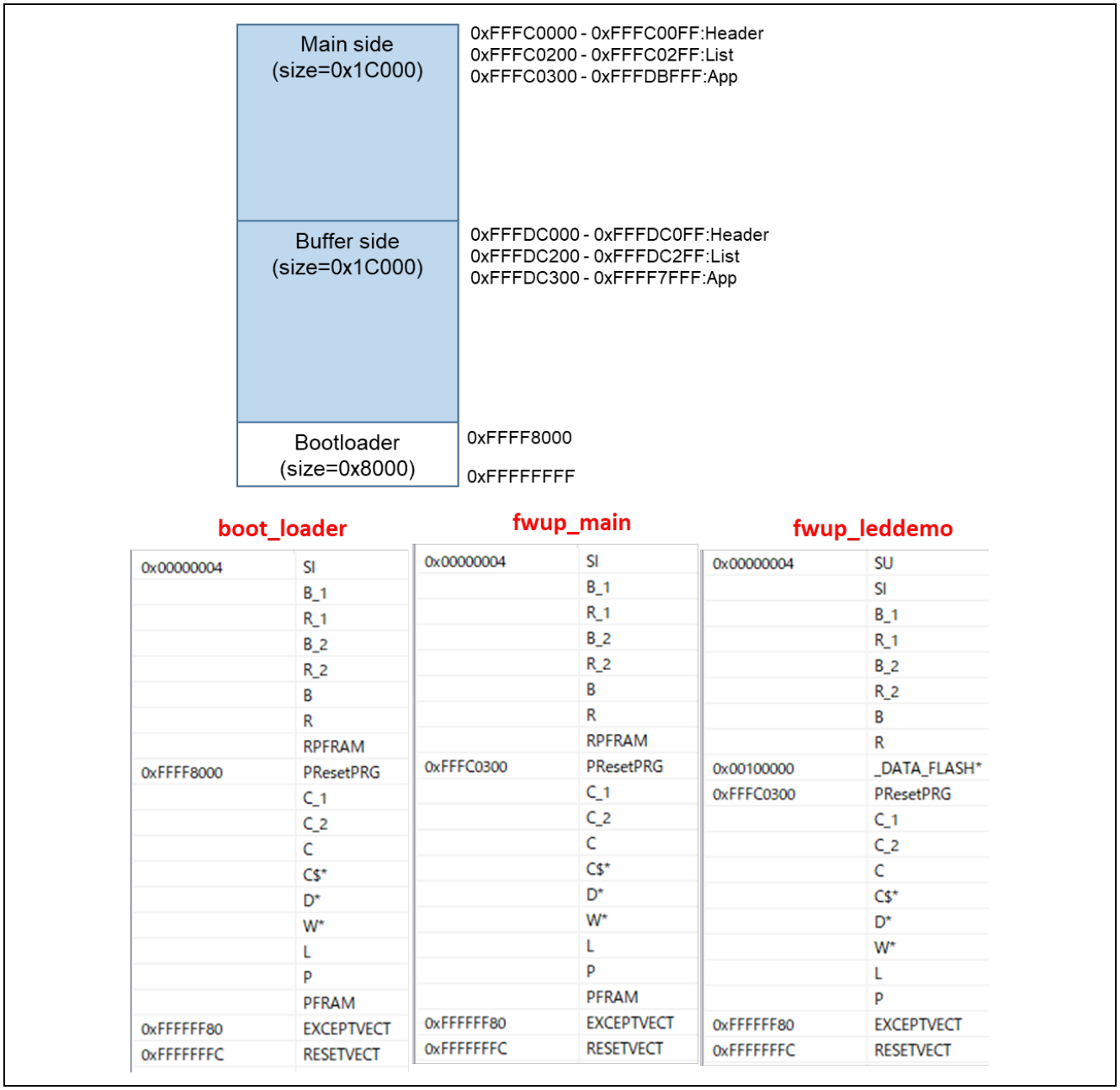


Figure 6.19 RX23E-B linear mode half-surface update method demo project memory map

Table 6.15 RX23E-B linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFDC000	
FWUP_CFG_AREA_SIZE	0x1C000	
FWUP_CFG_CF_BLK_SIZE	0x800	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x100000	
FWUP_CFG_DF_BLK_SIZE	1024	
FWUP_CFG_DF_NUM_BLKS	8	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.5.2 Memory map of demo project for full update method in linear mode

The memory map of the RX23E-B linear mode full update method demo project and the memory map of the configuration settings are shown below.

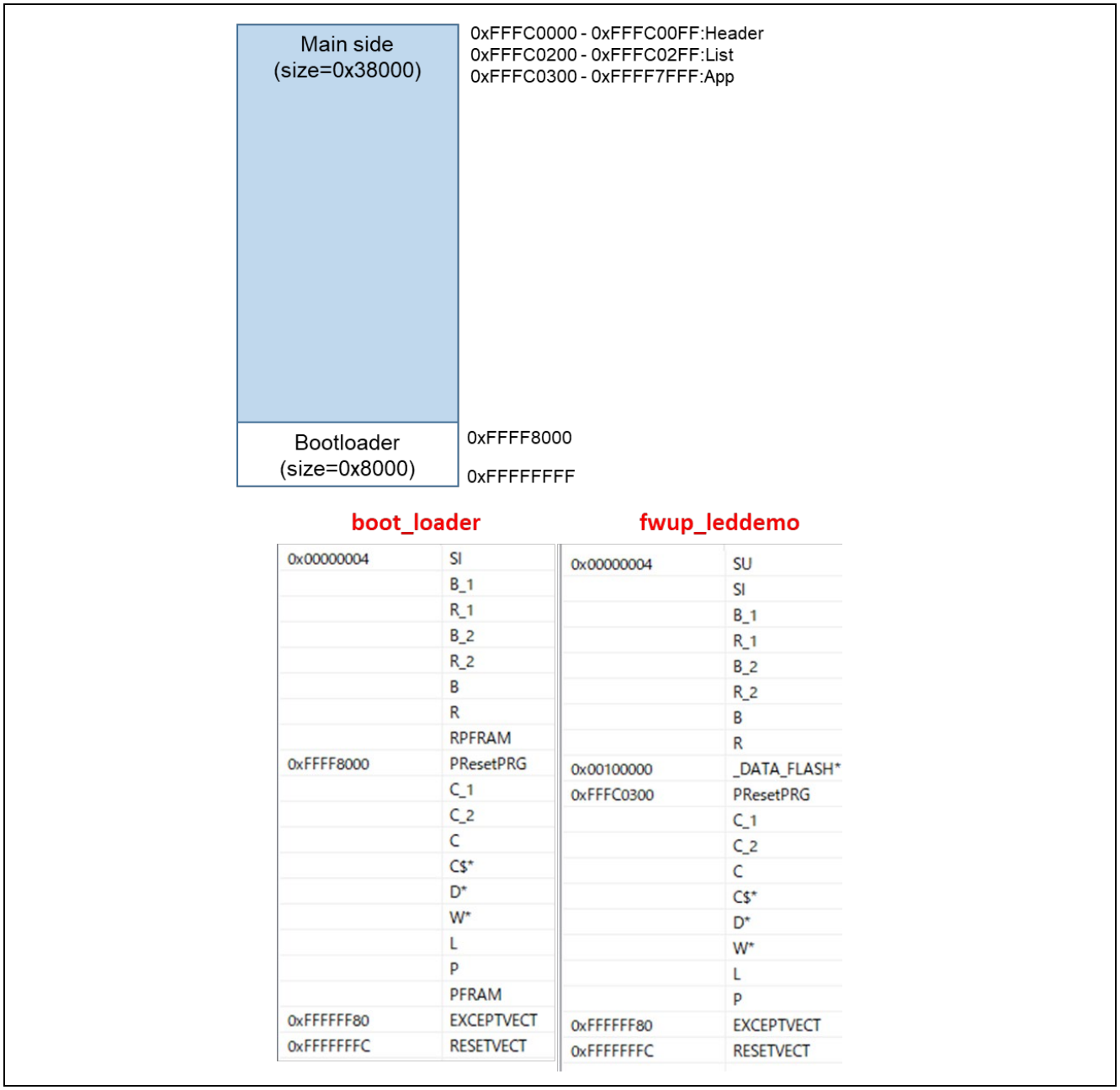


Figure 6.20 RX23E-B linear mode full update method demo project memory map

Table 6.16 RX23E-B linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFC0000
FWUP_CFG_AREA_SIZE	0x38000
FWUP_CFG_CF_BLK_SIZE	0x800
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x100000
FWUP_CFG_DF_BLK_SIZE	1024
FWUP_CFG_DF_NUM_BLKS	8
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.6 Operation Confirmation Environment for RX24T

The execution environment and connection diagram are shown below.

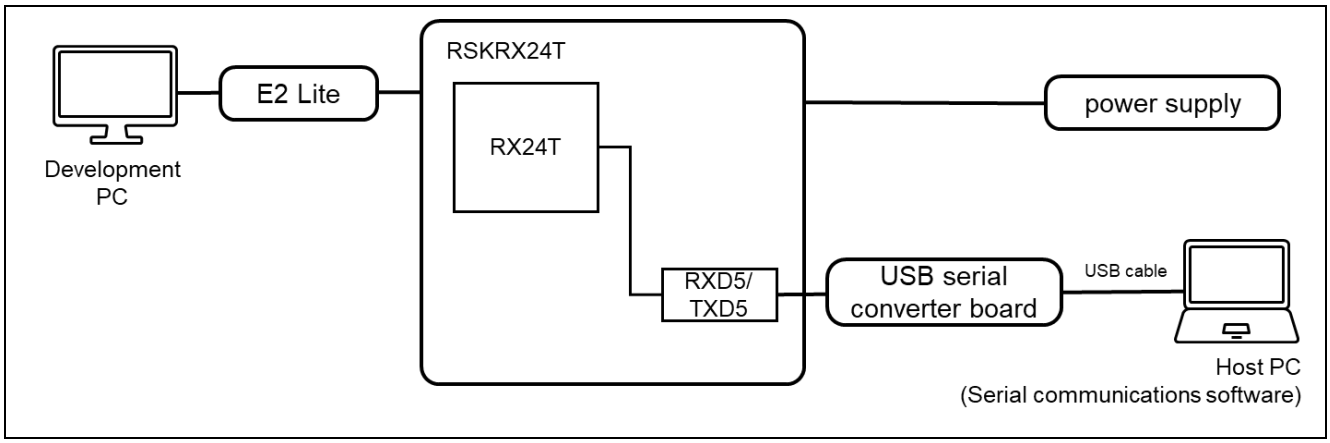


Figure 6.21 RSK-RX24T Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(Red)

PMOD1		USB-UART
2	TXD5	RX
3	RXD5	TX
4	PB7(RTS)	CTS

■ USER_SW (Green)

SW1	Note
P10	LOW : USER_SW is ON

■ Reset Switch (Blue)

RES	Note
RESn	Reset SW for MCU

■ USER_LED (Yellow)

LED0	Note
PB3	LED2

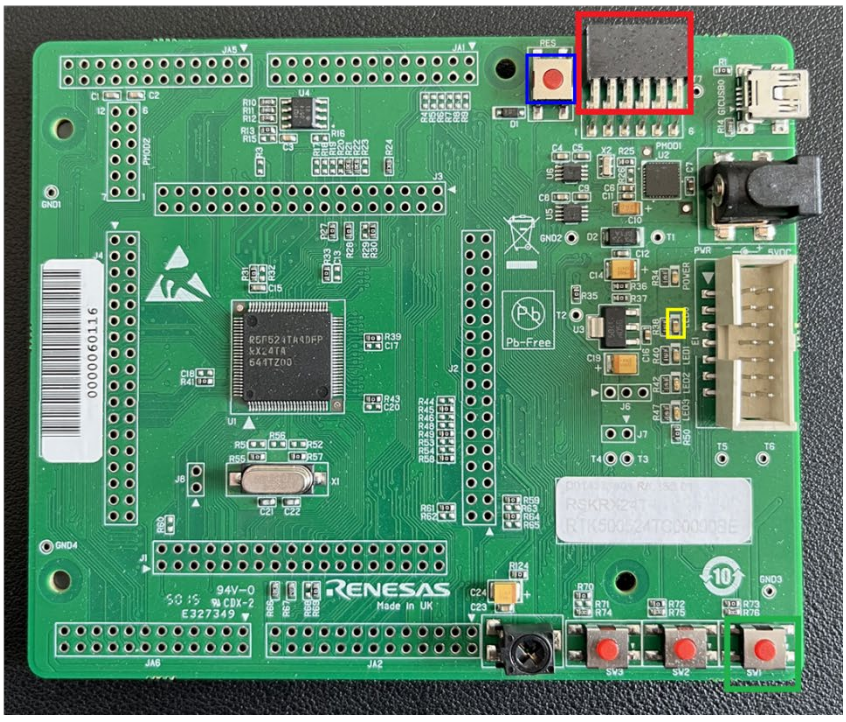
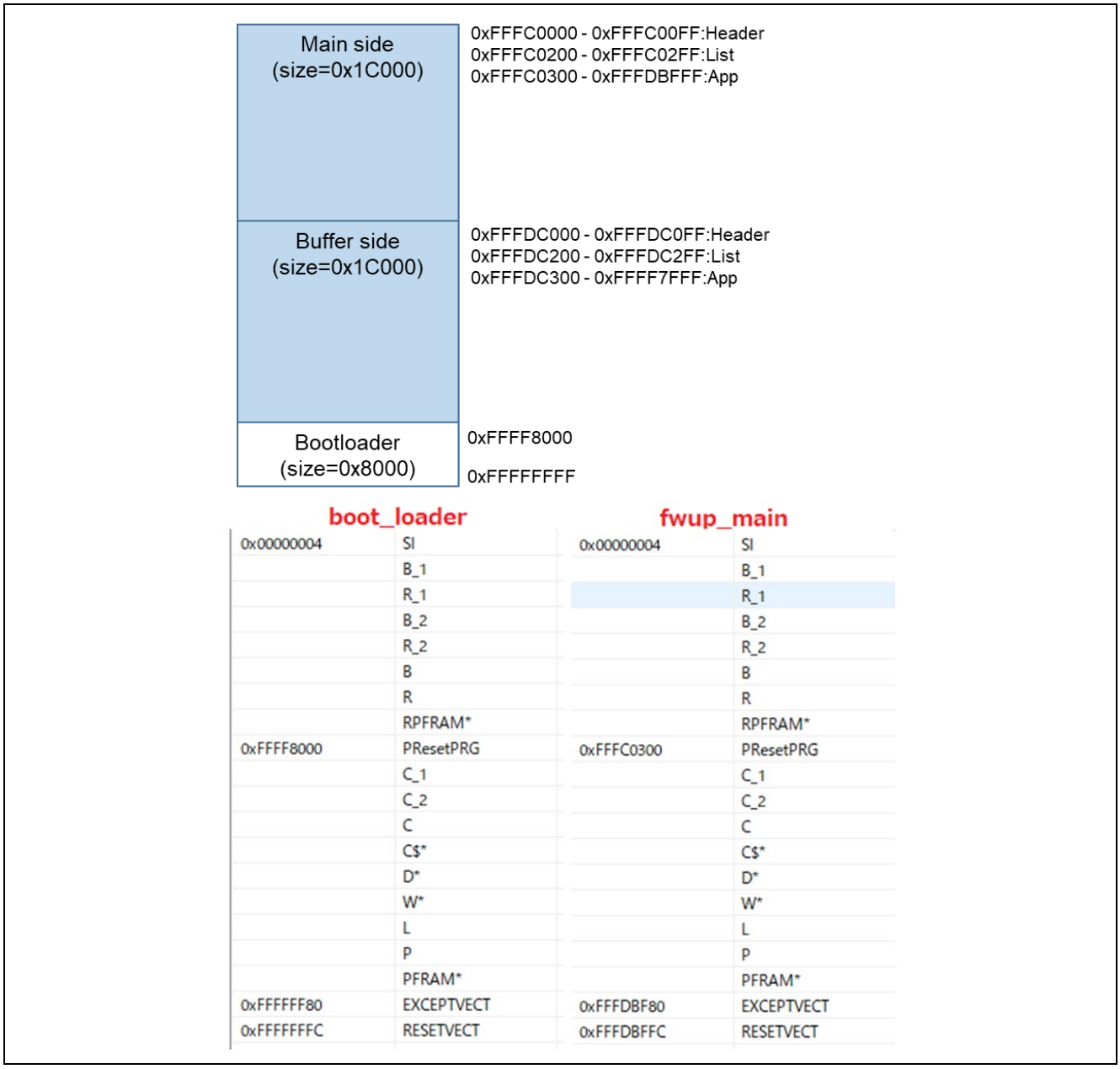


Figure 6.22 RSK-RX24T Pin Information

6.2.6.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX24T linear mode half-surface update method demo project and the memory map of the configuration settings.



boot_loader

0x00000004	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
	RPFRAM*
0xFFFF8000	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	P
	PFRAM*
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

fwup_main

0x00000004	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
	RPFRAM*
0xFFFC0300	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	P
	PFRAM*
0xFFFDBF80	EXCEPTVECT
0xFFFDBFFC	RESETVECT

Figure 6.23 RX24T linear mode half-surface update method demo project memory map

Table 6.17 RX24T linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFDC000	
FWUP_CFG_AREA_SIZE	0x1C000	
FWUP_CFG_CF_BLK_SIZE	0x800	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x100000	
FWUP_CFG_DF_BLK_SIZE	1024	
FWUP_CFG_DF_NUM_BLKS	8	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.6.2 Memory map of demo project for full update method in linear mode

The memory map of the RX24T linear mode full update method demo project and the memory map of the configuration settings are shown below.

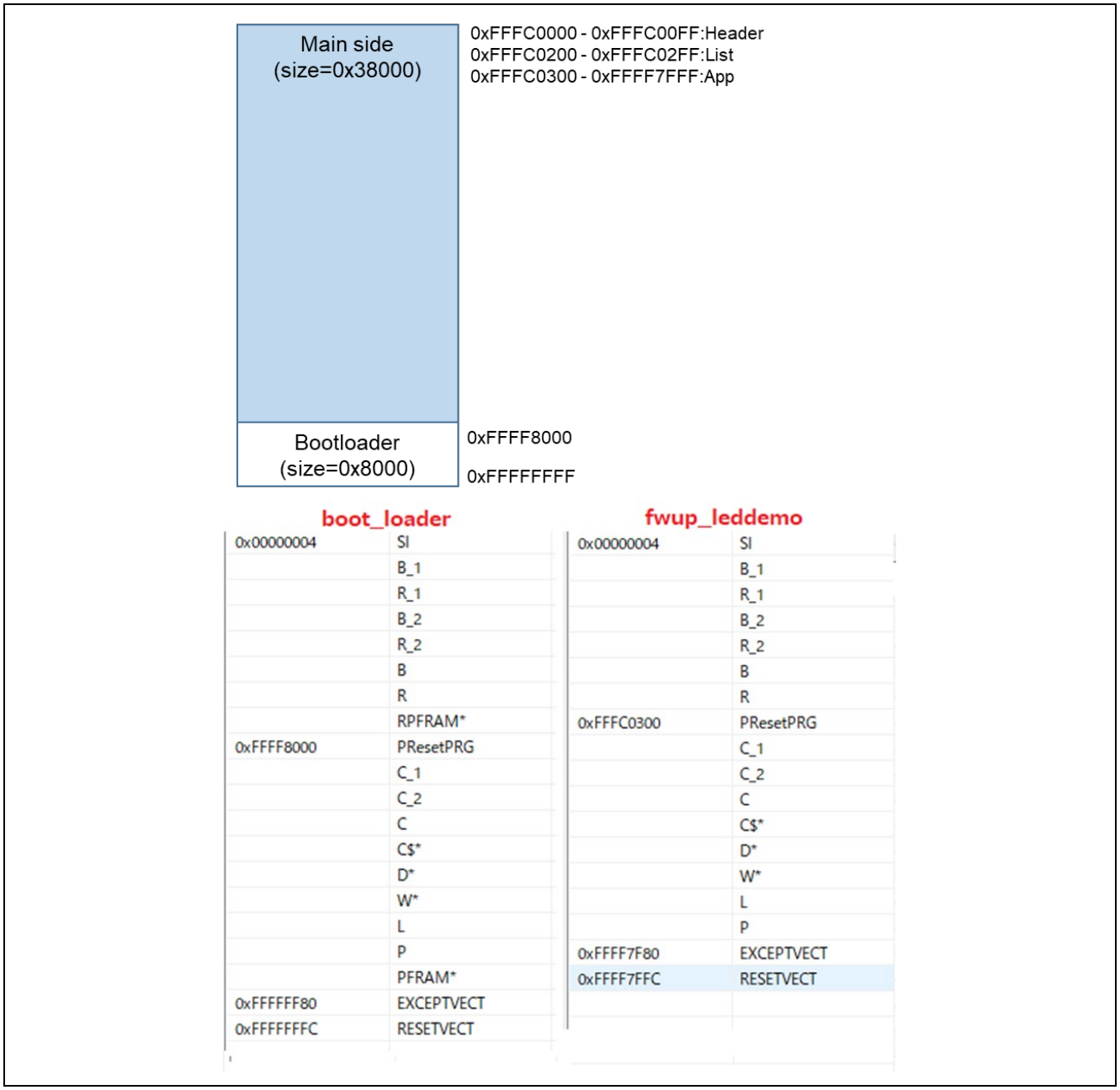


Figure 6.24 RX24T linear mode full update method demo project memory map

Table 6.18 RX24T linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFC0000
FWUP_CFG_AREA_SIZE	0x38000
FWUP_CFG_CF_BLK_SIZE	0x800
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x100000
FWUP_CFG_DF_BLK_SIZE	1024
FWUP_CFG_DF_NUM_BLKS	8
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.7 Operation Confirmation Environment for RX26T

The execution environment and connection diagram are shown below.

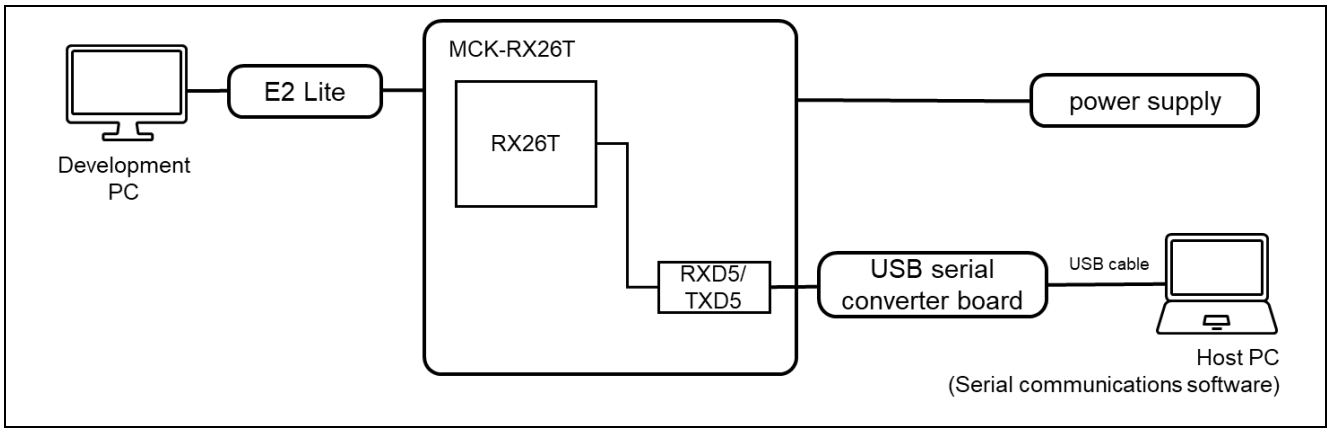


Figure 6.25 MCK-RX26T Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD2		USB-UART
2	TXD5	RX
3	RXD5	TX
4	PB0(RTS)	CTS

■ USER_SW (②) ※1

PMOD1		Note
1	PB3	LOW : USER_SW is ON
5	GND	Connect to PMOD1. 1

■ Reset Switch (③)

R92	Note
RES#	Reset SW for MCU

■ USER_LED (④)

LED0	Note
P20	LED2

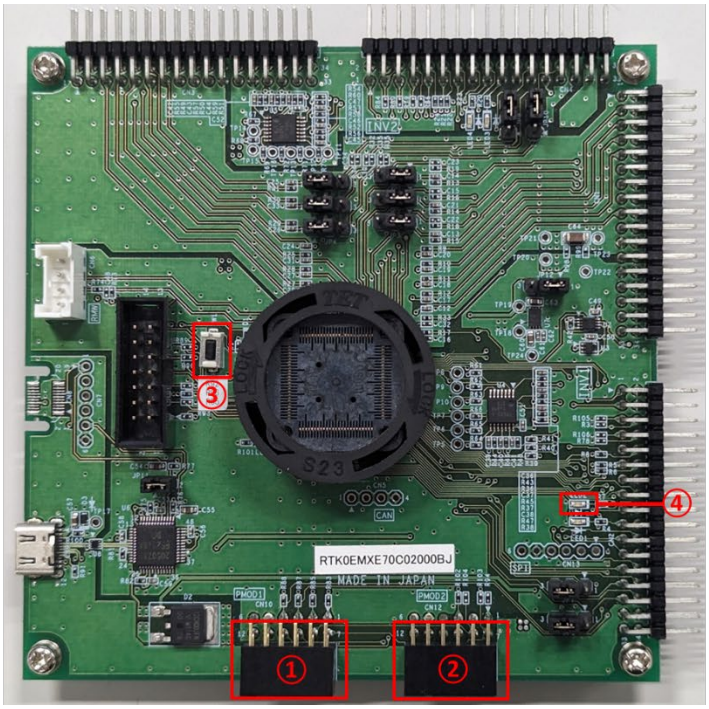


Figure 6.26 MCK-RX26T Pin Information

(*1) The connection of USER_SW (2) with external SW is as follows.

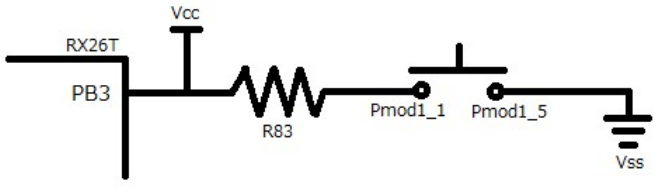


Figure 6.27 MCK-RX26T external connection SW

6.2.7.1 Memory map of dual bank method demo project

The memory map and configuration settings for the RX26T dual-bank method demo project are shown below.

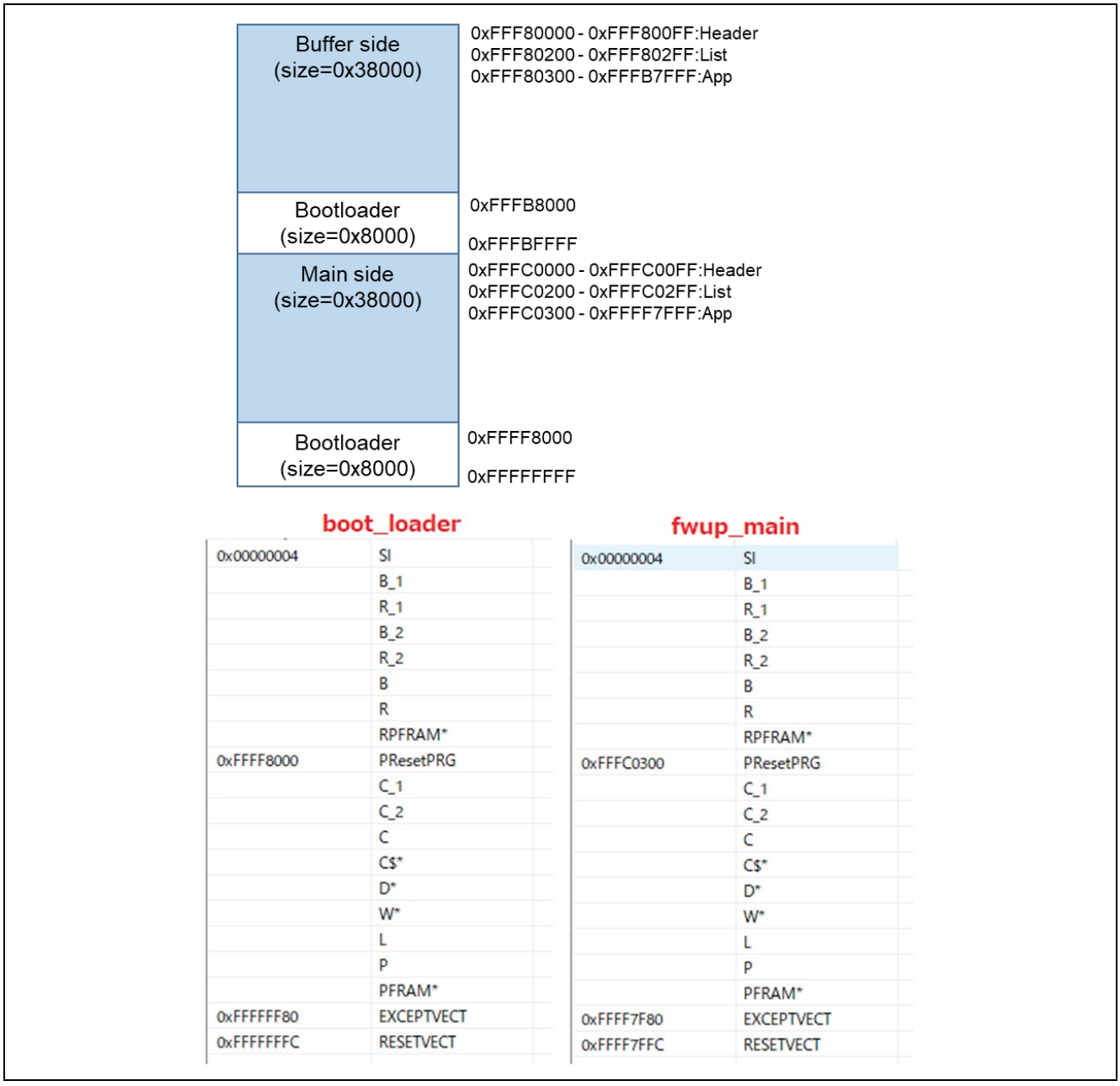


Figure 6.28 RX26T dual bank method demo project memory map

Table 6.19 RX26T dual bank method configuration settings

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	0	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFFC0000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF80000	
FWUP_CFG_AREA_SIZE	0x38000	
FWUP_CFG_CF_BLK_SIZE	0x4000	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	256	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	1	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	1	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	1	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	1	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	1	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	1	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.7.2 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX26T linear mode half-surface update method demo project and the memory map of the configuration settings.

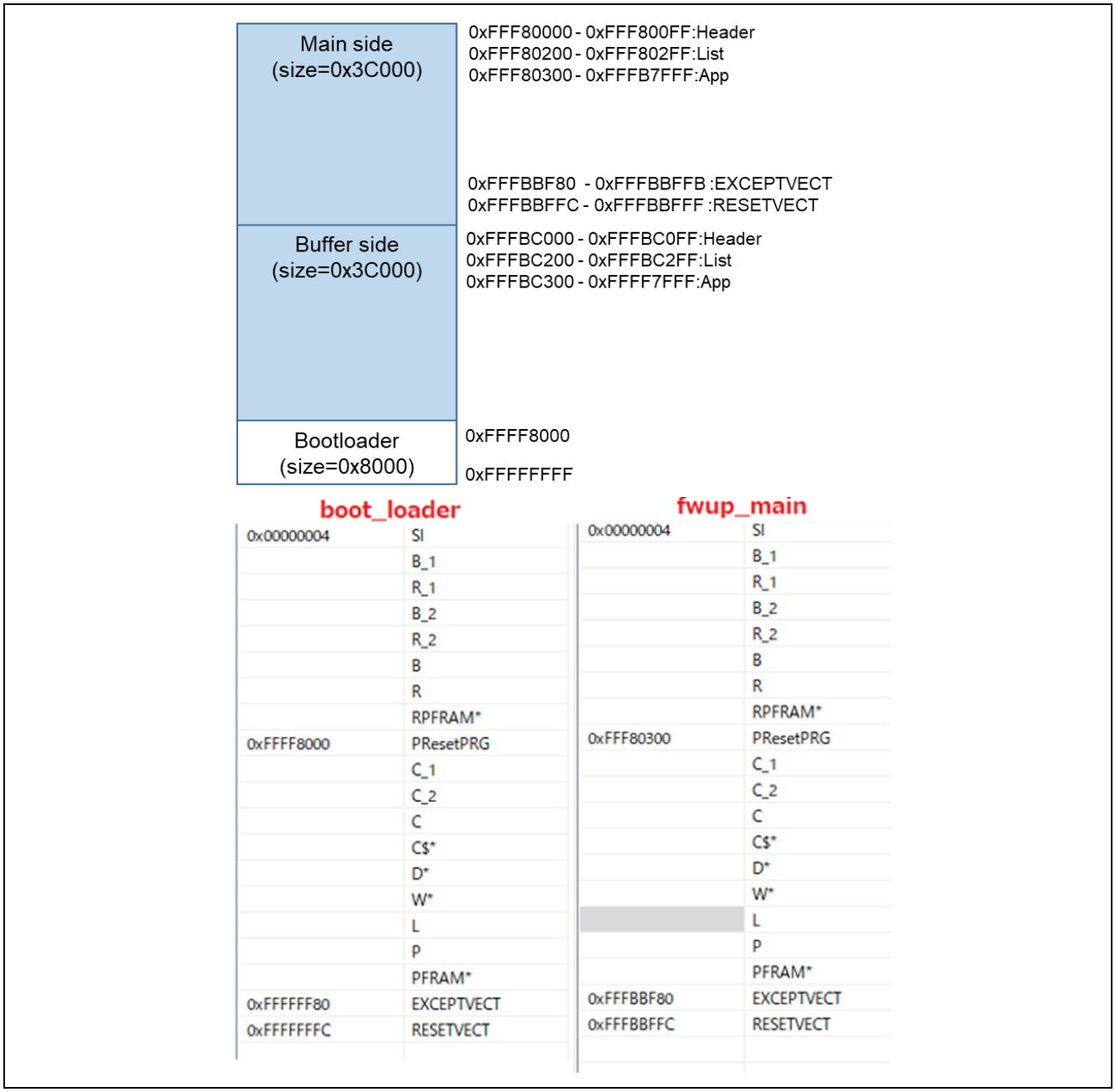


Figure 6.29 RX26T linear mode half-surface update method demo project memory map

Table 6.20 RX26T linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFBC000	
FWUP_CFG_AREA_SIZE	0x3C000	
FWUP_CFG_CF_BLK_SIZE	0x4000	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	256	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.7.3 Memory map of demo project for full update method in linear mode

The memory map of the RX26T linear mode full update method demo project and the memory map of the configuration settings are shown below.

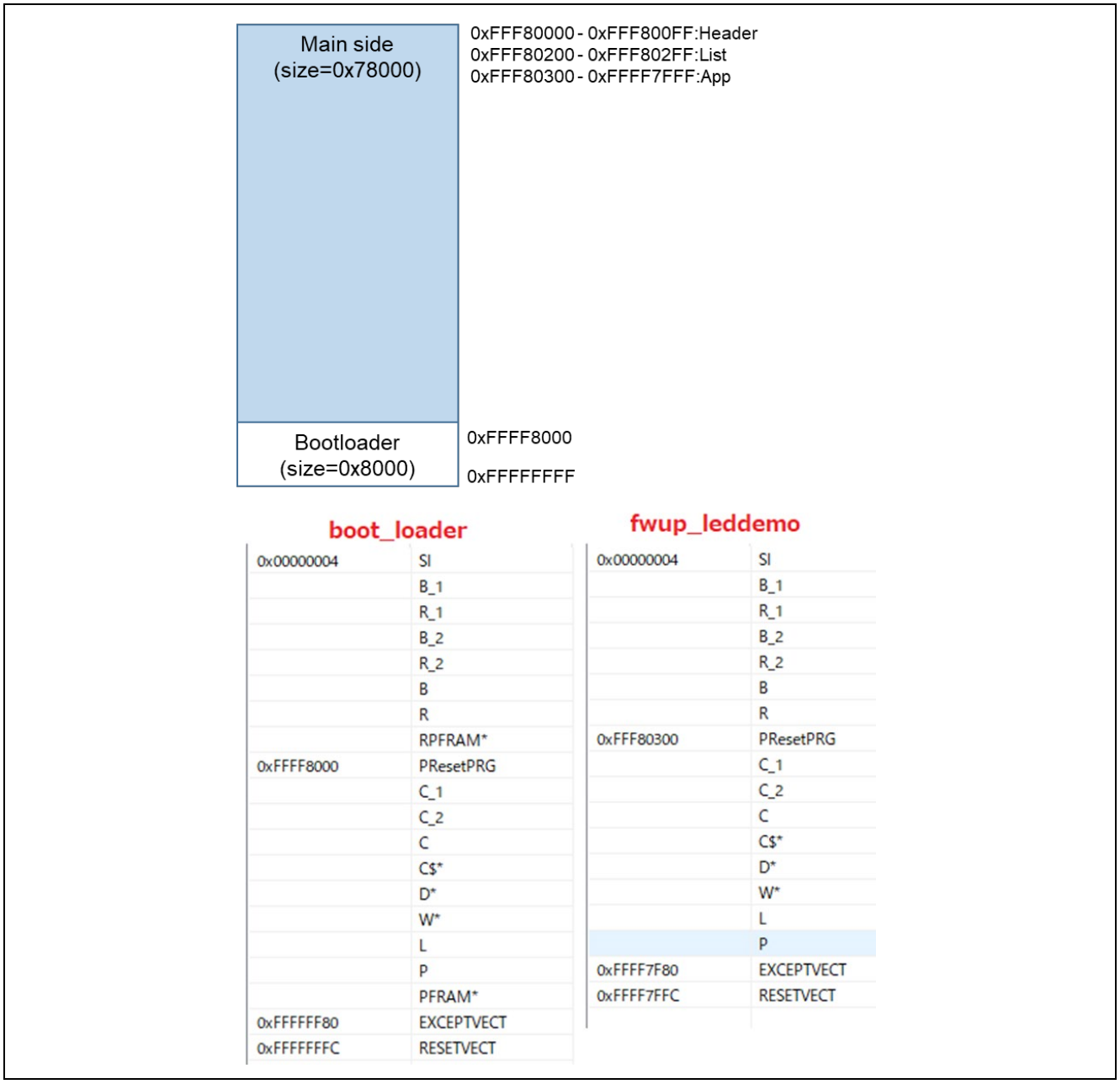


Figure 6.30 RX26T linear mode full update method demo project memory map

Table 6.21 RX26T linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF80000
FWUP_CFG_AREA_SIZE	0x78000
FWUP_CFG_CF_BLK_SIZE	0x4000
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x00100000
FWUP_CFG_DF_BLK_SIZE	64
FWUP_CFG_DF_NUM_BLKS	256
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.8 Operation Confirmation Environment for RX65N

The execution environment and connection diagram are shown below.

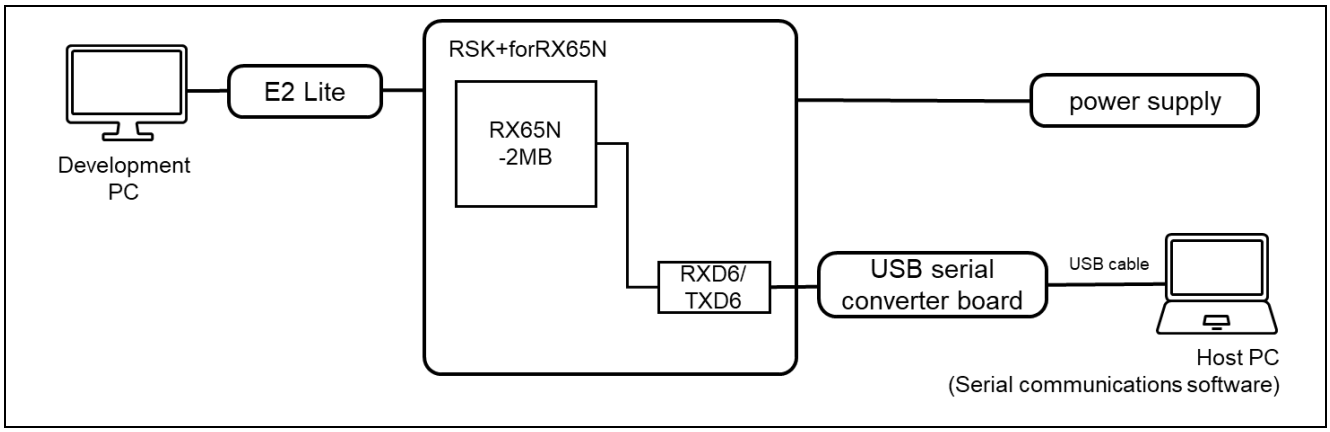


Figure 6.31 RSK-RX65N Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD6	RX
3	RXD6	TX
4	P02(RTS)	CTS

■ USER_SW (②)

SW1	Note
P03	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P73	LED0

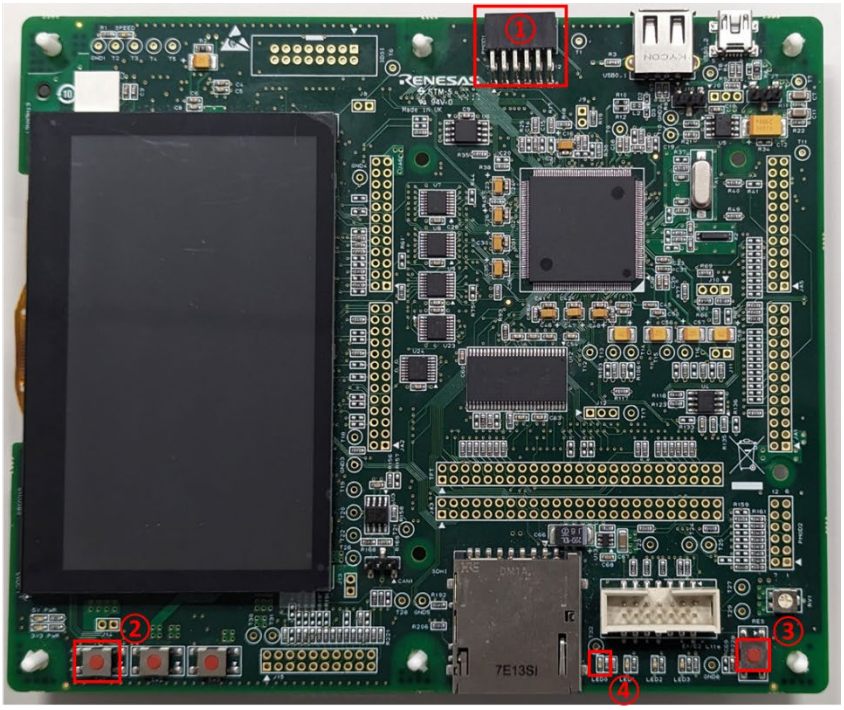


Figure 6.32 RSK-RX65N Pin Information

6.2.8.1 Memory map of dual bank method demo project

The memory map and configuration settings for the RX65N dual-bank method demo project are shown below.

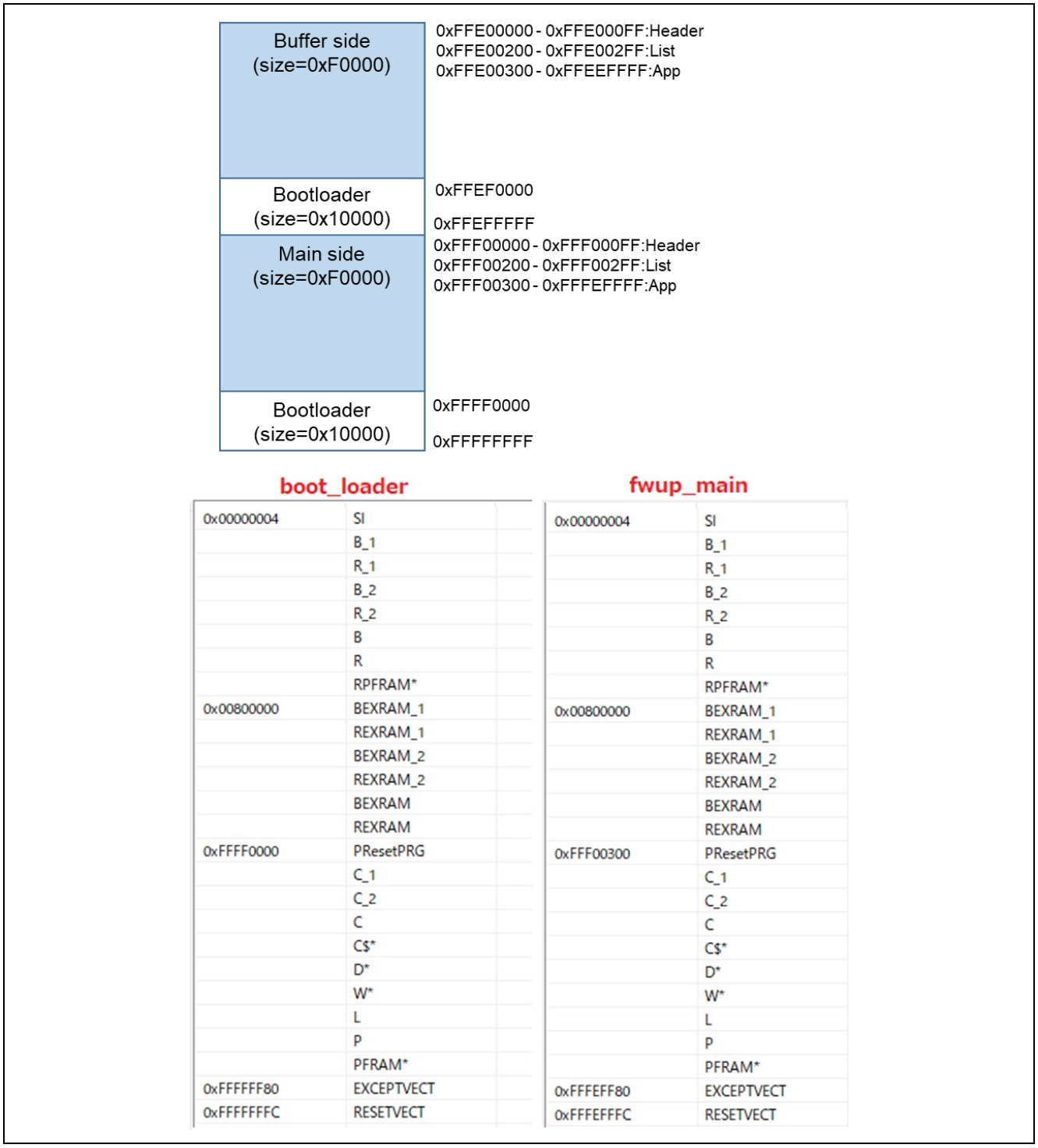


Figure 6.33 RX65N dual bank method demo project memory map

Table 6.22 RX65N dual bank method configuration settings

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	0	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF00000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFE00000	
FWUP_CFG_AREA_SIZE	0xF0000	
FWUP_CFG_CF_BLK_SIZE	0x8000	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	512	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	1	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	1	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	1	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	1	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	1	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	1	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.8.2 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX65N linear mode half-surface update method demo project and the memory map of the configuration settings.

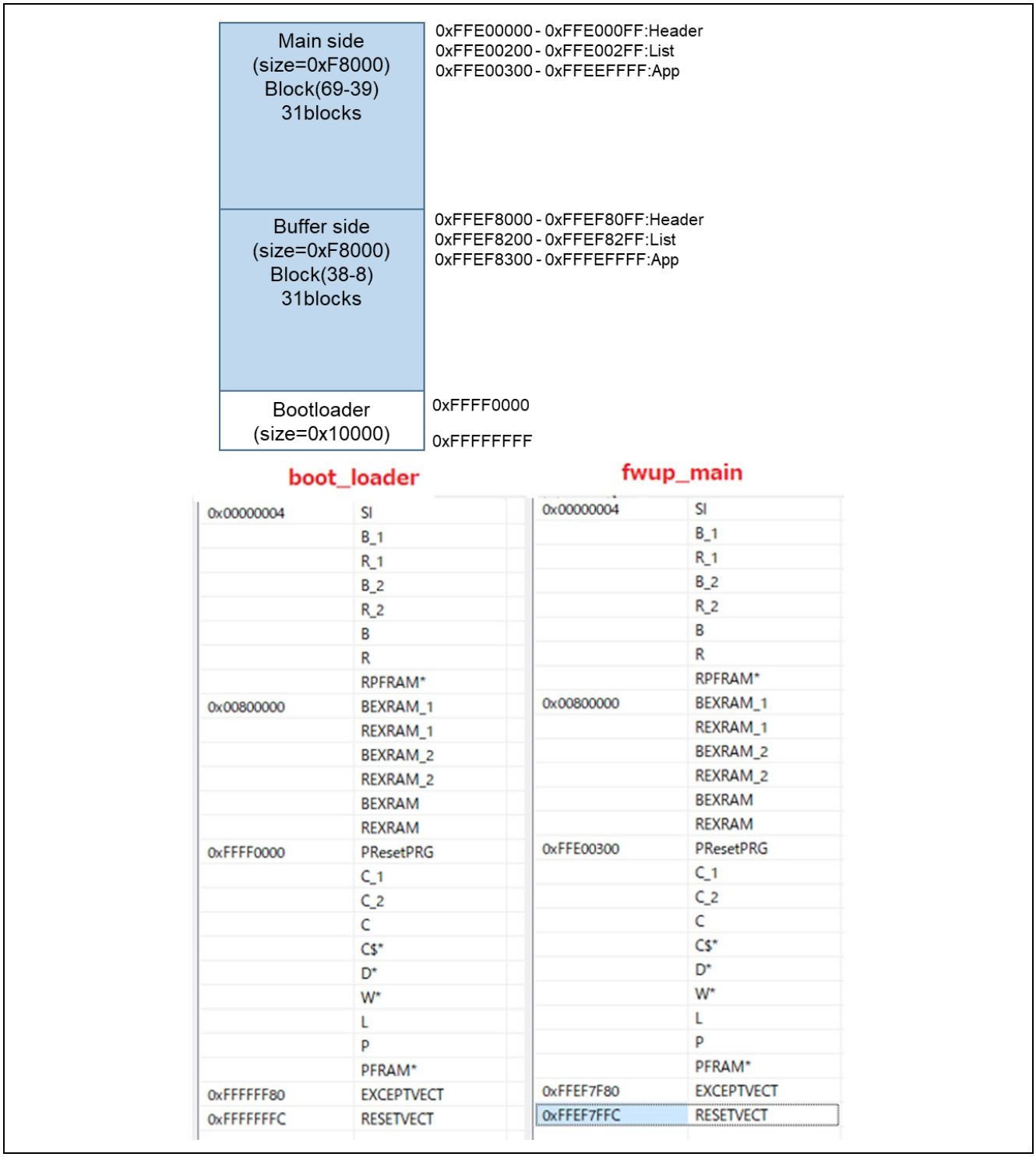


Figure 6.34 RX65N linear mode half-surface update method demo project memory map

Table 6.23 RX65N linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFE00000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFEF8000	
FWUP_CFG_AREA_SIZE	0xF8000	
FWUP_CFG_CF_BLK_SIZE	0x8000	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	512	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.8.3 Memory map of demo project for full update method in linear mode

The memory map of the RX65N linear mode full update method demo project and the memory map of the configuration settings are shown below.

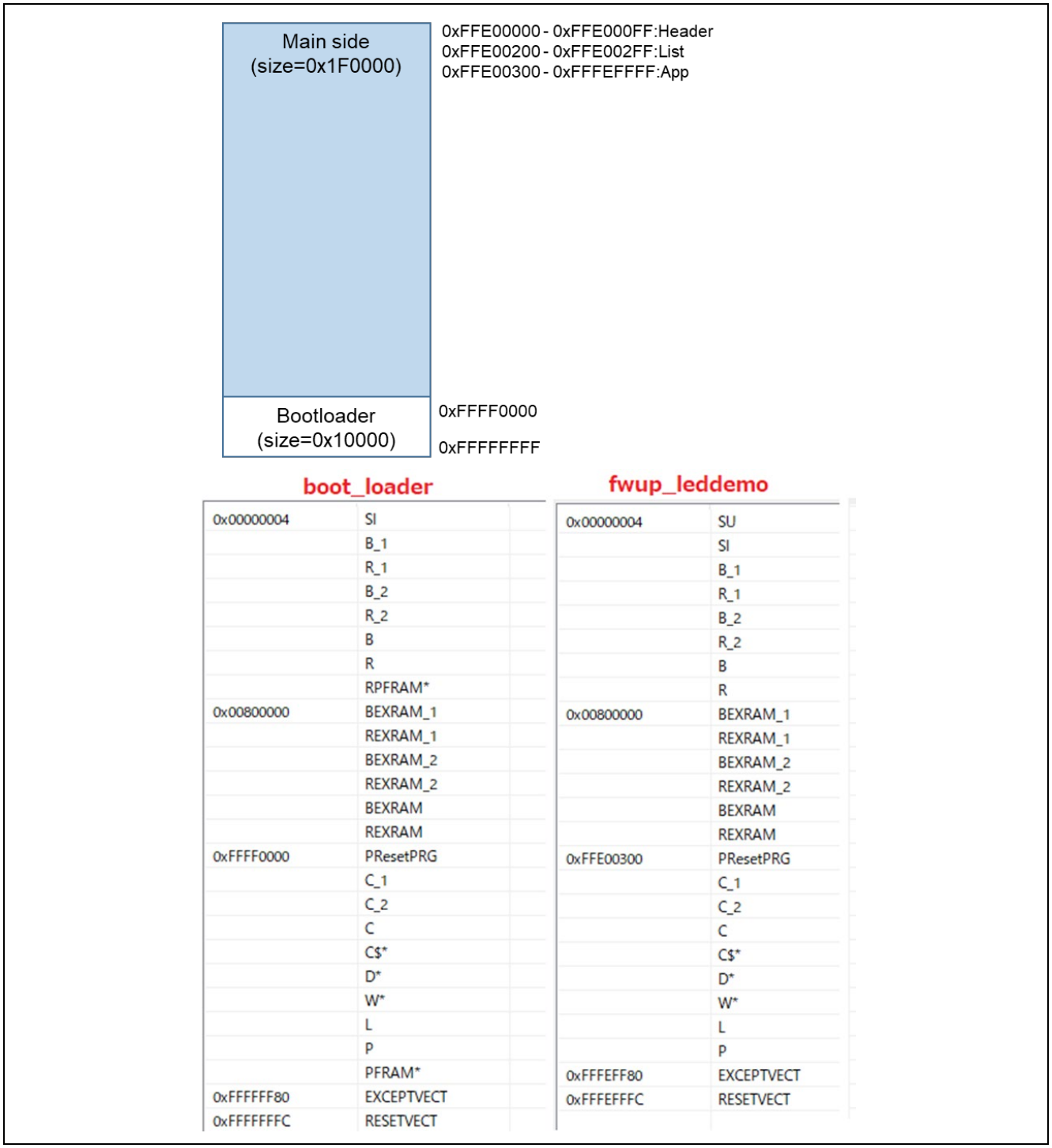


Figure 6.35 RX65N linear mode full update method demo project memory map

Table 6.24 RX65N linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFE00000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFE00000
FWUP_CFG_AREA_SIZE	0x1F0000
FWUP_CFG_CF_BLK_SIZE	0x8000
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x00100000
FWUP_CFG_DF_BLK_SIZE	64
FWUP_CFG_DF_NUM_BLKS	512
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.9 Operation Confirmation Environment for RX66T

The execution environment and connection diagram are shown below.

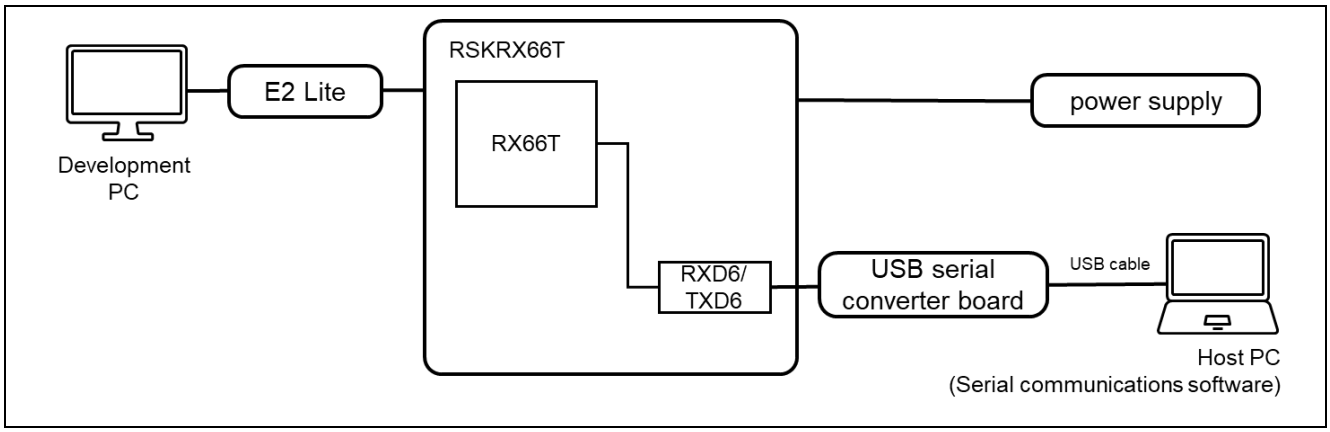


Figure 6.36 RSK-RX66T Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD6	RX
3	RXD6	TX
4	PA4(RTS)	CTS

■ USER_SW (②)

SW1	Note
P10	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P95	LED0

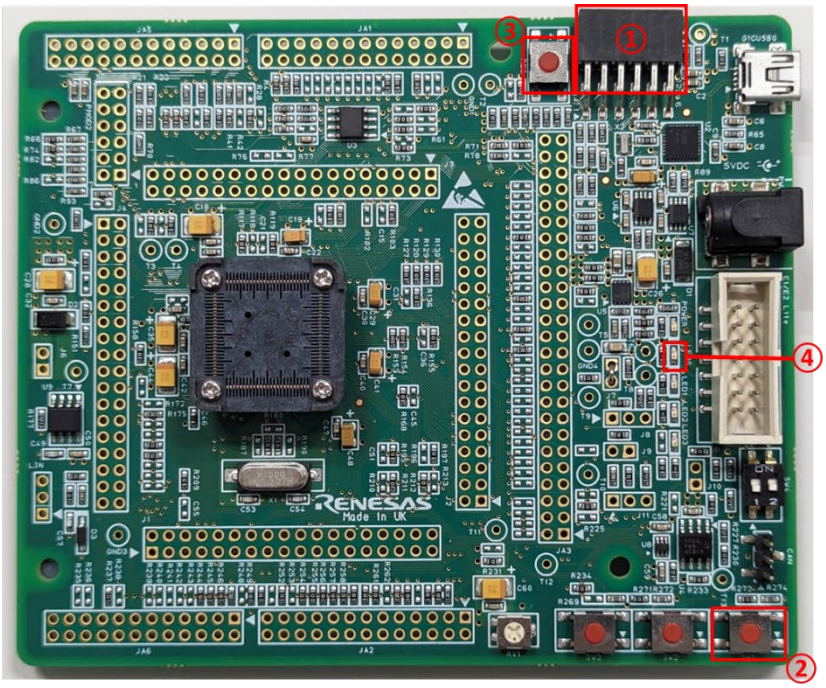


Figure 6.37 RSK-RX66T Pin Information

6.2.9.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX66T linear mode half-surface update method demo project and the memory map of the configuration settings.

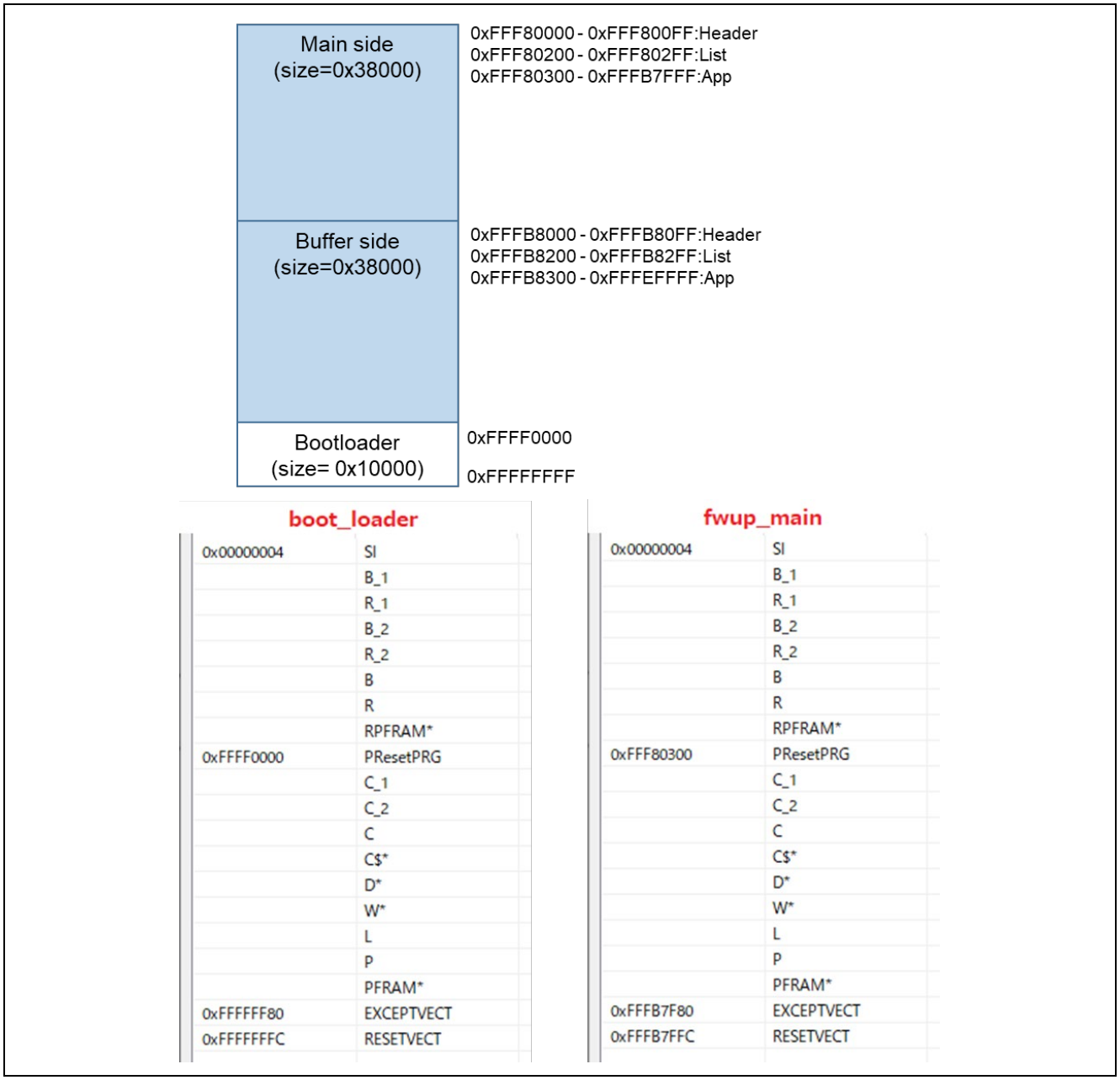


Figure 6.38 RX66T linear mode half-surface update method demo project memory map

Table 6.25 RX66T linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFFB8000	
FWUP_CFG_AREA_SIZE	0x38000	
FWUP_CFG_CF_BLK_SIZE	0x8000	
FWUP_CFG_CF_W_UNIT_SIZE	256	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	512	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.9.2 Memory map of demo project for full update method in linear mode

The memory map of the RX66T linear mode full update method demo project and the memory map of the configuration settings are shown below.

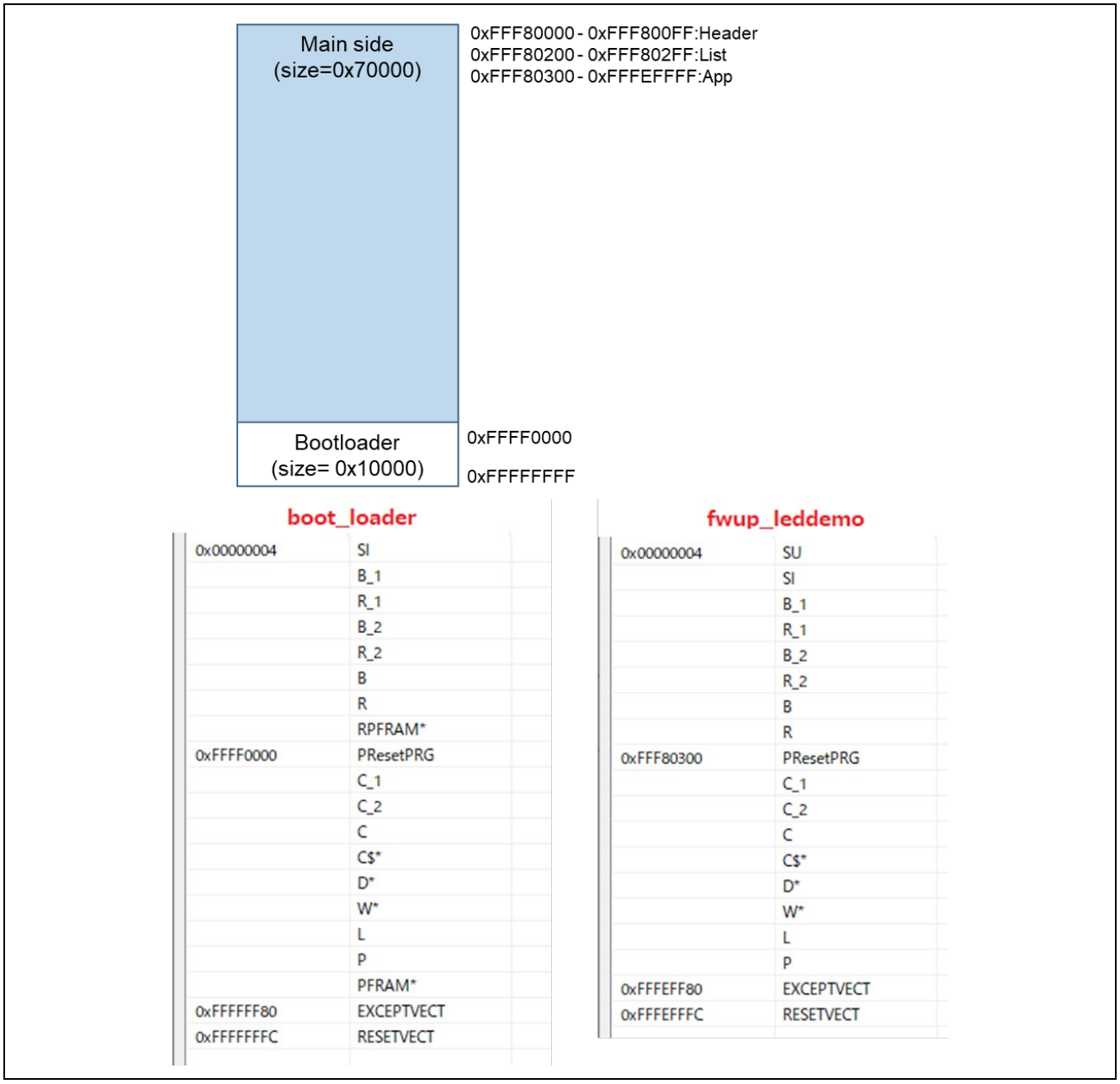


Figure 6.39 RX66T linear mode full update method demo project memory map

Table 6.26 RX66T linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF80000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF80000
FWUP_CFG_AREA_SIZE	0x70000
FWUP_CFG_CF_BLK_SIZE	0x8000
FWUP_CFG_CF_W_UNIT_SIZE	256
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x0000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x00100000
FWUP_CFG_DF_BLK_SIZE	64
FWUP_CFG_DF_NUM_BLKS	512
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.10 Operation Confirmation Environment for RX660

The execution environment and connection diagram are shown below.

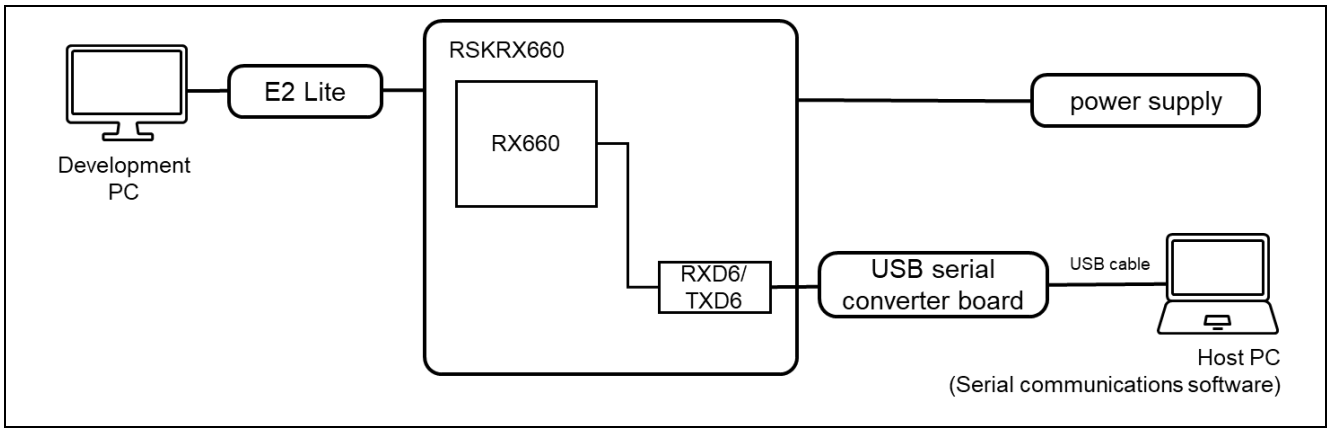


Figure 6.40 RSK-RX660 Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD6	RX
3	RXD6	TX
4	P02(RTS)	CTS

■ USER_SW (②)

SW1	Note
P91	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P17	LED0

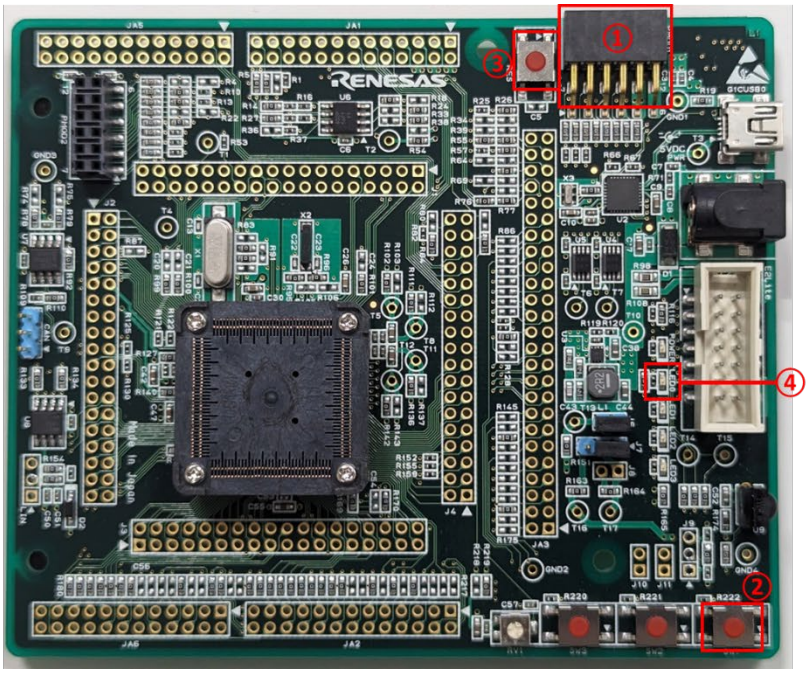


Figure 6.41 RSK-RX660 Pin Information

6.2.10.1 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX660 linear mode half-surface update method demo project and the memory map of the configuration settings.

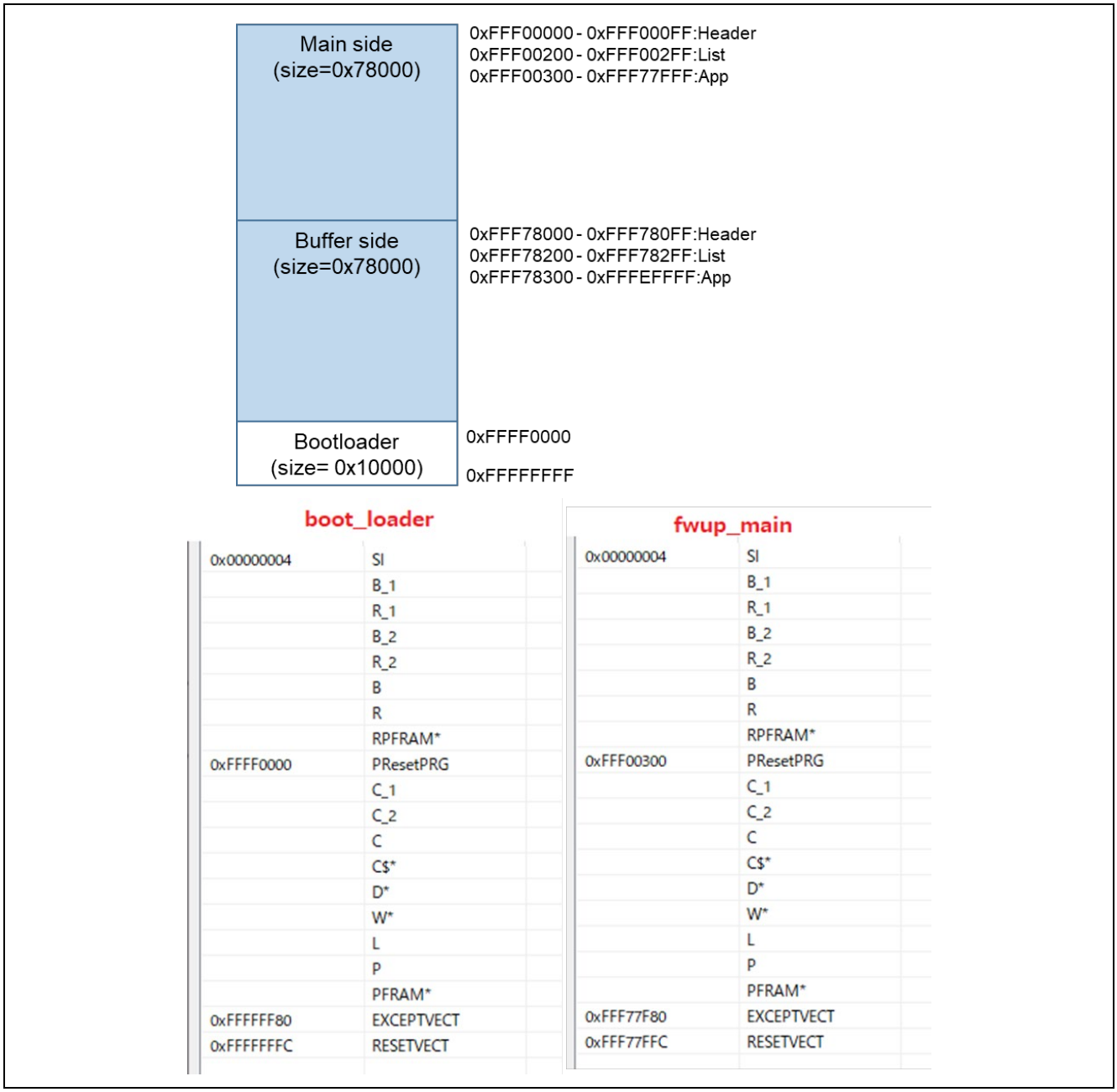


Figure 6.42 RX660 linear mode half-surface update method demo project memory map

Table 6.27 RX660 linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF00000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF78000	
FWUP_CFG_AREA_SIZE	0x78000	
FWUP_CFG_CF_BLK_SIZE	0x8000	
FWUP_CFG_CF_W_UNIT_SIZE	256	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	512	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.10.2 Memory map of demo project for full update method in linear mode

The memory map of the RX660 linear mode full update method demo project and the memory map of the configuration settings are shown below.

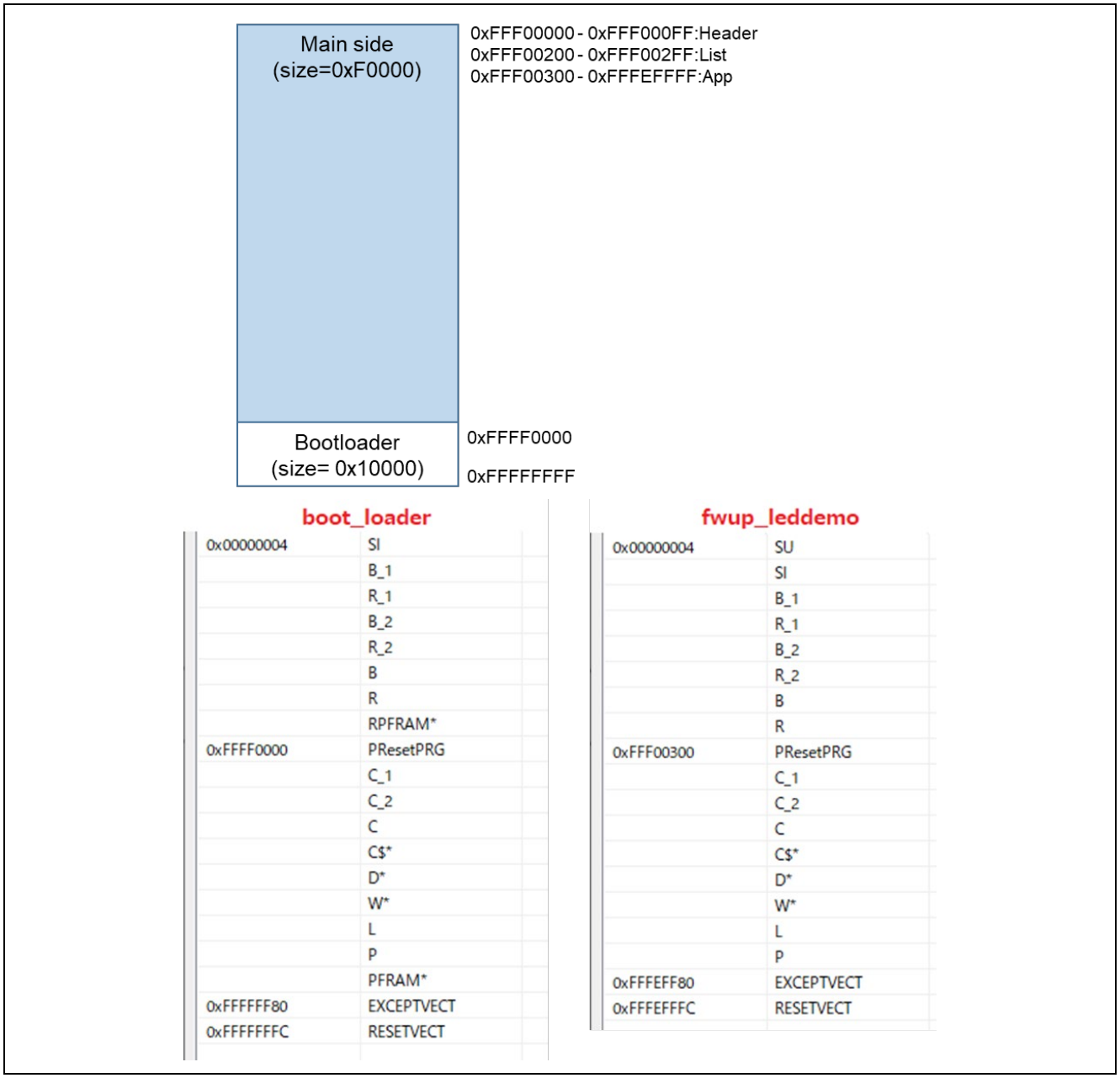


Figure 6.43 RX660 linear mode full update method demo project memory map

Table 6.28 RX660 linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF00000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFF00000
FWUP_CFG_AREA_SIZE	0xF0000
FWUP_CFG_CF_BLK_SIZE	0x8000
FWUP_CFG_CF_W_UNIT_SIZE	256
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x00100000
FWUP_CFG_DF_BLK_SIZE	64
FWUP_CFG_DF_NUM_BLKS	512
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.11 Operation Confirmation Environment for RX671

The execution environment and connection diagram are shown below.

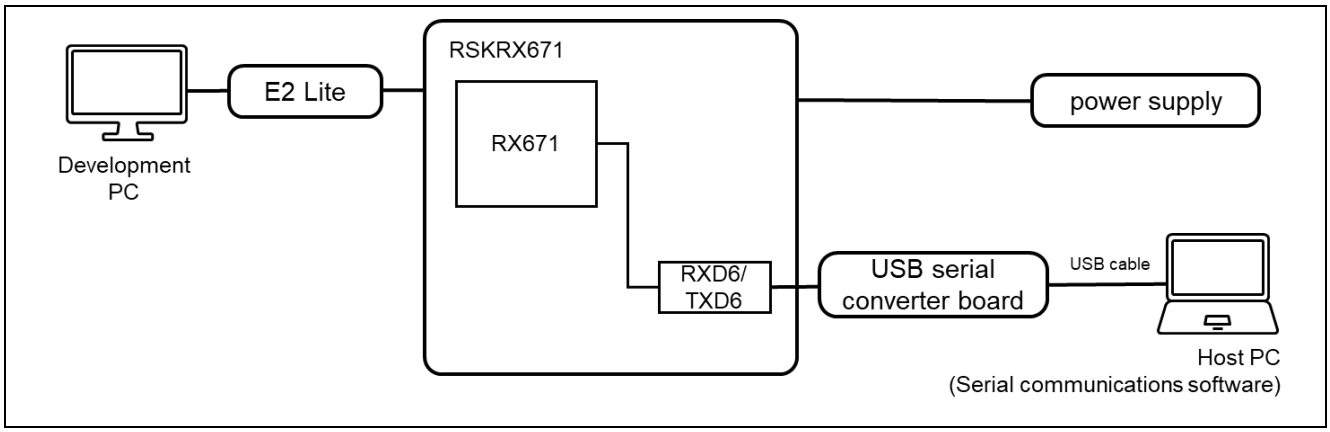


Figure 6.44 RSK-RX671 Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD6	RX
3	RXD6	TX
4	P02(RTS)	CTS

■ USER_SW (②)

SW1	Note
P91	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P17	LED0

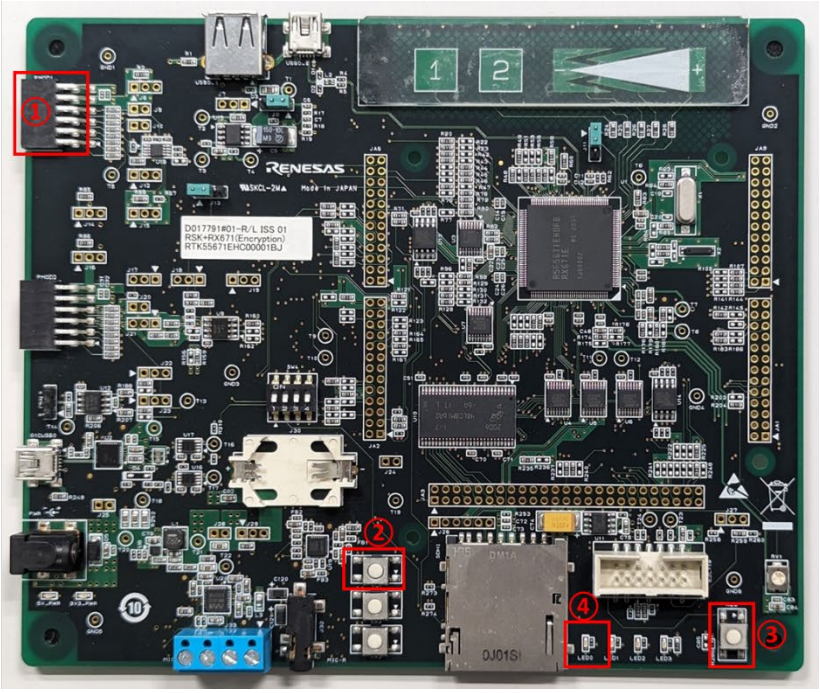


Figure 6.45 RSK-RX671 Pin Information

6.2.11.1 Memory map of dual bank method demo project

The memory map and configuration settings for the RX671 dual-bank method demo project are shown below.

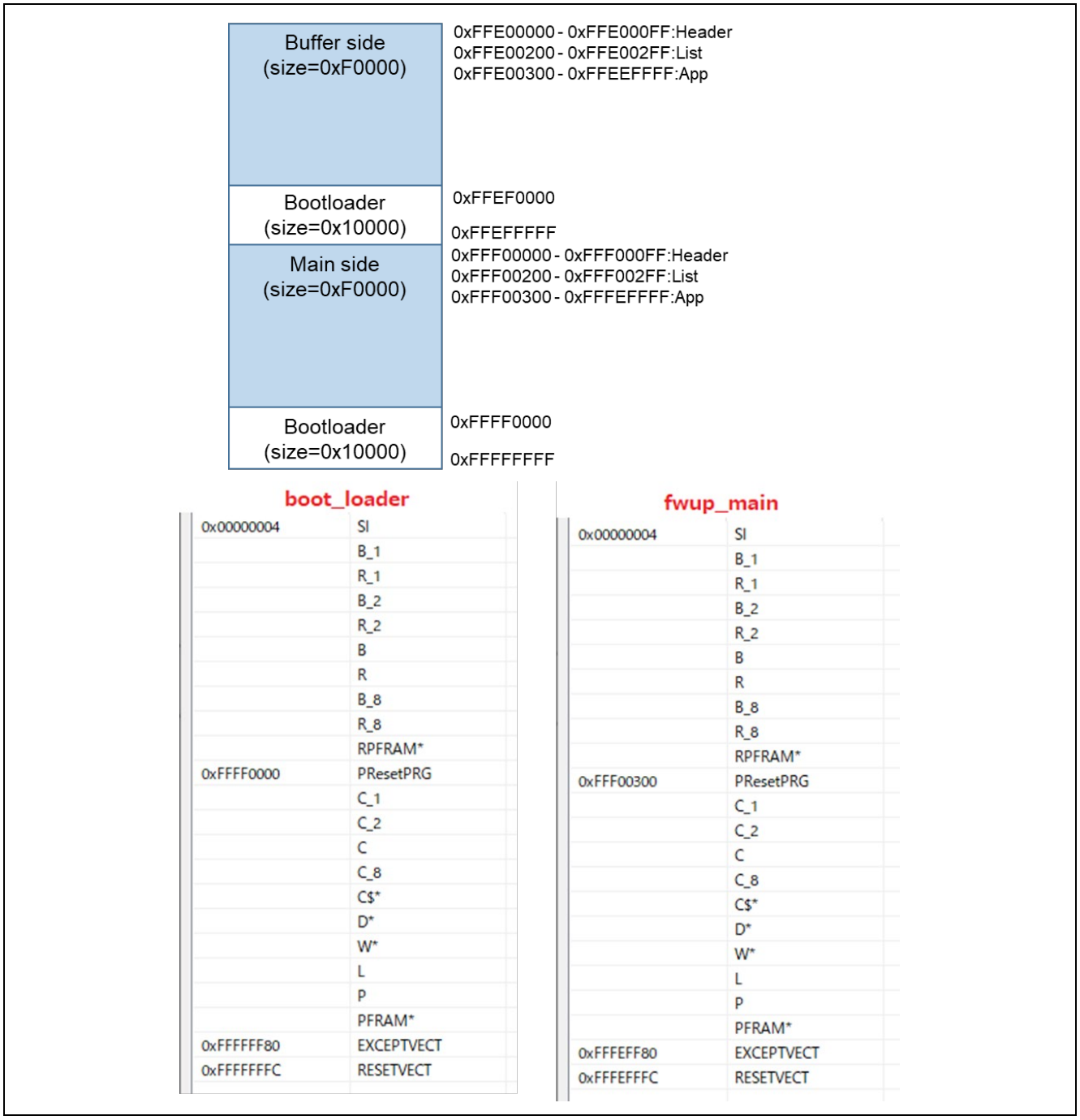


Figure 6.46 RX671 dual bank method demo project memory map

Table 6.29 RX671 dual bank method configuration settings

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	0	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFF00000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFE00000	
FWUP_CFG_AREA_SIZE	0xF0000	
FWUP_CFG_CF_BLK_SIZE	0x8000	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	128	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	1	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	1	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	1	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	1	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	1	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	1	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.11.2 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX671 linear mode half-surface update method demo project and the memory map of the configuration settings.

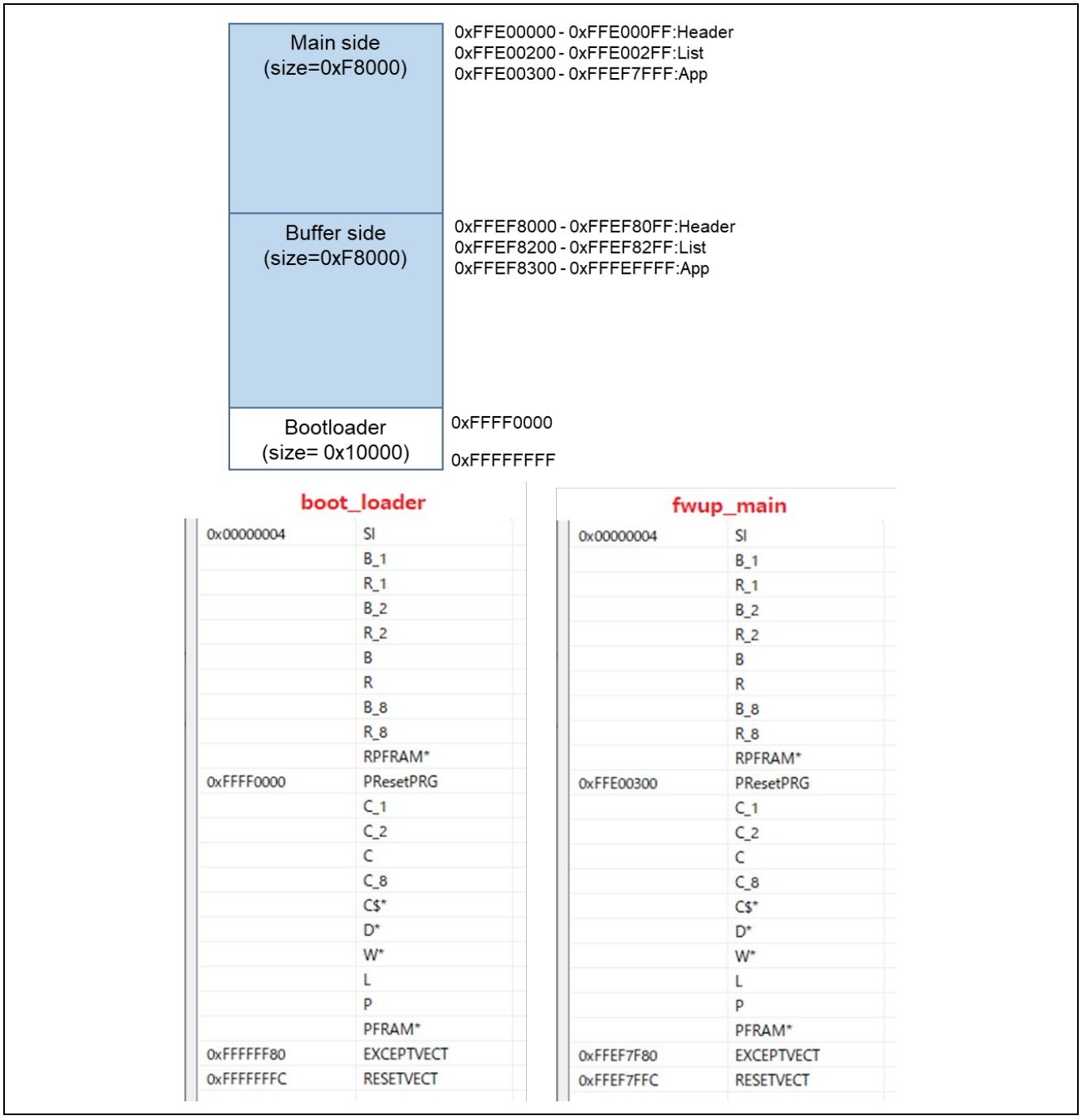


Figure 6.47 RX671 linear mode half-surface update method demo project memory map

Table 6.30 RX671 linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFE00000	
FWUP_CFG_BUF_AREA_ADDR_L	0xFFEF8000	
FWUP_CFG_AREA_SIZE	0xF8000	
FWUP_CFG_CF_BLK_SIZE	0x8000	
FWUP_CFG_CF_W_UNIT_SIZE	128	
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000	
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	
FWUP_CFG_DF_ADDR_L	0x00100000	
FWUP_CFG_DF_BLK_SIZE	64	
FWUP_CFG_DF_NUM_BLKS	128	
FWUP_CFG_FWUPV1_COMPATIBLE	0	
FWUP_CFG_SIGNATURE_VERIFICATION	0	
FWUP_CFG_PRINTF_DISABLE	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.11.3 Memory map of demo project for full update method in linear mode

The memory map of the RX671 linear mode full update method demo project and the memory map of the configuration settings are shown below.

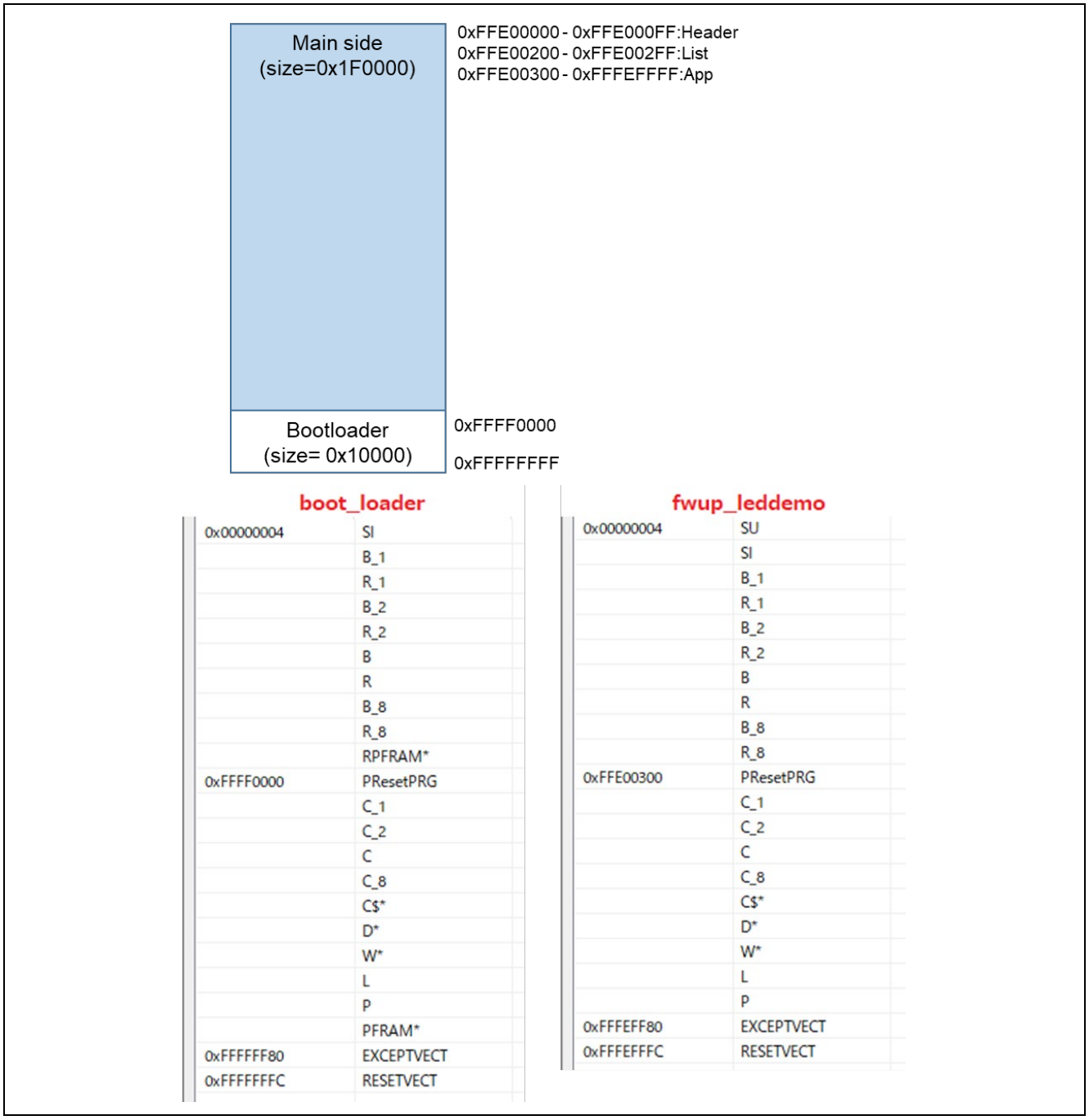


Figure 6.48 RX671 linear mode full update method demo project memory map

Table 6.31 RX671 linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFE00000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFE00000
FWUP_CFG_AREA_SIZE	0x1F0000
FWUP_CFG_CF_BLK_SIZE	0x8000
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x00100000
FWUP_CFG_DF_BLK_SIZE	64
FWUP_CFG_DF_NUM_BLKS	128
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.2.12 Operation Confirmation Environment for RX72N

The execution environment and connection diagram are shown below.

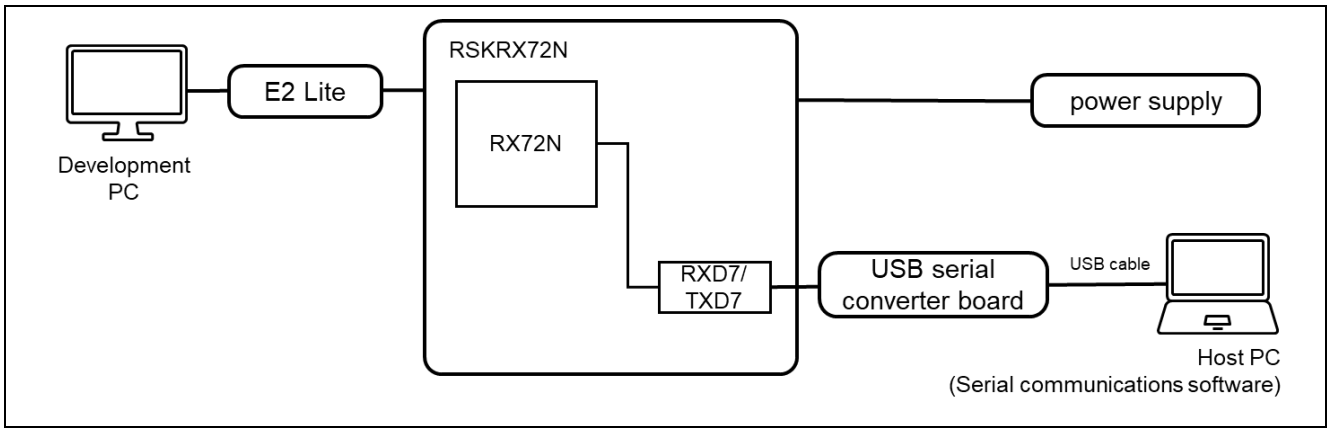


Figure 6.49 RSK-RX72N Device Connection Diagram

The pin assignment is shown in the figure below.

■ UART(①)

PMOD1		USB-UART
2	TXD7	RX
3	RXD7	TX
4	PH0(RTS)	CTS

■ USER_SW (②)

SW1	Note
P45	LOW : USER_SW is ON

■ Reset Switch (③)

RES1	Note
RES#	LOW : USER_SW is ON

■ USER_LED (④)

LED0	Note
P71	LED0

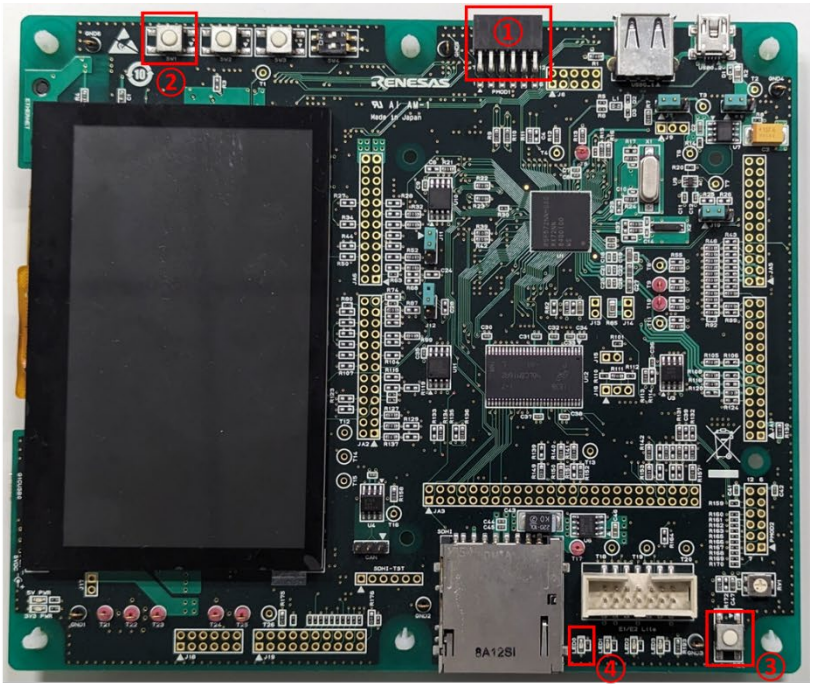


Figure 6.50 RSK-RX72N Pin Information

6.2.12.1 Memory map of dual bank method demo project

The memory map and configuration settings for the RX72N dual-bank method demo project are shown below.

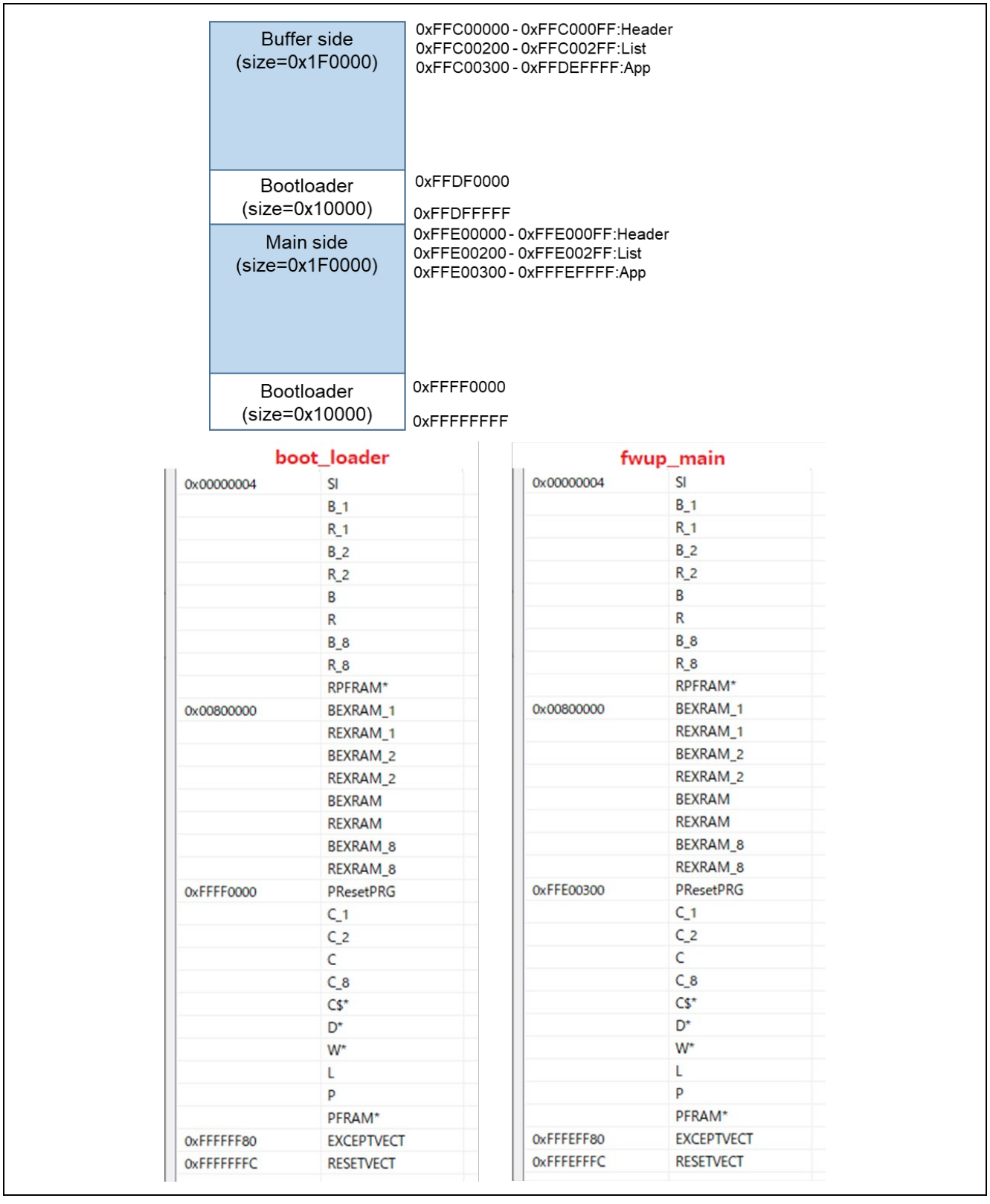


Figure 6.51 RX72N dual bank method demo project memory map

Table 6.32 RX72N dual bank method configuration settings

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	0	0
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFE00000	0xFFE00000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFC00000	0xFFC00000
FWUP_CFG_AREA_SIZE	0x1F0000	0x1F0000
FWUP_CFG_CF_BLK_SIZE	0x8000	0x8000
FWUP_CFG_CF_W_UNIT_SIZE	128	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000	0x00000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	4096
FWUP_CFG_DF_ADDR_L	0x00100000	0x00100000
FWUP_CFG_DF_BLK_SIZE	64	64
FWUP_CFG_DF_NUM_BLKS	512	512
FWUP_CFG_FWUPV1_COMPATIBLE	0	0
FWUP_CFG_SIGNATURE_VERIFICATION	0	0
FWUP_CFG_PRINTF_DISABLE	0	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	1	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	1	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	1	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	1	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	1	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	1	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.12.2 Memory map of demo project for half-surface update method in linear mode

Shown below are the memory map of the RX72N linear mode half-surface update method demo project and the memory map of the configuration settings.

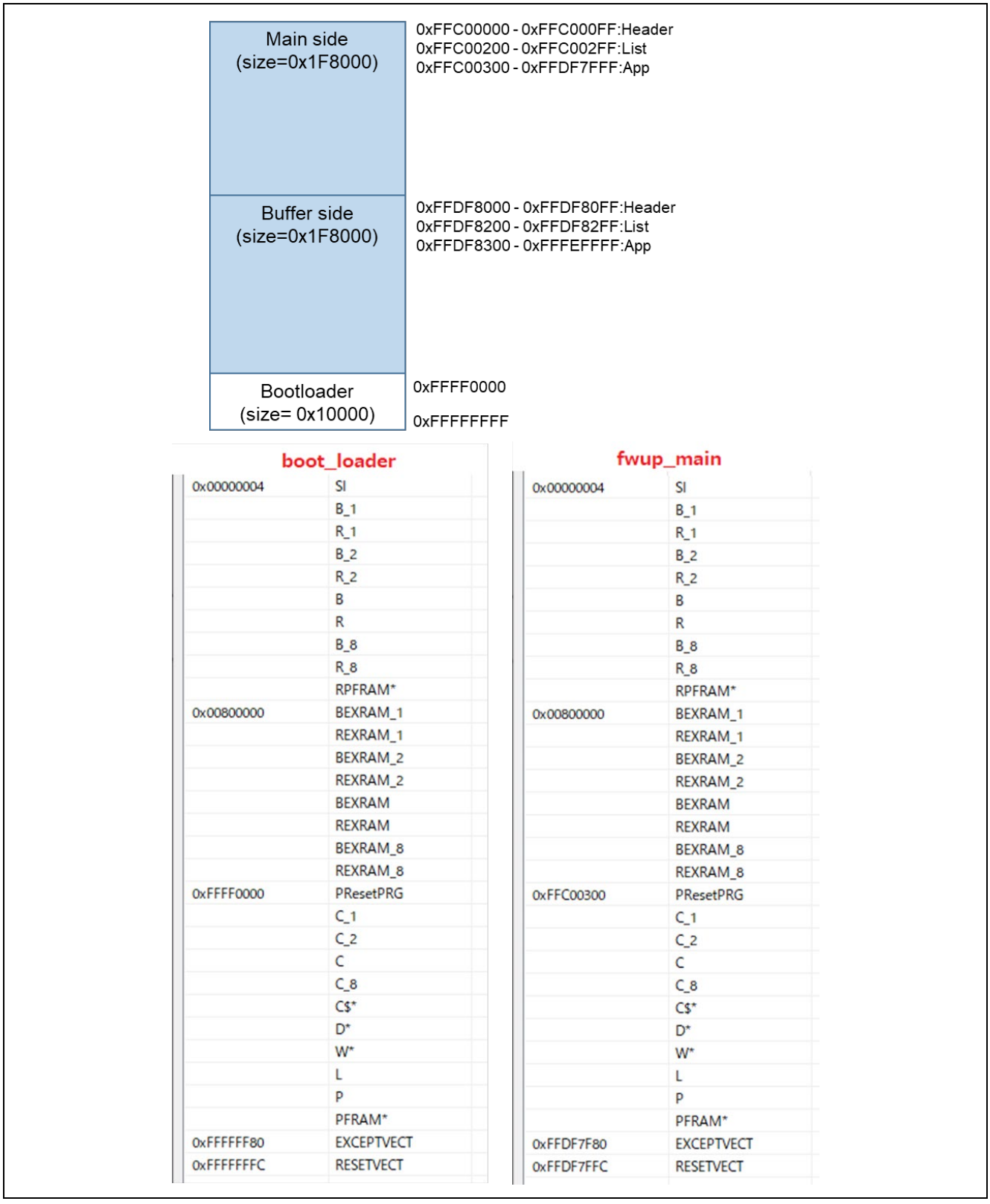


Figure 6.52 RX72N linear mode half-surface update method demo project memory map

Table 6.33 RX72N linear mode half-surface update method configuration setting

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_UPDATE_MODE	1	1
FWUP_CFG_FUNCTION_MODE	0	1
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFC00000	0xFFC00000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFDF8000	0xFFDF8000
FWUP_CFG_AREA_SIZE	0x1F8000	0x1F8000
FWUP_CFG_CF_BLK_SIZE	0x8000	0x8000
FWUP_CFG_CF_W_UNIT_SIZE	128	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000	0x00000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096	4096
FWUP_CFG_DF_ADDR_L	0x00100000	0x00100000
FWUP_CFG_DF_BLK_SIZE	64	64
FWUP_CFG_DF_NUM_BLKS	512	512
FWUP_CFG_FWUPV1_COMPATIBLE	0	0
FWUP_CFG_SIGNATURE_VERIFICATION	0	0
FWUP_CFG_PRINTF_DISABLE	0	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function	
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0	
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function	
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function	
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0	
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function	
FWUP_CFG_USER_SHA256_INIT_ENABLED	0	
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function	
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0	
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function	
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0	
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function	
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0	
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0	
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function	

Configuration options in r_fwup _config.h		
parameter name	**boot_loader**	**fwup_main**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0	
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function	
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0	
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function	
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0	
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function	
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0	
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function	
FWUP_CFG_USER_FLASH_READ_ENABLED	0	
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function	
FWUP_CFG_USER_BANK_SWAP_ENABLED	0	
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function	

6.2.12.3 Memory map of demo project for full update method in linear mode

The memory map of the RX72N linear mode full update method demo project and the memory map of the configuration settings are shown below.

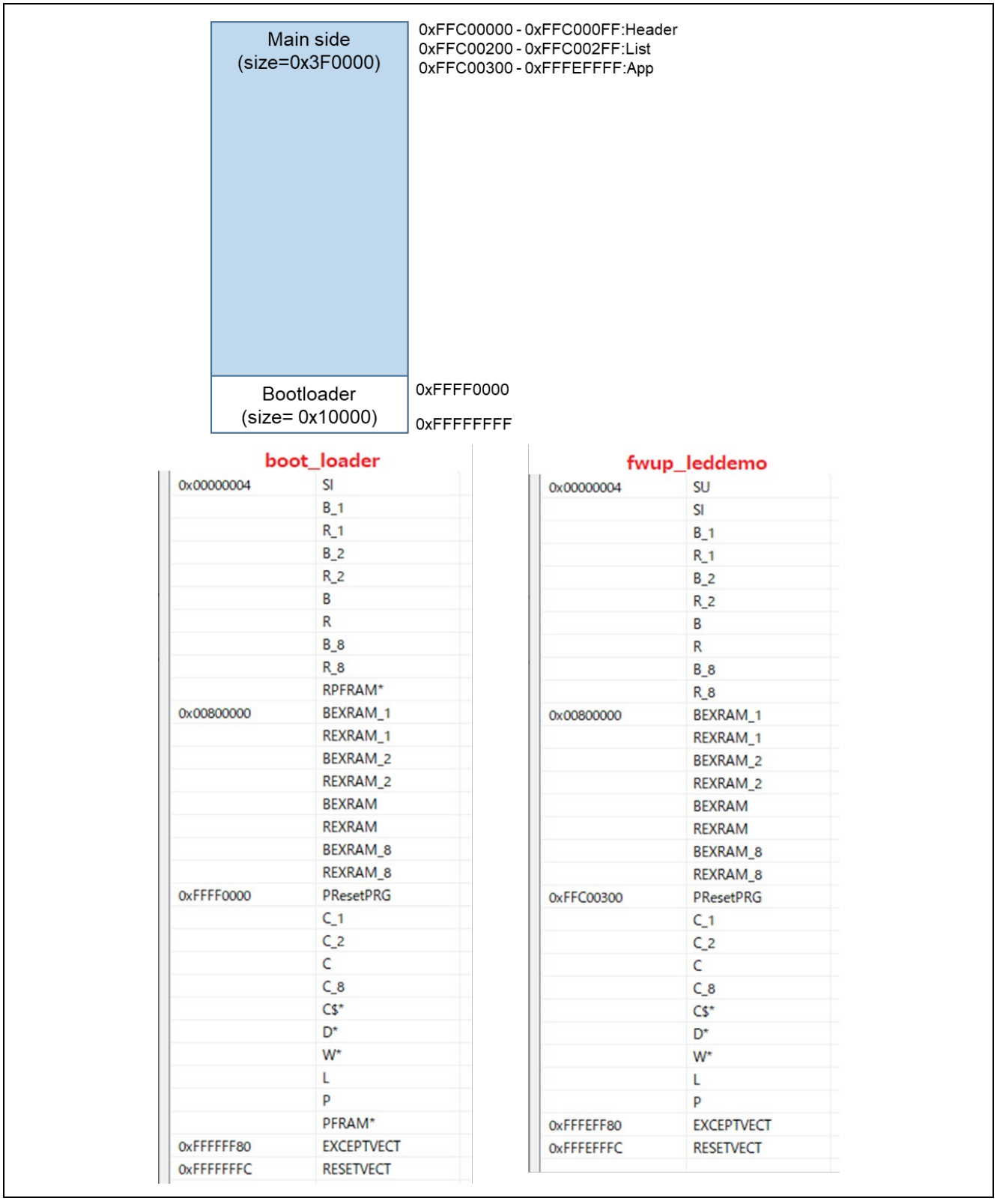


Figure 6.53 RX72N linear mode full update method demo project memory map

Table 6.34 RX72N linear mode full update method configuration setting

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_UPDATE_MODE	2
FWUP_CFG_FUNCTION_MODE	0
FWUP_CFG_MAIN_AREA_ADDR_L	0xFFC00000
FWUP_CFG_BUF_AREA_ADDR_L	0xFFC00000
FWUP_CFG_AREA_SIZE	0x3F0000
FWUP_CFG_CF_BLK_SIZE	0x8000
FWUP_CFG_CF_W_UNIT_SIZE	128
FWUP_CFG_EXT_BUF_AREA_ADDR_L (unused)	0x00000
FWUP_CFG_EXT_BUF_AREA_BLK_SIZE (unused)	4096
FWUP_CFG_DF_ADDR_L	0x00100000
FWUP_CFG_DF_BLK_SIZE	64
FWUP_CFG_DF_NUM_BLKS	512
FWUP_CFG_FWUPV1_COMPATIBLE	0
FWUP_CFG_SIGNATURE_VERIFICATION	0
FWUP_CFG_PRINTF_DISABLE	0
FWUP_CFG_USER_DISABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_DISABLE_INTERRUPT_FUNCTION	my_disable_interrupt_function
FWUP_CFG_USER_ENABLE_INTERRUPT_ENABLED	0
FWUP_CFG_USER_ENABLE_INTERRUPT_FUNCTION	my_enable_interrupt_function
FWUP_CFG_USER_SOFTWARE_DELAY_ENABLED	0
FWUP_CFG_USER_SOFTWARE_DELAY_FUNCTION	my_software_delay_function
FWUP_CFG_USER_SOFTWARE_RESET_ENABLED	0
FWUP_CFG_USER_SOFTWARE_RESET_FUNCTION	my_software_reset_function
FWUP_CFG_USER_SHA256_INIT_ENABLED	0
FWUP_CFG_USER_SHA256_INIT_FUNCTION	my_sha256_init_function
FWUP_CFG_USER_SHA256_UPDATE_ENABLED	0
FWUP_CFG_USER_SHA256_UPDATE_FUNCTION	my_sha256_update_function
FWUP_CFG_USER_SHA256_FINAL_ENABLED	0
FWUP_CFG_USER_SHA256_FINAL_FUNCTION	my_sha256_final_function
FWUP_CFG_USER_VERIFY_ECDSA_ENABLED	0
FWUP_CFG_USER_VERIFY_ECDSA_FUNCTION	my_verify_ecdsa_function
FWUP_CFG_USER_GET_CRYPT_CONTEXT_ENABLED	0
FWUP_CFG_USER_GET_CRYPT_CONTEXT_FUNCTION	my_get_crypt_context_function

Configuration options in r_fwup _config.h	
parameter name	**boot_loader**
FWUP_CFG_USER_FLASH_OPEN_ENABLED	0
FWUP_CFG_USER_FLASH_OPEN_FUNCTION	my_flash_open_function
FWUP_CFG_USER_FLASH_CLOSE_ENABLED	0
FWUP_CFG_USER_FLASH_CLOSE_FUNCTION	my_flash_close_function
FWUP_CFG_USER_FLASH_ERASE_ENABLED	0
FWUP_CFG_USER_FLASH_ERASE_FUNCTION	my_flash_erase_function
FWUP_CFG_USER_FLASH_WRITE_ENABLED	0
FWUP_CFG_USER_FLASH_WRITE_FUNCTION	my_flash_write_function
FWUP_CFG_USER_FLASH_READ_ENABLED	0
FWUP_CFG_USER_FLASH_READ_FUNCTION	my_flash_read_function
FWUP_CFG_USER_BANK_SWAP_ENABLED	0
FWUP_CFG_USER_BANK_SWAP_FUNCTION	my_bank_swap_function

6.3 How to debug the demo project

If you wish to debug this project (bootloader + application program) in the e2 studio environment, the following procedure can be used.

This demo project is set to be powered by the emulator in the debugger (E2 Lite). If you want to connect with other debuggers or supply power from the target board, change the debugger settings.

(1) Build the bootloader and application program without optimization.

Build the bootloader (boot_loader) and application program (fwup_main) with the e2 studio optimization level set to no optimization.

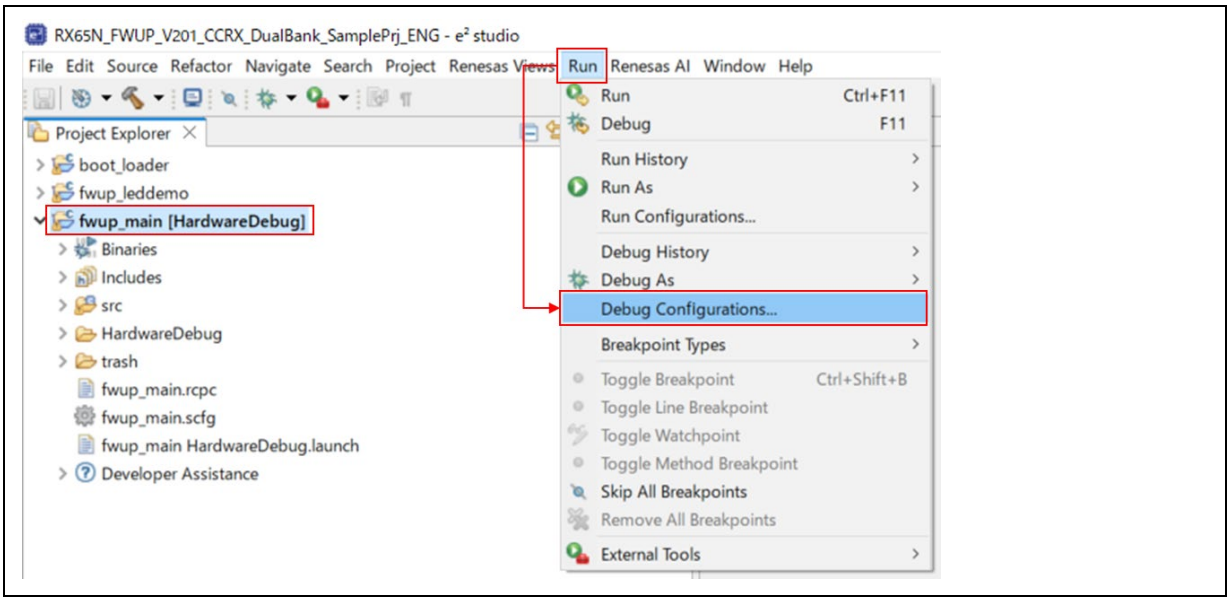
(2) Generate the initial image.

The Renesas Image Generator generates an initial image file (.mot) consisting of a bootloader (boot_loader) and an application program (fwup_main).

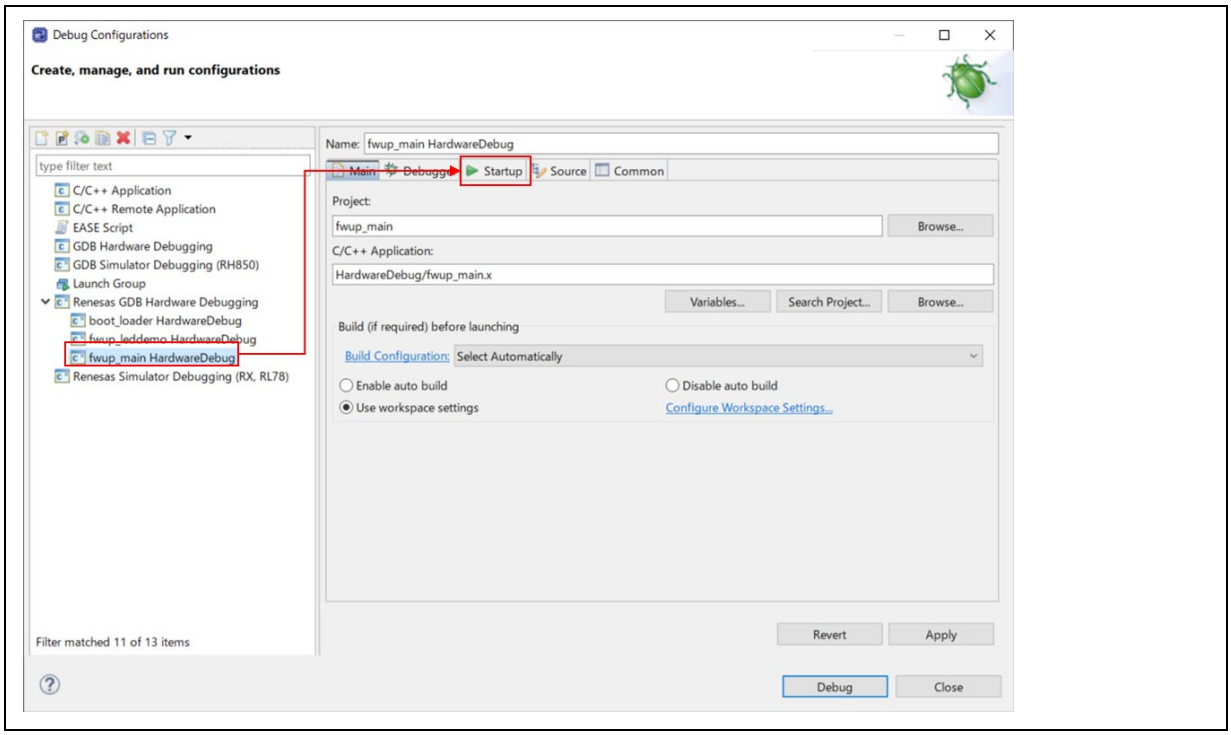
(3) Debug settings for the application program (fwup_main).

Follow the steps below to configure debugging settings for the application program (fwup_main).

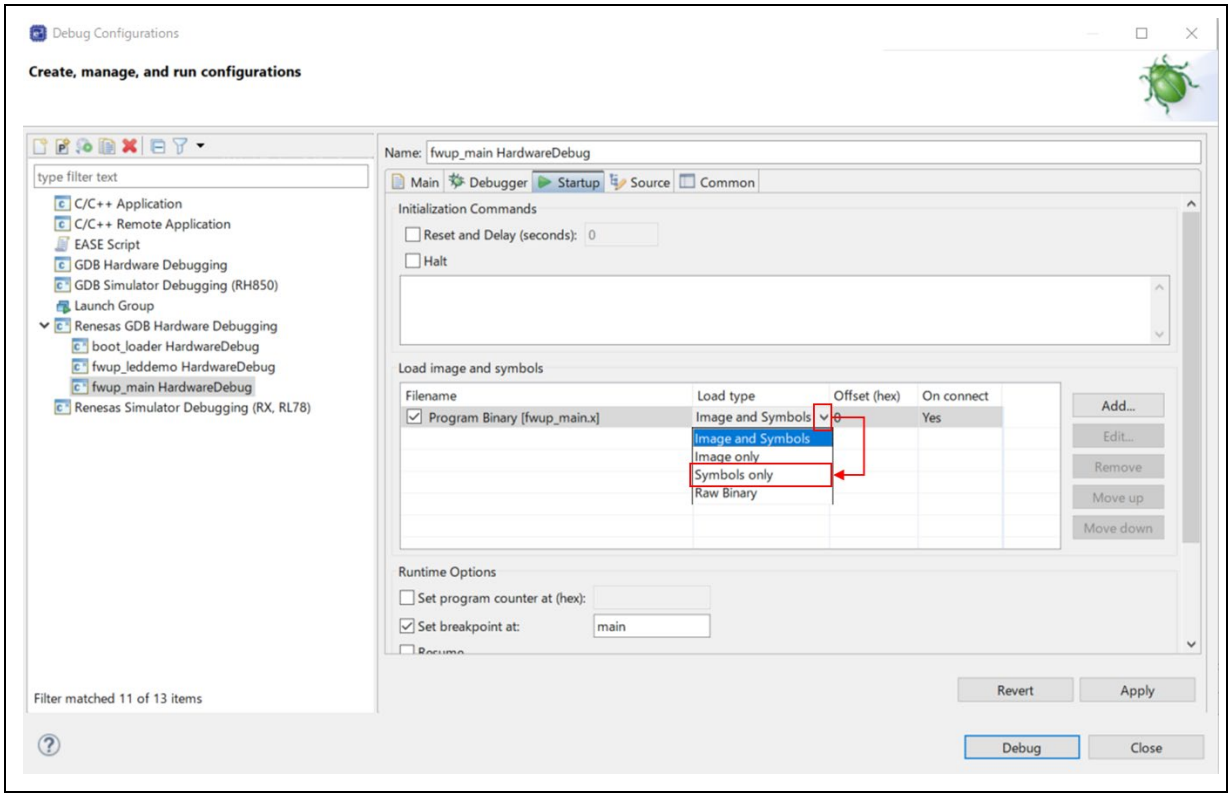
a) Open Run->Debug Configuration and select fwup_main_HardwareDebug.



b) Select fwup_main_HardwareDebug and click Startup.



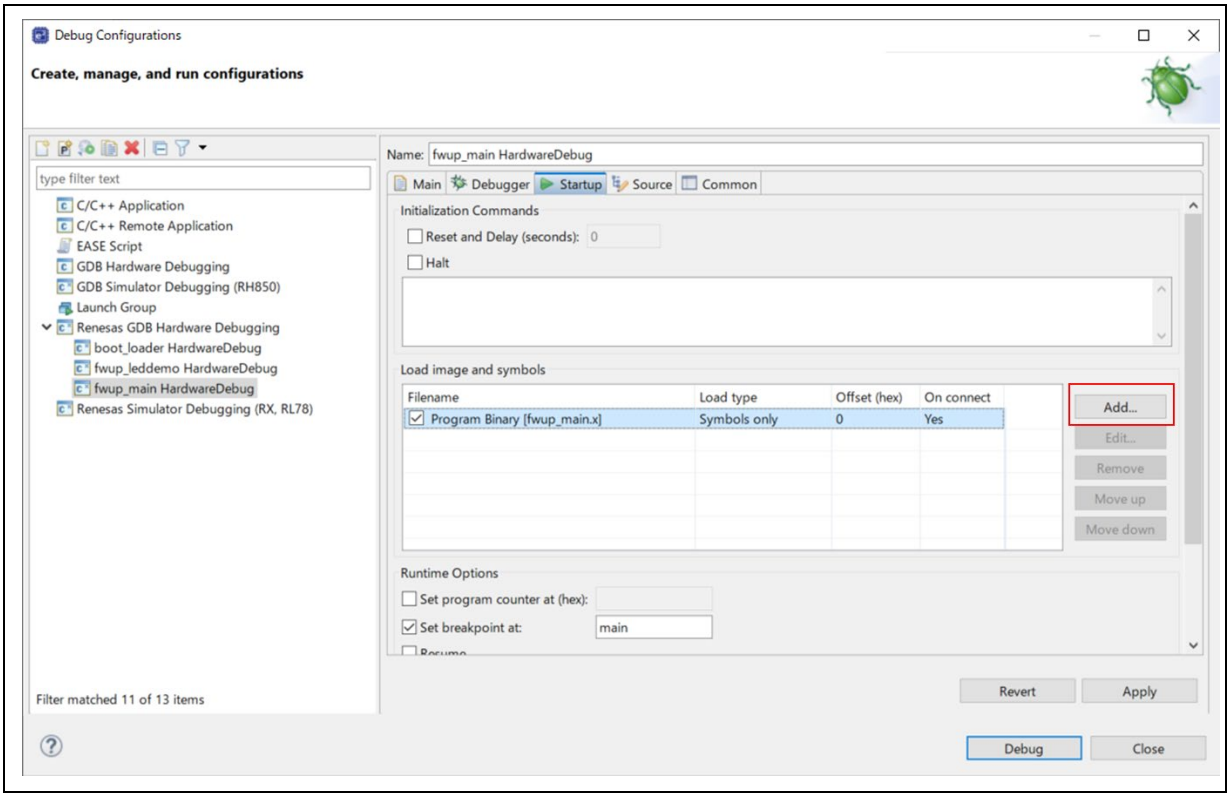
c) Change the load type of the program binary [fwup_main.x] from "Image and Symbol" to "Symbol Only". In the GCC environment, this will be fwup_main.elf.



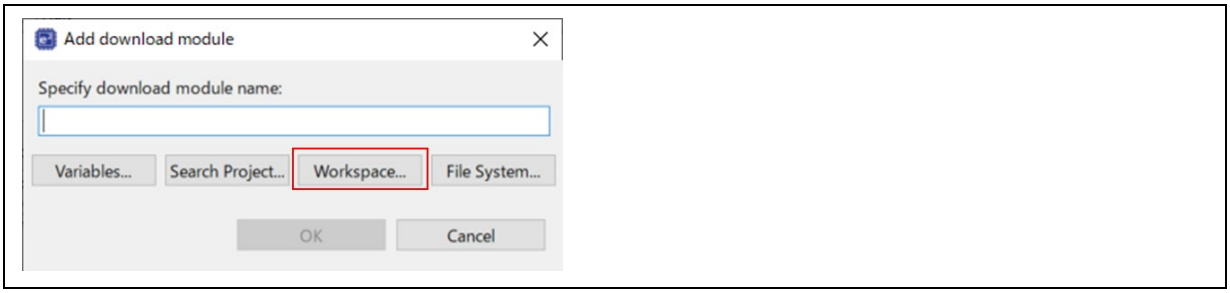
(4) Add the bootloader (boot_loader) symbol.

Follow the procedure below to add the boot loader (boot_loader) symbol built in step (1).

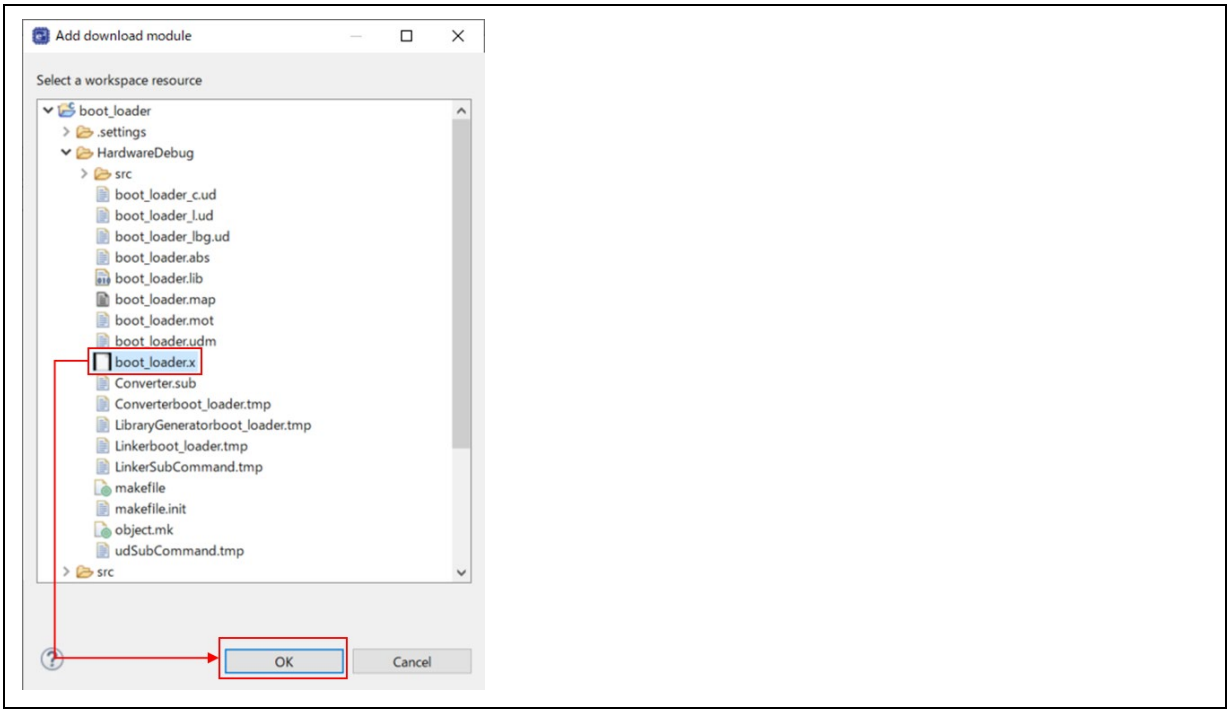
a) Click "Add".



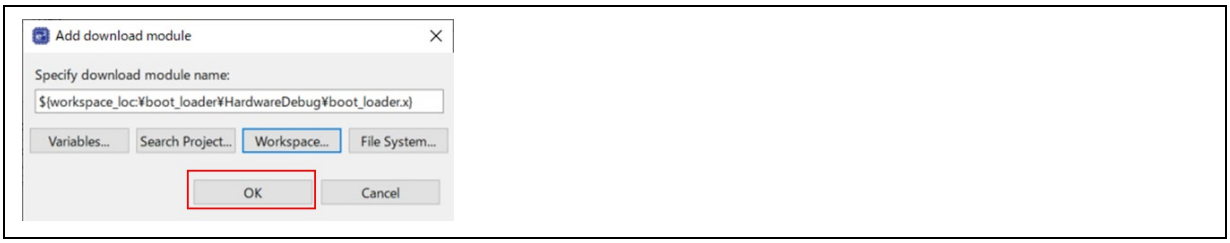
b) Click Workspace.



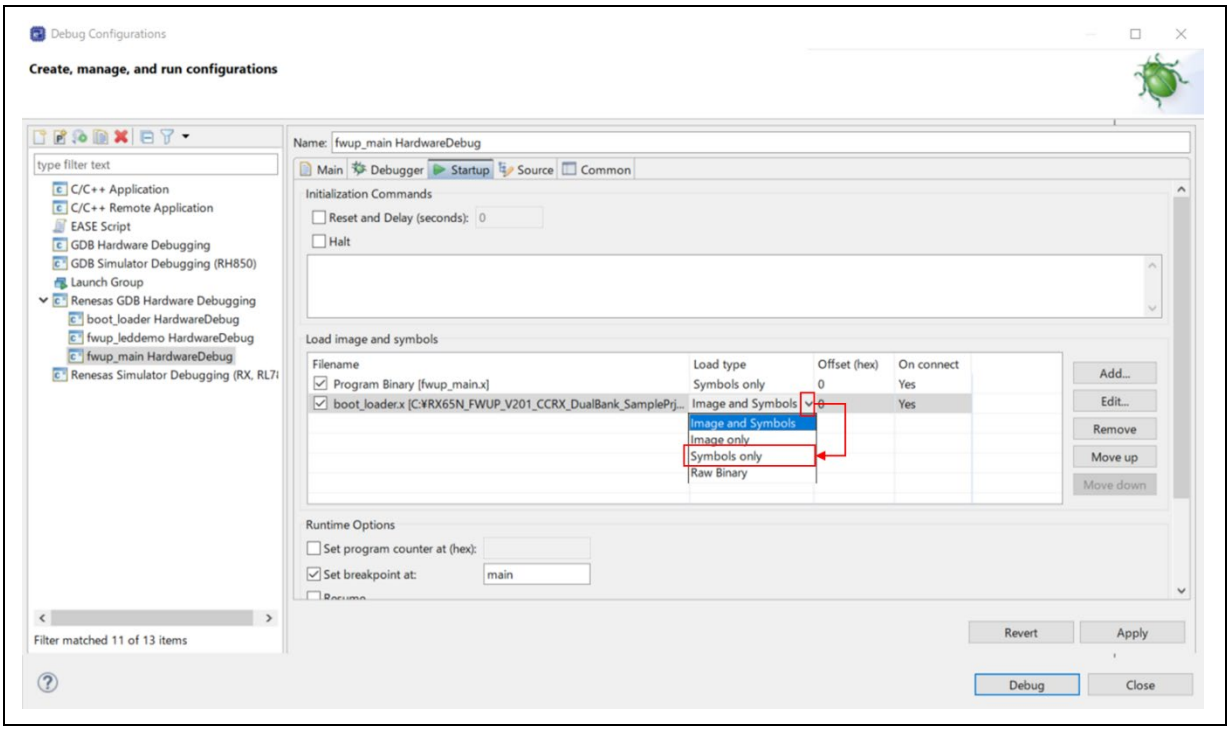
c) Select the bootloader (boot_loader.x) and click "OK." In the GCC environment, the symbol is boot_loader.elf.



d) Confirm that the download module name is set to bootloader (boot_loader.x) and click "OK.



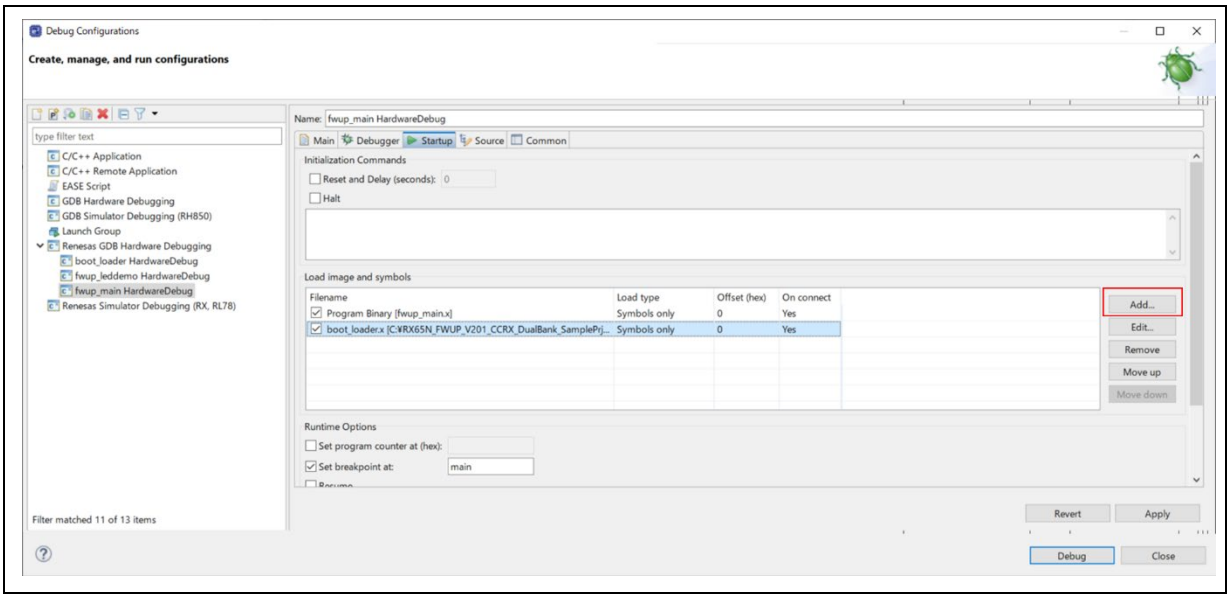
e) Change the load type of the bootloader (boot_loader.x) from "Image and Symbol" to "Symbol only".



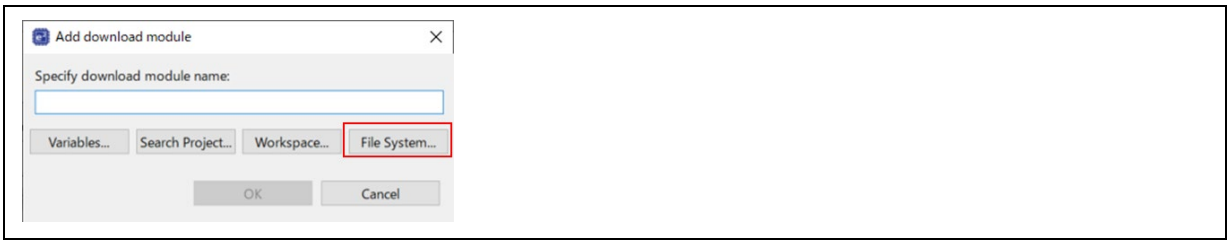
(5) Add the image of the initial image (initial_firm.mot).

Add the image of the initial image (initial_firm.mot) generated in step (2) according to the following procedure.

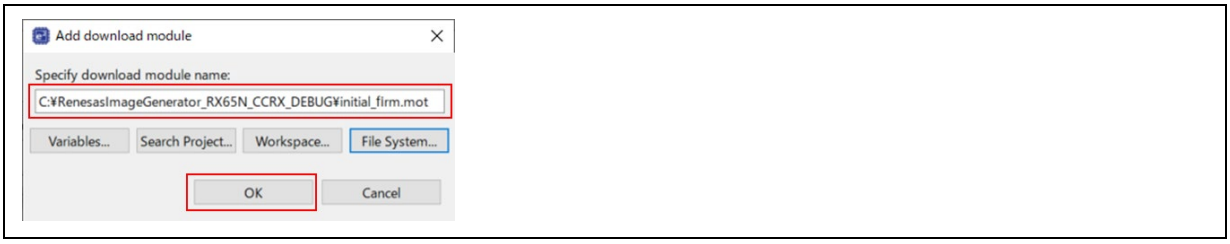
a) Click "Add".



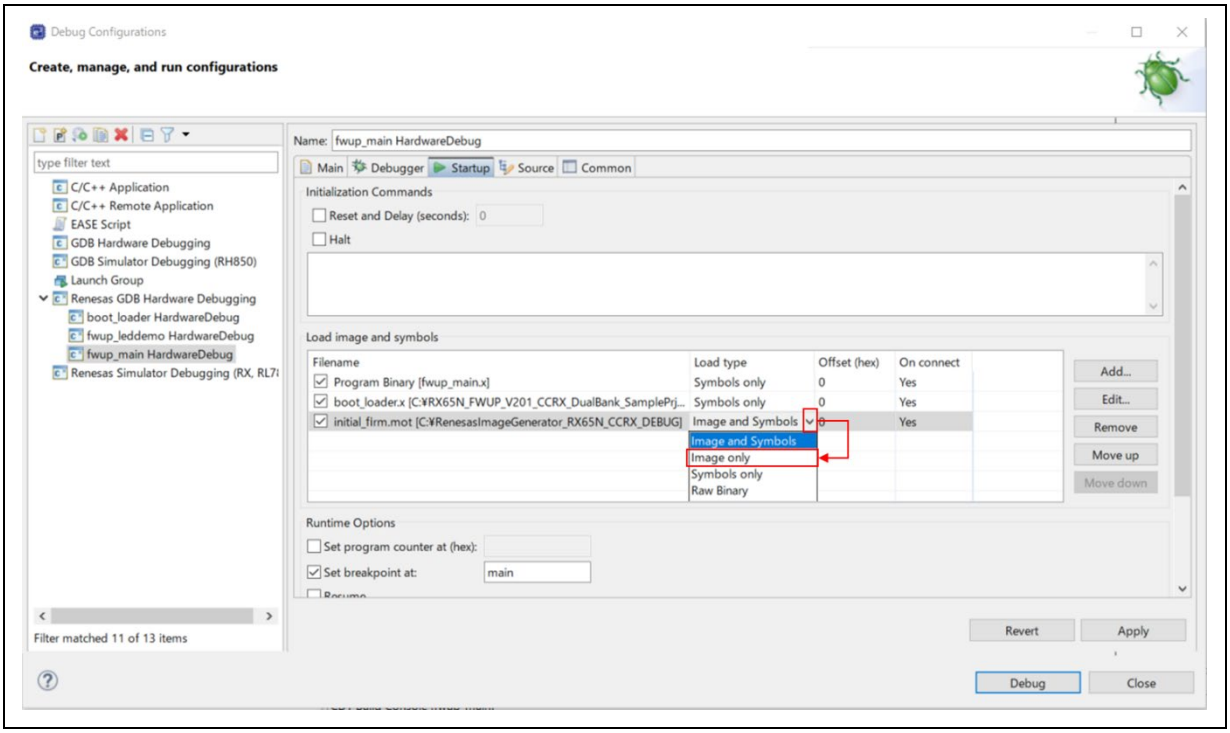
b) Click File System.



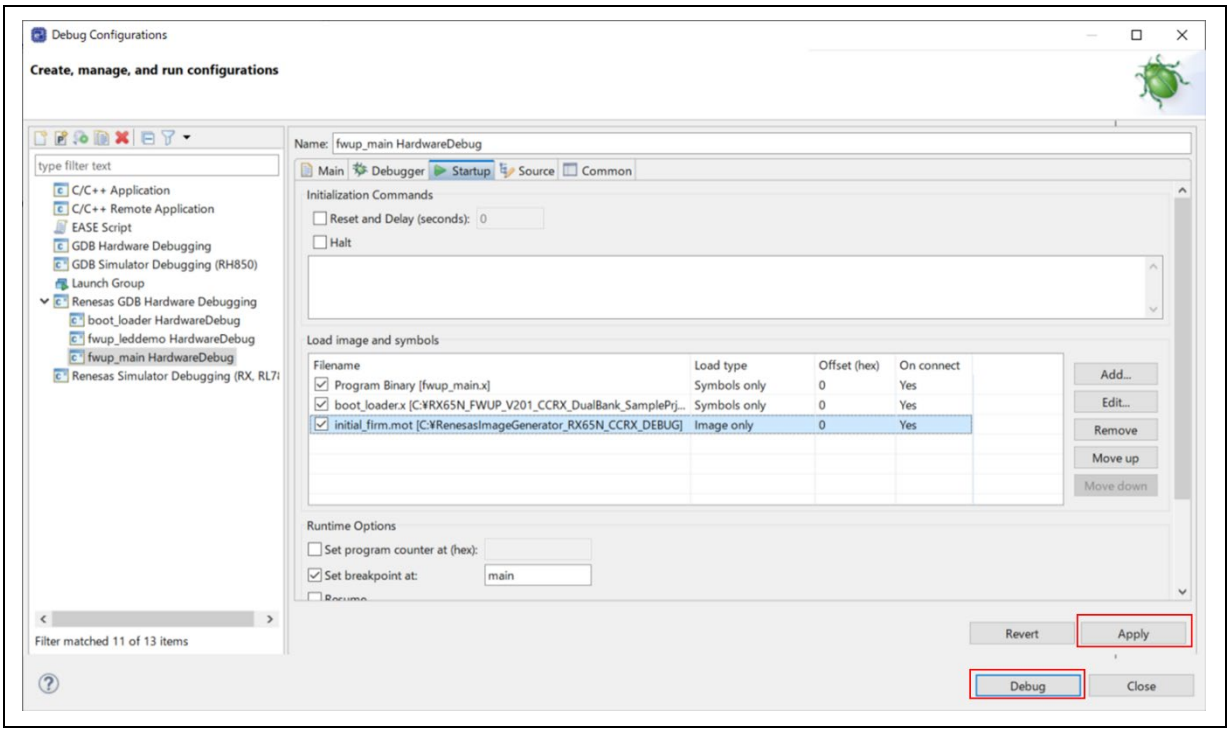
c) Select the initial image (initial_firm.mot) and click "OK".



d) Change the load type of the initial image (initial_firm.mot) from "Image and Symbol" to "Image only" and click "OK".

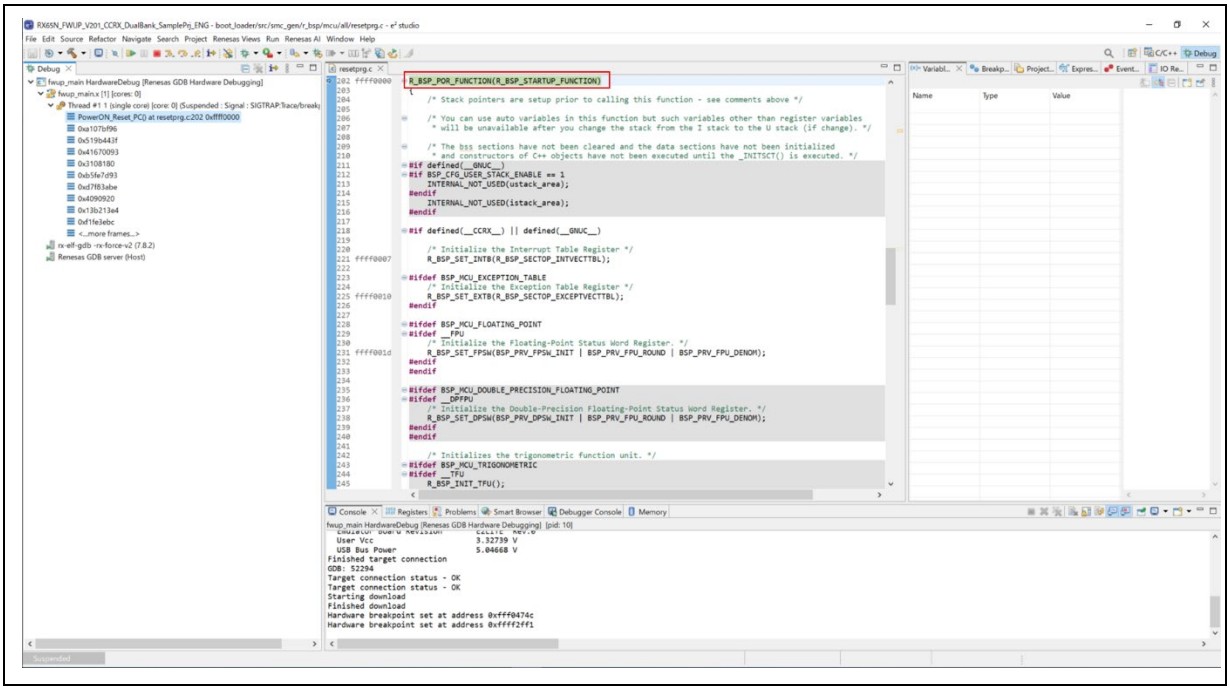


e) Click "Apply" and click "Debug".



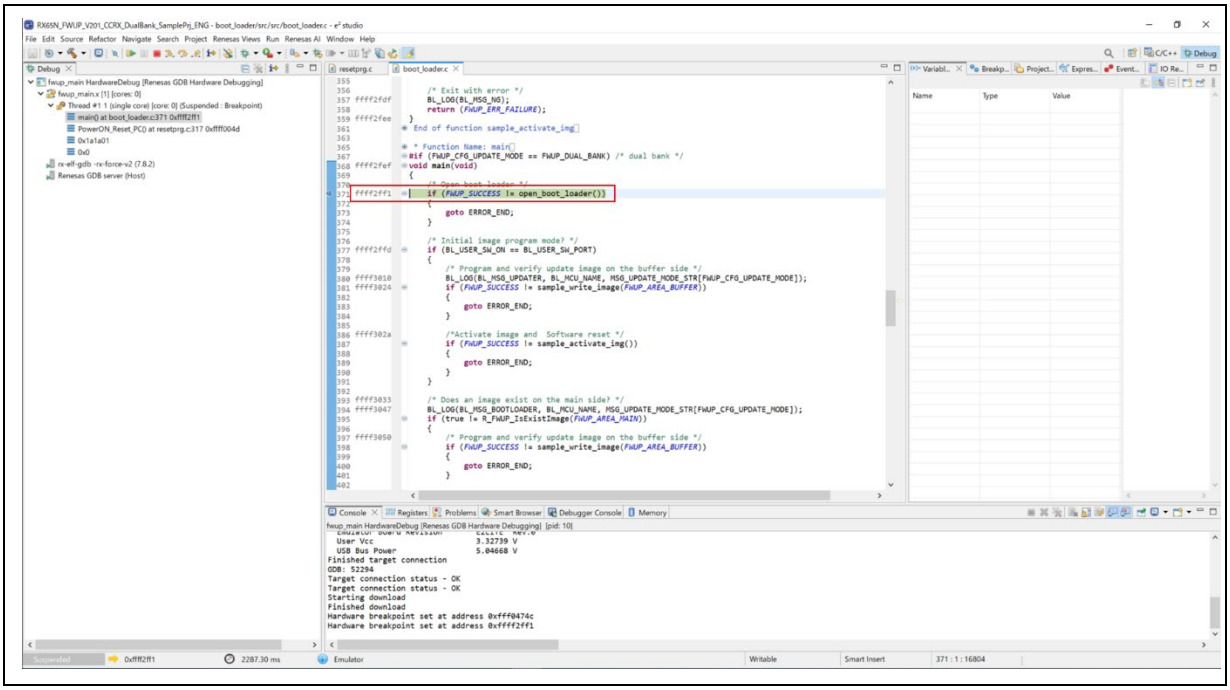
(6) Start the debugger.

When the debugger starts, it jumps to resetprg.c in the boot_loader project.



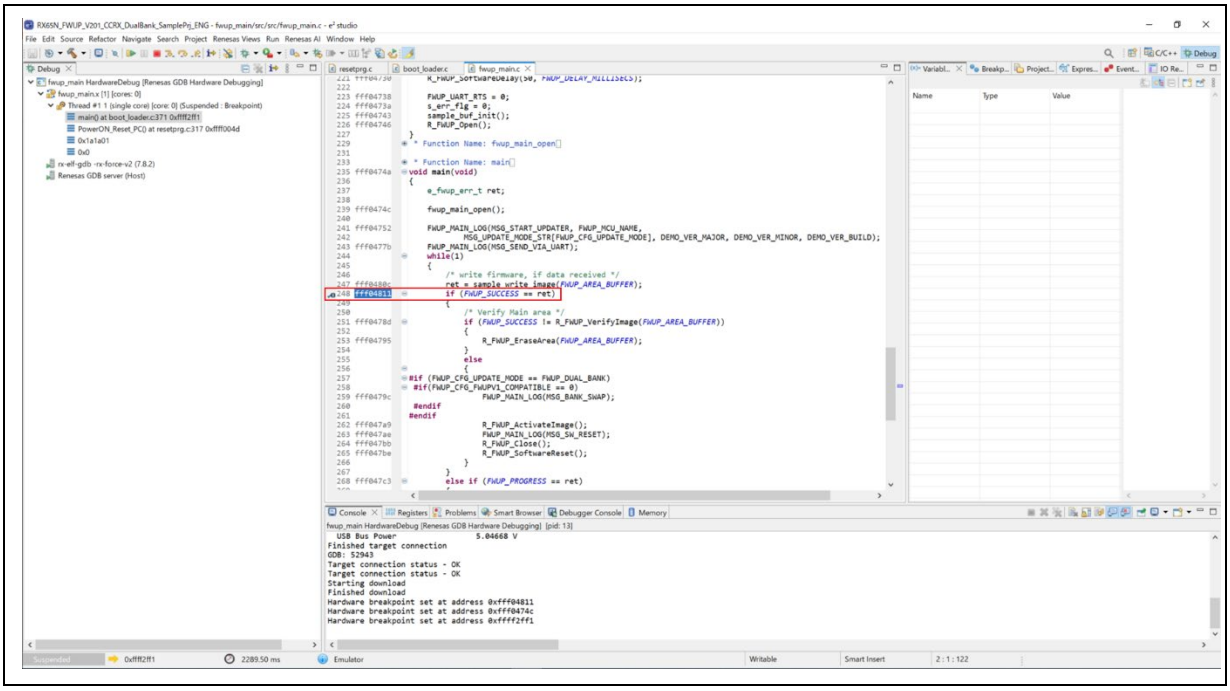
(7) Resume the program.

When you click Resume, the program stops at the beginning of the main() function in boot_loader.c. project.



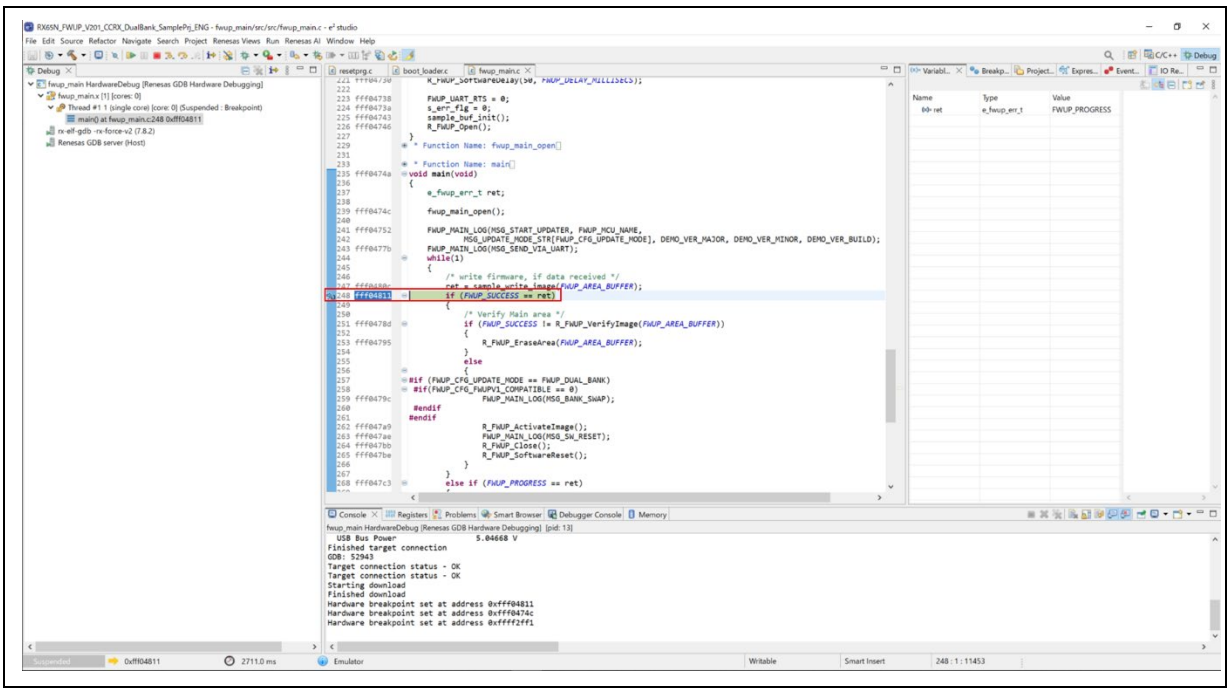
(8) Set a breakpoint in main() of the fwup_main project.

Set a breakpoint in the following red frame in main() of the fwup_main project.



(9) Resume the program.

Click restart and stop at the breakpoint set in (8).



6.4 Open source license information used in the demo project

The demo project for this product uses the open source TinyCrypt. If you use TinyCrypto for your cryptographic library, you must comply with the terms of use set forth in TinyCrypt's license terms.

Check out the TinyCrypt license terms below.

URL：https://github.com/intel/tinycrypt

license：https://github.com/intel/tinycrypt/blob/master/LICENSE

7. Notes

7.1 Notes on Transition from Bootloader to Application.

When transitioning from the sample bootloader program to the application, the settings of the bootloader's peripheral functions will be taken over by the application.

For the peripheral functions used in the sample bootloader (Table 7.1), the API functions of each FIT module are closed at the end of the bootloader. Other settings are default values when the smart configurator is used.

If the customer modifies the bootloader sample program for use, the settings of the peripheral functions set in the bootloader will be inherited by the application side. Therefore, it is recommended to initialize the settings of the peripheral functions before moving from the bootloader to the application, or to share the settings of the peripheral functions with the application.

When creating an application, please take the implementation of the bootloader into consideration.

Table 7.1 Notes on peripheral functions used in the bootloader

Peripheral Functions	FIT Modeule	Settings and Notes on the Boot Loader
Board Functions	r_bsp	These are the default values when the BSP FIT module is embedded in the Smart Configurator. The settings are not changed in the bootloader. Please note that the PMR and PFS registers are also set to match the board.
Functions of Flash Memory	r_flash_rx	The Flash FIT API performs Close for peripheral functions related to flash memory and transitions to the application.
Serial Communication Functions	r_sci_rx	For peripheral functions related to serial communication, Close is performed by the SCI FIT API and the transition is made to the application. For the SCI channels used in the bootloader, refer to the device connection diagram for each product in 6.2 Operating Environment for Demo Project.
Option Setting Memory	-	For the option setting memory, set the same value in the bootloader and the application program.
Other Functions	-	As for the settings of other functions, these are the default values when using the Smart Configurator. The PSW's interrupt enable flag is set to interrupt disabled to transition to the application.

7.2 Notes when using with DATFRX

DATFRX (Flash Memory Data Management Module FIT: R20AN0507) and Firmware Update FIT module can only be used together in dual mode. In this case, the data flash must be excluded from the update target of firmware update FIT module.

Note: DATFRX Rev2.20 or earlier cannot be used together.

7.3 Security measures for the bootloader area

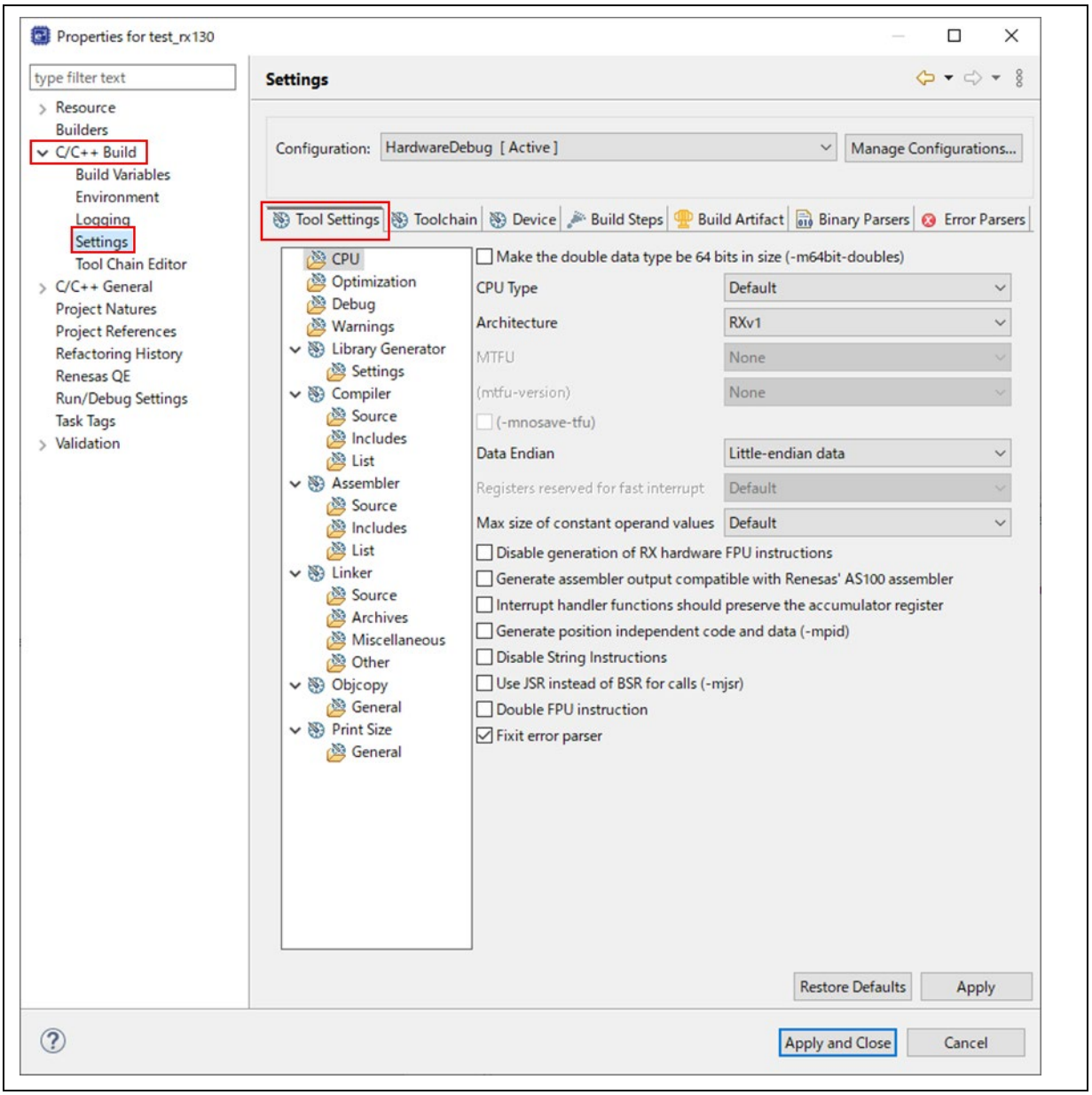
When the firmware update module is commercialized by the customer, it is recommended to protect the area of the code flash where the bootloader (boot_loader) is deployed.

7.4 When creating a new RX130 project in GCC environment

When creating a new RX130 project in GCC environment, the following settings are required.

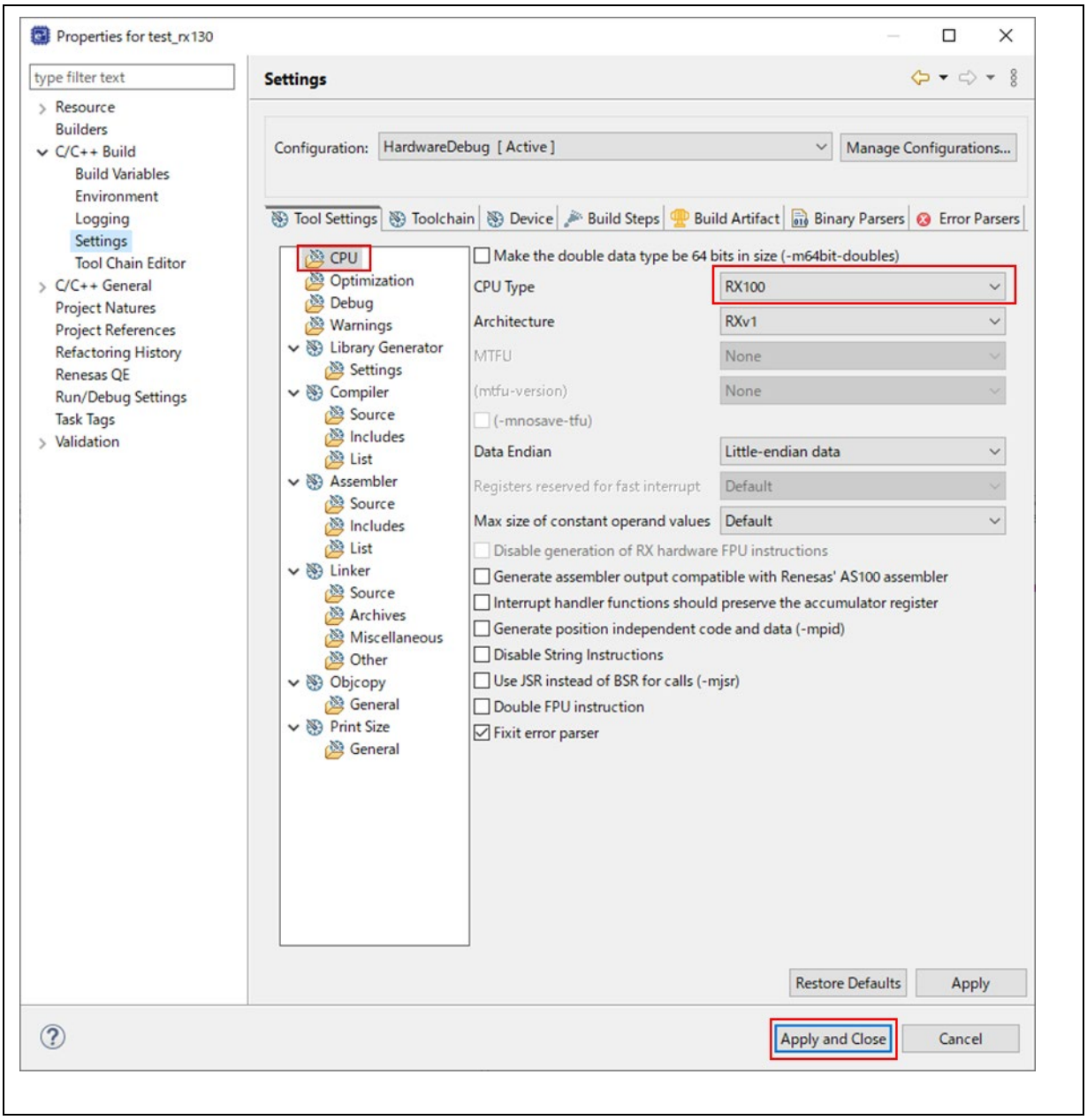
(1) Open the Tool Settings menu of a newly created project in e2 studio.

Project->Properties->C/C++ Build->Settings->Tool Settings



(2) Change the CPU Type from "Default" to "RX100".

Click "CPU" in the tool settings menu, change CPU Type from "Default" to "RX100", and click "Apply and Close".



Revision History

Rev.	Date	Description	
		Page	Summary
2.00	Jul. 20. 2023		First edition issued
2.01	Nov. 17. 2023	1	Added RX66N,RX66T,RX660,RX671,RX72M,RX72N Group to Target Devices
		8	Added device to Supported Update Methods
		14-16	Added device to folder structure
		21	Added FWUP_CFG_CF_W_UNIT_SIZE and FWUP_CFG_FWUPV1_COMPATIBLE to configuration settings
		23-24	Added device in ROM/RAM/Stack
		28	Added parameter to R_FWUP_EraseArea function
		28	Added description to R_FWUP_GetImageSize function
		28	Added description to R_FWUP_GetImageSize function
		29	Added parameter to R_FWUP_WriteImageProgram function
		29	Added return value to R_FWUP_WriteImage function
		30	Added return value to R_FWUP_VerifyImage function
		64,65	Added board used for operation check environment
		66-68	added device to related FIT module version
		88-111	Added device to Operation check environment
		113	Added note
2.02	Mar. 29. 2024	1	Added RX130, RX140, RX230, RX231, RX23E-A, and RX23E-B to Target Devices
		10	Added RX130, RX140, RX230/RX231, RX23E-A, and RX23E-B products to Table 1 2. Also, added products compatible with the sample program.
		14-16	Modify 1.4
		17-18	Modify Table 1.3.
		19	Moved R_FWUP_WriteImageHeader and R_FWUP_WriteImageProgram from Table 1 4 to the end of the table and specified them as for special use.
		21-23	Added config settings to 2.6
		24-27	Updated compiler version in 2.7
		29	Added "About for, while, and do while statements" to 2.11.
		30	Added "Implementation Examples of APIs" in 2.12
		30-34	Moved flow chat to 2.12.1 to 2.12.5
		35-36	Added " Example Implementation of API when Used in Non-blocking Mode" in 2.12.6
		37	Modify the description of "Special Notes" in 3.3.
		42	Modify the description of 3.15
		42	Added 3.15.1
		44-45	Added 3.15.2
		46-47	Added 3.15.3
		52	Modify the description of 4.4
		68-69	Modify the descriptions in Table 5 1 and Table 5 2.
		78-79	Add board to Table 6 1 - Table 6 3
		80-82	Modify Table 6 4 - Table 6 6
		83-117	Added RX130, RX140, RX231, RX23E-A, and RX23E-B to 6.2.1 to 5.
		179-187	Added "How to debug the demo project" in 6.3
		189	Added "Notes when using with DATFRX" to 7.2.
		189	Added "Security measures for the bootloader area" to 7.3
		190-191	Added "When creating a new RX130 project in GCC environment" to 7.4.
2.03	Apr. 15. 2024	-	Modify FWUP FIT version

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

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

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

Notice

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

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Trademarks

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

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/.