RX Family

Internal Flash ROM rewrite program via USB Mass Storage

R01AN3503EJ0105
Rev.1.05
Mar 1, 2021

Overview

This application note explains Flash ROM rewrite program that uses a USB flash drive device.

Target Devices

RX111, RX113, RX231, RX23W
RX62N/RX621, RX63N/RX631, RX63T
RX64M, RX71M, RX65N/RX651, RX66T/RX72T
RX72M, RX72N, RX66N, RX671

When implementing this application note in the user system, conduct an extensive evaluation to ensure compatibility.

Table of Contents

1. Document Outline

This application note explains the Updater used for USB peripheral controllers. Please use in combination with the documents listed in Section **1.2 Related Documents**.

1.1 Functions

This updater updates the user program using the Communication Device Class of the Universal Serial Bus Specification (referred to as USB herein).

1.2 Related Documents

1. Universal Serial Bus Revision 2.0 specification
2. RX Family Flash Module Using Firmware Integration Technology Application Note
3. RX Family Board Support Package Model Application Note
4. User's Hardware Manual corresponding to each MCU

The latest versions of all documents are available for download from the Renesas Electronics website.

Renesas Electronics website

http://www.renesas.com/

USB device page

http://www.renesas.com/prod/usb/

1.3 Cautions

1. The operation of the software described in this application note is not guaranteed. When using the software described in this application note in a system, perform thorough verification of its operation.
2. The Internal Flash ROM rewrite program via USB Mass Storage may not operate properly with some USB flash drive devices.
3. Can not use USB Hub.
4. Make sure to use a USB flash drive device formatted in FAT32 format.
5. The program uses little-endian byte order. If the user program uses big-endian byte order, it is necessary to convert the endianness (byte order) of the program for programUUSming on-chip flash ROM via USB mass storage to big-endian. For information on the endianness setting, see **6.2, Internal Flash ROM rewrite program via USB Mass Storage Settings**.
6. When implementing the program in a system, make sure to refer to **6, Internal Flash ROM rewrite program via USB Mass Storage of and User Program Setting** and **8.4, Cautions**.
7. This program uses each FIT module. In this program, the FTI module source code which is released in Renesas Web is changed for the Firmware Updater.
8. It is necessary to move the following resistance on the RSSK board when using RSSK(RX23W).

   R89 --> R90
   R96 --> R97
   R112 --> R113

9. Please refer to the following about the term "USB0 module" and "USB1 module" described in this documentation.

Term	MCU	USB module name
USB0 module (start address:0xA0000)	RX62N/RX621	USB module
	RX63N/RX631	USBa module
	RX63T	USBa module
	RX65N/RX651	USBb module
	RX64M	USBb module
	RX71M	USBb module
	RX72M	USBb module

	RX72N	USBb module
	RX66N	USBb module
	RX66T/RX72T	USBb module
	RX671	USBb module
	RX111	USBc module
	RX113	USBc module
	RX231	USBd module
	RX23W	USBc module
USB1 module (start address:0xA0200 / 0xD0400)	RX62N/RX621	USB module
	RX64M	USBA module
	RX71M	USBAa module
	RX671	USBb module

1.4 List of Abbreviations and Acronyms

The following lists terms and abbreviations used in this document.

API	:	Application Program Interface
BSP	:	Renesas Board support package module
e^2 studio	:	Eclipse embedded studio (RX-supported)
H/W	:	Renesas USB device
MCU	:	Micro control Unit
MSC	:	Mass Storage Class
P/E	:	Program / Erase
RSK	:	Renesas Starter Kit
RSSK	:	Renesas Solution Starter Kit
USB	:	Universal Serial Bus

2. Overview of Internal Flash ROM rewrite program via USB Mass Storage

2.1 Overview

This program transfers the user program (mot file) stored on a USB flash drive device to the RSK/RSSK via USB (mass storage class), then uses the flash self-programming library to write the transferred user program to a user-defined location in the ROM.

The program is configured as follows:

1. Internal Flash ROM rewrite program via USB Mass Storage
2. Sample program for confirming operation

   This file is used to confirm the operation of the first program. These programs are written in the flash ROM by Internal Flash ROM rewrite program via USB Mass Storage.

The following shows the program's data flow.

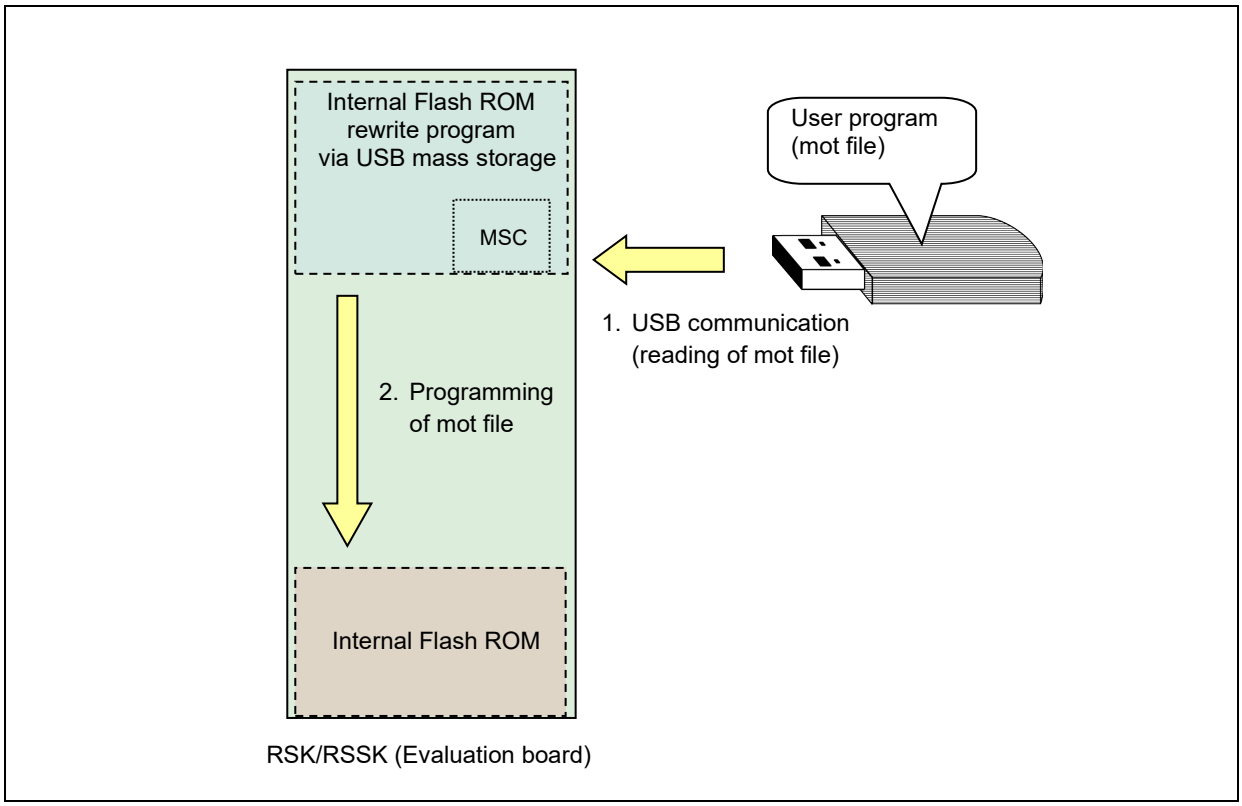


Figure 2-1 Internal Flash ROM Rewrite Program Data Flow

After the RSK/RSSK is activated under the specified conditions, Internal Flash ROM rewrite program via USB Mass Storage is operated when the USB flash drive device is connected, and the user program (mot file) stored in the USB flash drive is written to the internal flash ROM.

2.2 Features

This program has the following features.

1. This program is operated in USB Host mode.
2. This program supports the following USB specifications:
   (1). USB 2.0 specification, Full-speed transfer
   (2). USB mass storage device class
   (3). SFF-8070i mass storage subclass
3. This program erases the internal flash ROM and writes the user program (mot file) to it.
4. This program supports the Motorola S format for the user program.
5. The user program can be allocated to the entire internal flash ROM area, except for the area where this program itself is located.
6. The program supports writing and verification with respect to the Flash ROM
7. The program supports dual mode. (For information regarding dual mode, see the hardware manual of an MCU that supports dual mode.)
8. A backup function is supported. For details of the backup function, refer to section **8.1, Backup Function.**
9. This user program may use all interrupts.

2.3 Software Structure

Figure 2-2 shows the software configuration of Internal Flash ROM rewrite program via USB Mass Storage.

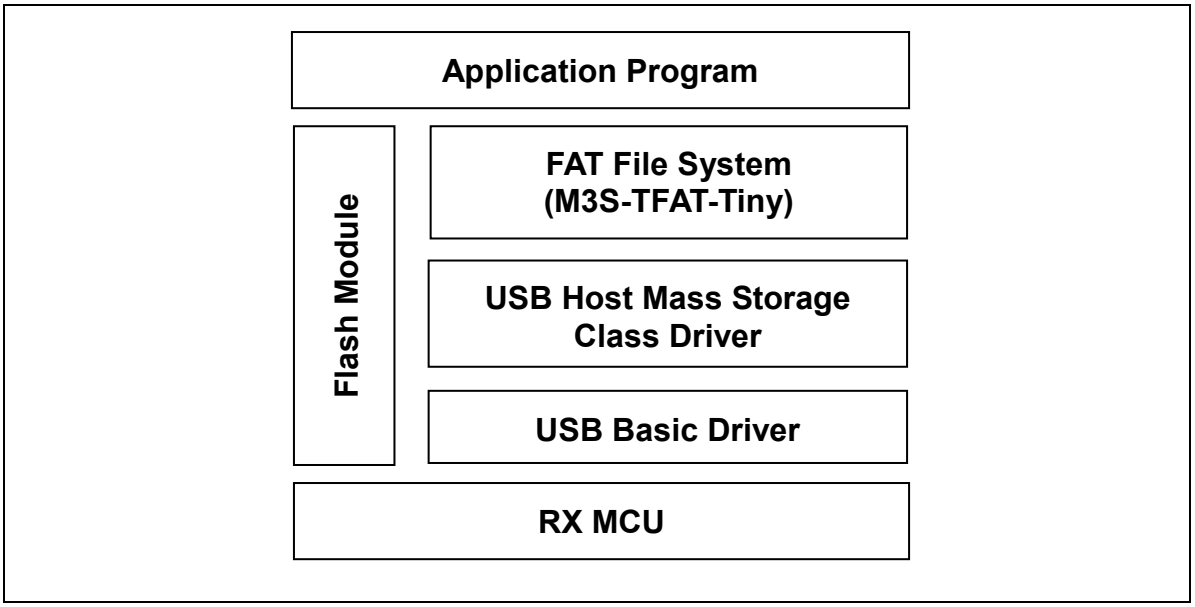


Figure 2-2 Internal Flash ROM rewrite program via USB Mass Storage Software Structure

The application program performs the following processing:

1. Uses the FAT file system to read the user program (mot file) stored on the USB flash drive device.
2. Analyzes the user program (mot file).
3. Uses the flash module to erase the flash ROM and write the user program (mot file) to it.

2.4 ROM Size

The following is ROM size used by this program.

ROM size : 29K bytes

Note:

ROM size is the following when allocating **Internal Flash ROM rewrite program via USB Mass Storage** in the user boot area.

ROM size : 32K bytes

2.5 Target Device & Flash Type

Four types of flash memory are used in RX Family microcontrollers. Refer to the table below to determine the flash type of the microcontroller you are using. For details, see the application note *RX Family Flash Module Using Firmware Integration Technology*.

Table 2-1 MCU Flash Programming Type

Flash Programming Type	Target Device
Flash Type1	RX111, RX113, RX231, RX23W
Flash Type2	RX62N, RX63N/RX631, RX63T
Flash Type3	RX64M, RX71M, RX66T/RX72T
Flash Type4	RX65N/RX651, RX72M, RX72N, RX66N

2.6 Operation Confirmation Environment

Operations for this program have been confirmed under the following environment:

1. Hardware environment

(1).	Evaluation Board	RSK/RSSK
(2).	MCU	RX71M, RX64M, RX63N, RX65N, RX62N, RX63T, RX111, RX113, RX231, RX72T RX72M, RX72N, RX66N, RX23W, RX671
(3).	Emulator	E2 Lite

Note:

RSSK board is used when using RX23W.

2. Software environment

(1).	Integrated Development Environment (IDE)	e^2 studio
(2).	Compiler	RX Family C/C++ Compiler Package CC-RX V.3.01
(3).	FAT File System	Open Source FAT File System [M3S-TFAT-Tiny] Module Firmware Integraion (R20AN0038)
(4).	Flash Library	Flash Module Using Firmware Integration Technology(R01AN2184)

(5).	Host Mass Storage Class	USB Host Mass Storage Class Driver for USB Mini Firmware Firmware Integration Technology (R01AN2169)
(6).	Flash Programming Tool	Renesas Flash Programmer

Note:

a. Operations for this program has not been confirmed when using USB0 module in RX62N.

b. The operation was checked using RX Family C/C++ Compiler Package CC-RX V.3.01 in RX671.

2.7 Folder Configuration

The following is the folder configuration for this program.

```
(Top Directory)
 +—reference
 |       +--SampleProgram (Sample program for operation confirmation)
 |            +-- (MCU name)
 +—workspace (Internal Flash ROM rewrite program via USB Mass Storage Sample projects)
         +-- (MCU name_UserProgSmpl)
```

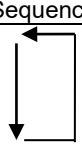
The following provides a description of each folder.

(1). reference¥SampleProgram

This folder stores the sample user program.

sample.mot: LEDs light up in the sequence shown below.

●: Off, ○: On

LED Display				
LED3	LED2	LED1	LED0	Sequence
○	●	●	●	
●	○	●	●	
●	●	○	●	
●	●	●	○	

(2). workspace

This is the folder containing the versions of Internal Flash ROM rewrite program via USB Mass Storage for each MCU.

For details of the folder structure beneath this folder, see **8.3, File/Folder Configuration.**

3. Internal Flash ROM rewrite program via USB Mass Storage Setup

This section explains the setup sequence for this program.

3.1 Project Setup

Select the folder with the name of the MCU you are using from the Workspace folder tab. Set up the project according to the following sequence. This sequence is for setting up with e^2 studio.

1. Start up e^2 studio.

*If running e^2 studio for the first time, the Workspace Launcher dialog box will appear first. Specify the folder which will store the project.

2. Select [**File**] → [**Import**]; the import dialog box will appear.
3. In the Import dialog box, select [**Existing Projects into Workspace**].

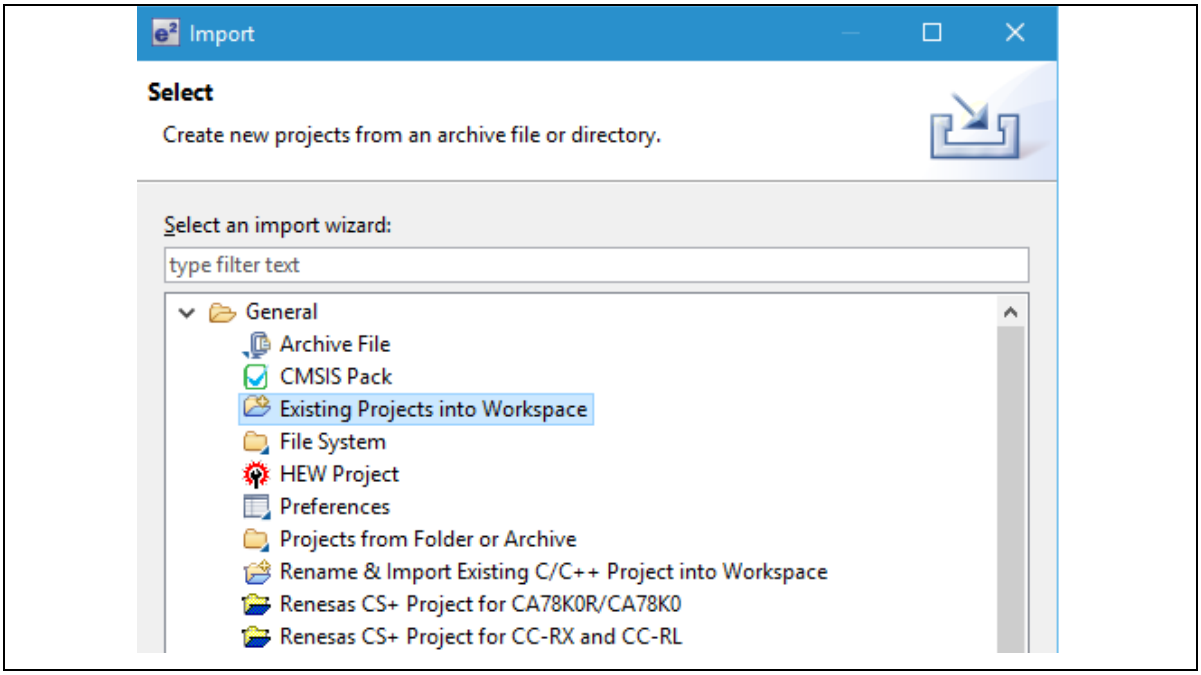


Figure 3-1 Select Import Source

4. Press [**Browse**] for [**Select root directory**]. Select the folder in which [.cproject] (project file) is stored.

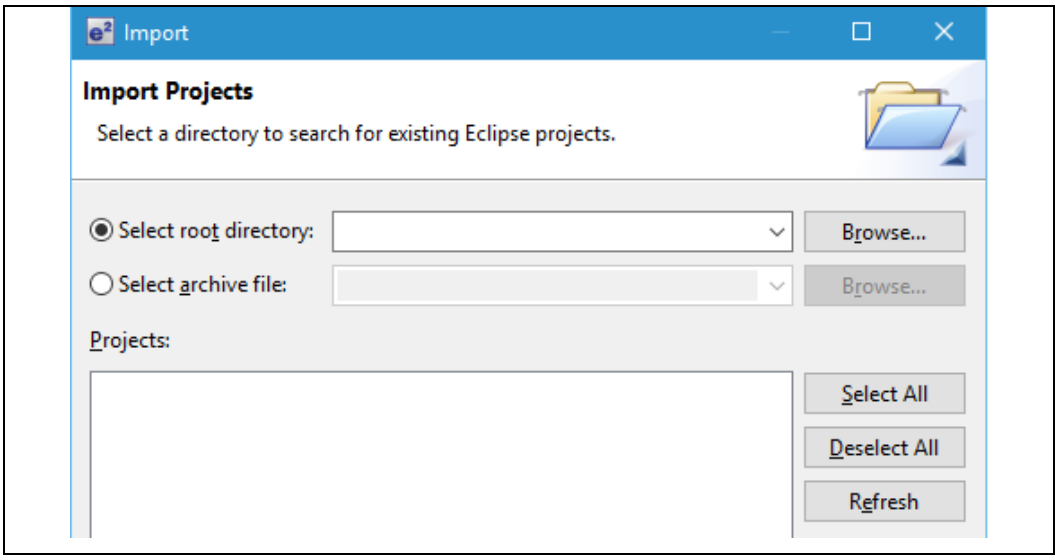


Figure 3-2 Project Import Dialog Box

5. Click [**Finish**].

This completes the step for importing a project to the project workspace.

Note:

Please change to the device for linear mode from "Change Device:" (red frame) in **Figure 3-3** when using MCU supporting dual mode is used as linear mode. For example, please change from "R5F565NEHxFB_DUAL" (Dual mode) to "R5F565NEHxFB" (Linear mode) when using the device (R5F565NEHxFB)

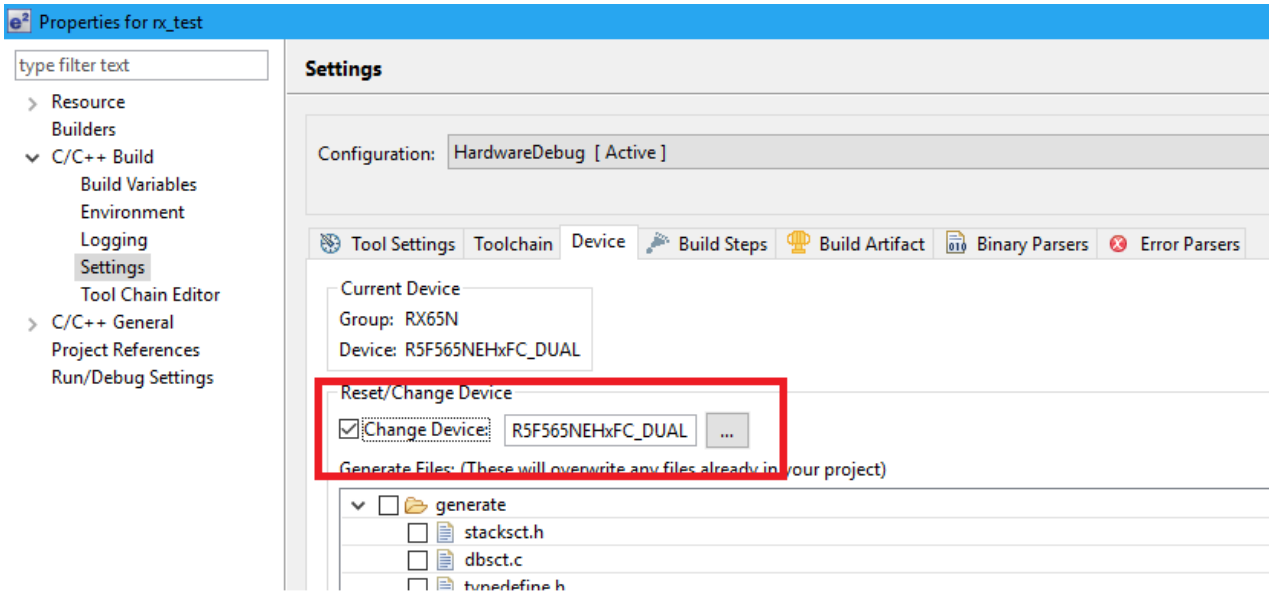


Figure 3-3 Project Import Dialog Box

4. Execute Internal Flash ROM rewrite program via USB Mass Storage

This section describes how to execute this program.

Here is described the procedure for running the sample program for confirming operation, using the RSK/RSSK board.

4.1 Writing Internal Flash ROM rewrite program via USB Mass Storage to Flash ROM write and execution

This section explains the sequence for writing and executing the Internal Flash ROM rewrite program.

4.1.1 Writing Internal Flash ROM rewrite program via USB Mass Storage to ROM

(1). Hardware setup

The following figures show connection diagrams for writing Internal Flash ROM rewrite program via USB Mass Storage to the MCU.

a. Using an emulator

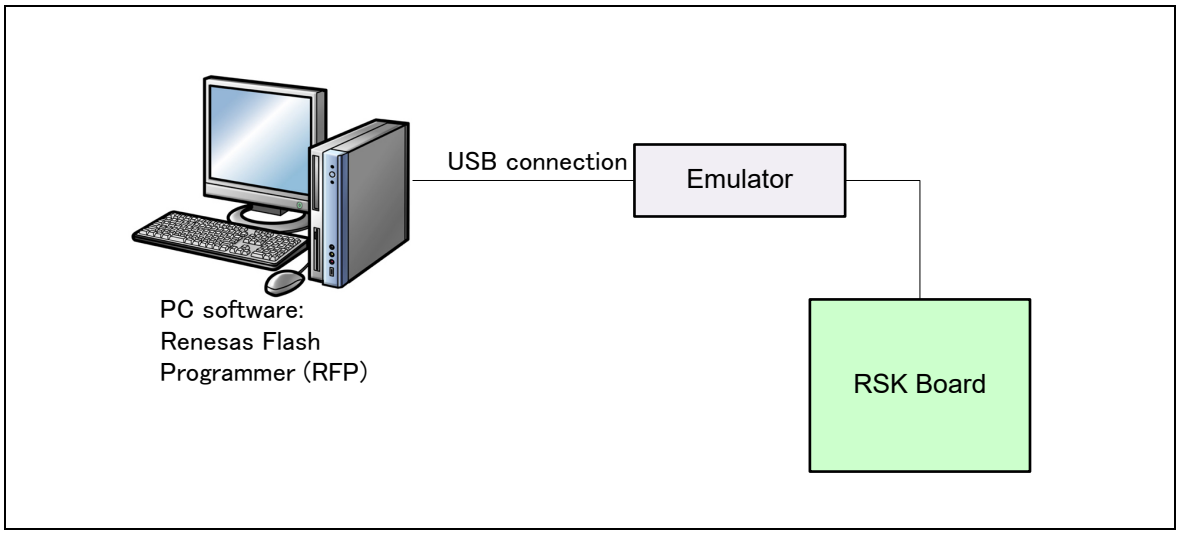


Figure 4-1 Connection Diagram Using an Emulator

b. Not using an emulator

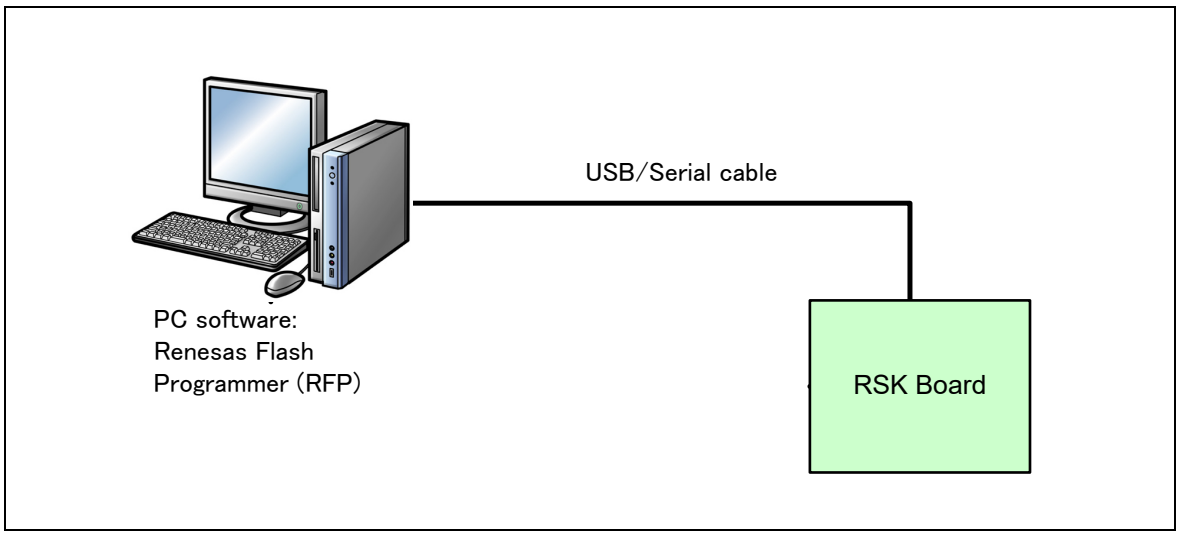


Figure 4-2 Connection Diagram with No Emulator

(2). Writing the Internal Flash ROM rewrite program via USB Mass Storage

Run the Renesas Flash Programmer (RFP) and, using the [**Browse**] for [Program File] button, select Internal Flash ROM rewrite program via USB Mass Storage file to be written from the Workspace/(MCU *name*) folder. Press **Start** to download the program to the target board. The write operation is complete when **OK** is displayed.

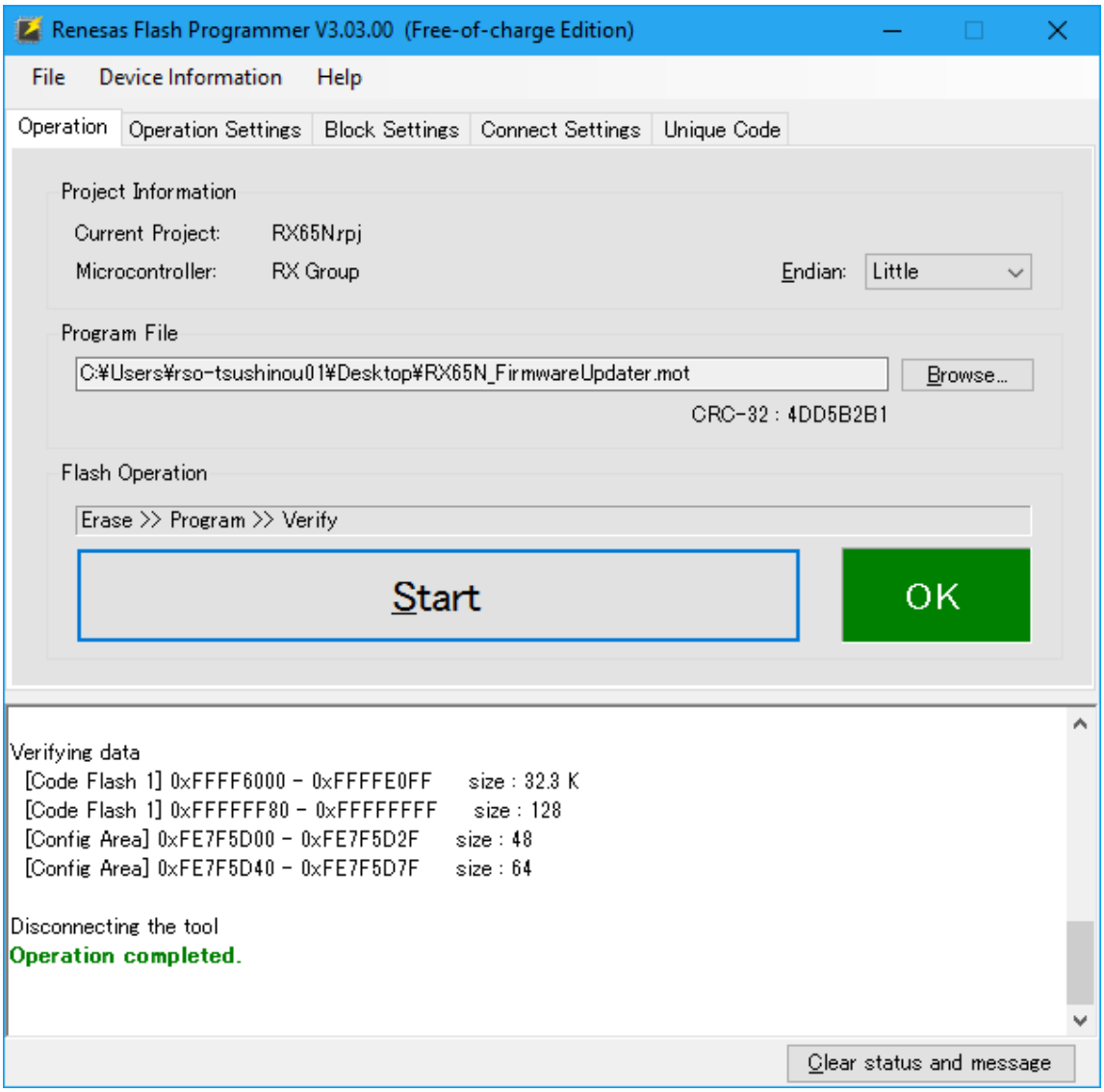


Figure 4-3 File Specification

Notes:

a) Refer to the following URLs for more details on the Renesas Flash Programmer:
URL:
https://www.renesas.com/en-us/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html

b) Refer to section **4.1.2, Allocation of Internal Flash ROM rewrite program via USB Mass Storage** for more details concering positioning of Internal Flash ROM rewrite program via USB Mass Storage.

c) Please make settings to erase all blocks in the internal Flash ROM (program ROM Area).

(3). Copying the Flash ROM rewrite program to the update target area (when dual mode is selected)

After writing of Internal Flash ROM rewrite program via USB Mass Storage, as described in step (2), is complete, the Internal Flash ROM rewrite program via USB Mass Storage in the startup bank will copy itself to the update target area when the RSK/RSSK powered-on or reset.

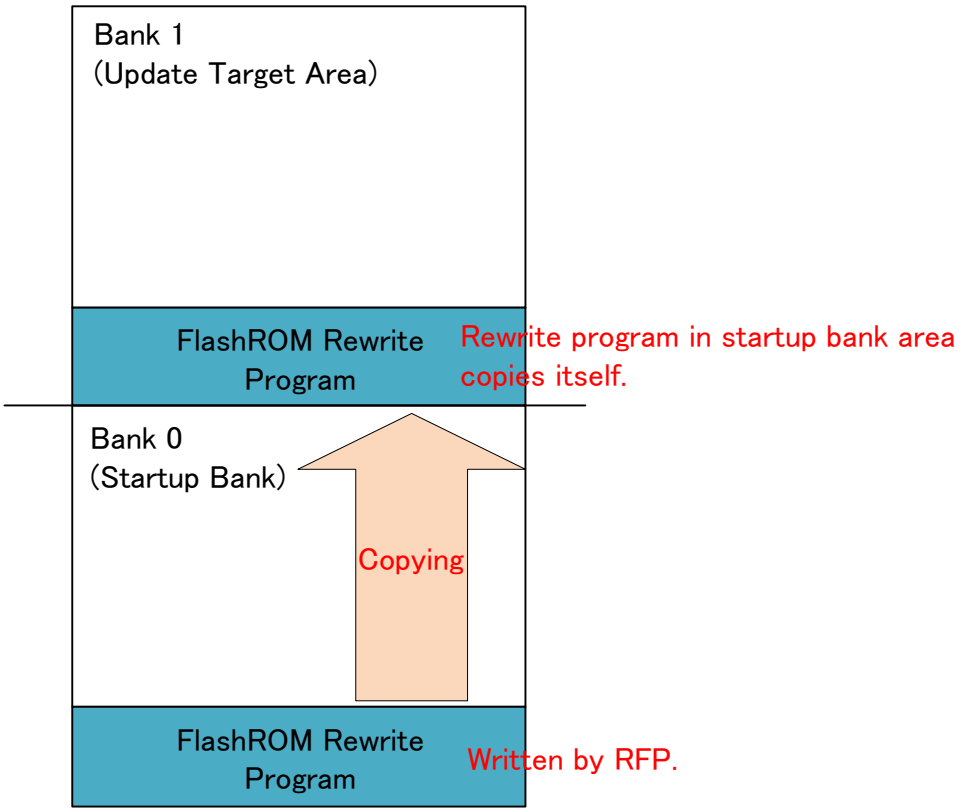


Figure 4-4 Copying the Flash ROM Rewrite Program

Note:

If this copy process fails, the following lighting pattern will blink.

●: Off ○: Blink

LED Display			
LED3	LED2	LED1	LED0
○	○	○	●

4.1.2 Allocation of Internal Flash ROM rewrite program via USB Mass Storage

This section explains the assigned address of this program.

(1). Assignment to ROM area other than user boot area

Allocate Internal Flash ROM rewrite program via USB Mass Storage in the following area.

Alloation Areas for Internal Flash ROM rewrite program via USB Mass Storage		
0xFFFF8000	-	0xFFFFFFFF

Internal Flash ROM rewrite program via USB Mass Storage copies itself to the following area when using dumo mode.

Table 4-1 Case of ROM capacities 2MB

Alloation Areas for Internal Flash ROM rewrite program via USB Mass Storage		
0xFFEF8000	-	0xFFEFFFFF

Table 4-2 Case of ROM capacities 4MB

Alloation Areas for Internal Flash ROM rewrite program via USB Mass Storage		
0xFFDF8000	-	0xFFDFFFFF

The following shows the memory map for RX63N. For more details, refer to the user's hardware manual corresponding to the target MCU.

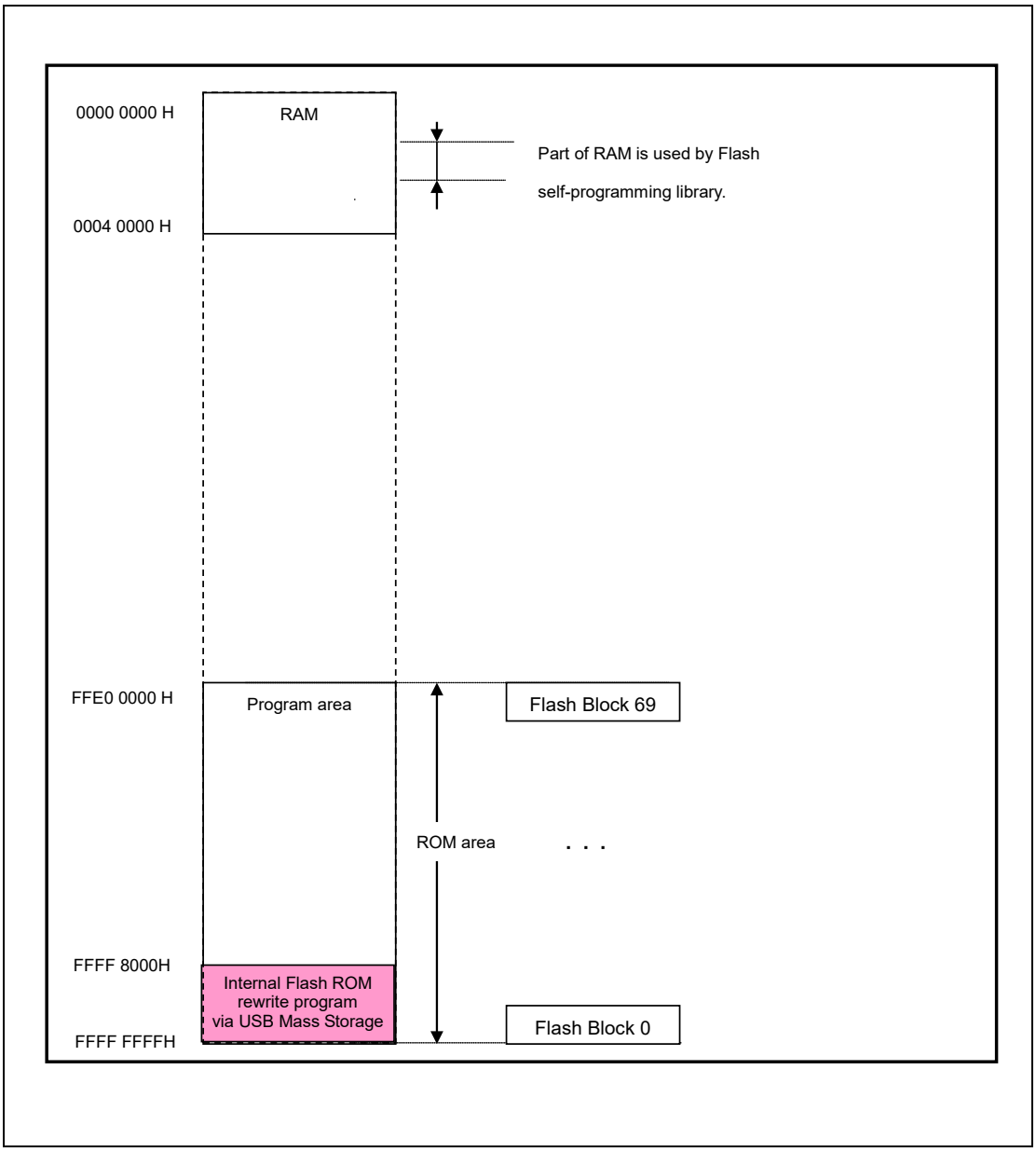


Figure 4-5 Memory Map (user boot area not used and dual mode not used)

The momory map when using dual mode is shown below.

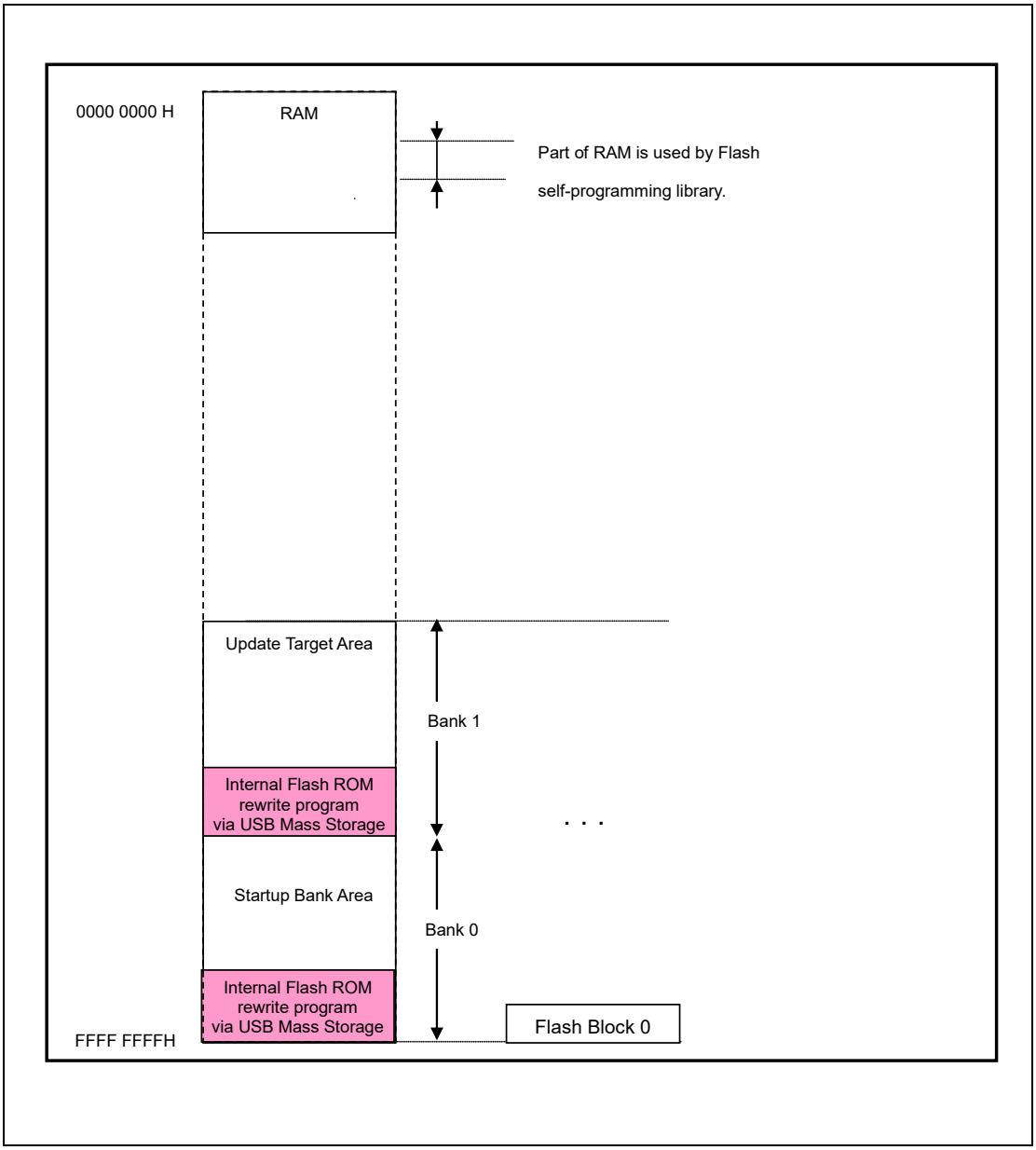


Figure 4-6 Memory Map (Using Daul Mode)

Notes:

1. When compiling **Internal Flash ROM rewrite program via USB Mass Storage**, select 24 bits as the [Function branch width] in e^2 studio. To specify the [Branch width size], select [File] → [Properties] → [C/C+ Build] → [Settings], specify [Common] → [CPU].
2. You can not assign **Internal Flash ROM rewrite program via USB Mass Storage** in the user boot area when using dual mode.

(2). Assignment to ROM area other than user boot area

Internal Flash ROM rewrite program via USB Mass Storage can be assigned to the user boot area when using RX71M, RX64M or RX66T/RX72T. Table 4-3 provides user boot area information.

Table 4-3 MCU User Boot Area Information

MCU	Size	user boot address		
RX71M RX64M RX66T/RX72T	32KB	0xFF7F8000	-	0xFF7FFFFF

Note:

1. When compiling **Internal Flash ROM rewrite program via USB Mass Storage**, select [None] as the [Branch width size] in e^2 studio. To specify the [Branch width size], select [File] → [Properties] → [C/C+ Build] → [Settings], specify [Common] → [CPU].
2. You can not assign in the user boot area when using RX62N/RX621, RX63N/RX631 or RX63T.

The following shows the memory map when Internal Flash ROM rewrite program via USB Mass Storage is assigned to the user boot area in RX64M.

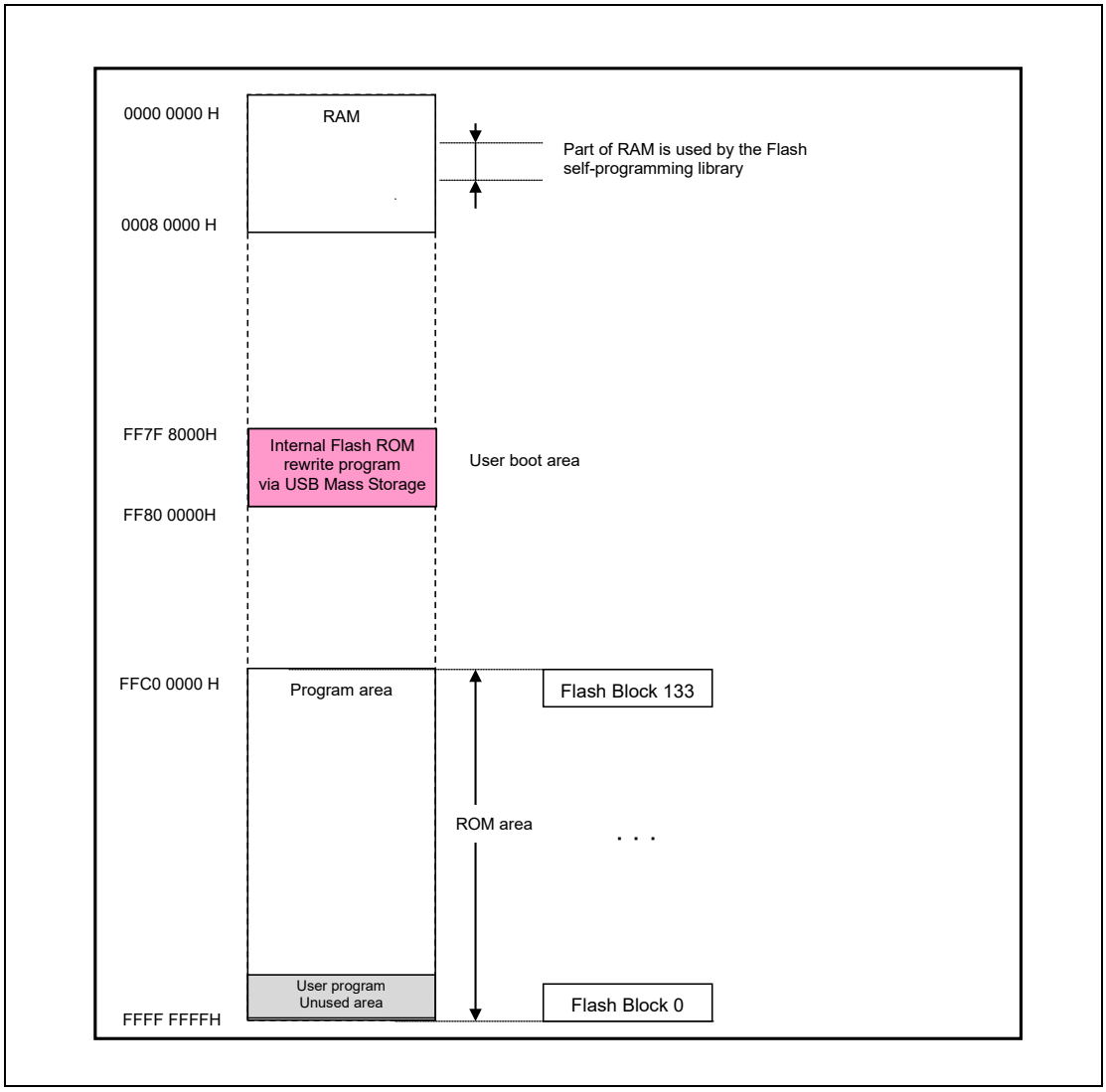


Figure 4-7 Memory Map (when using user boot area)

4.2 Execution of Internal Flash ROM rewrite program via USB Mass Storage (user program programming operation)

This section explains the sequence for **Internal Flash ROM rewrite program via USB Mass Storage** execution and user program (mot file) write operation.

1. Operating Internal Flash ROM rewrite program via USB Mass Storage

Press the reset button on RSK/RSSK while holding down the switch (RSK: Switch 3, RSSK: Switch 2). **Internal Flash ROM rewrite program via USB Mass Storage** starts, and then waits for a USB flash drive device to be connected. LED0 on RSK/RSSK blinks at this time. If the issue is generated in the initialization processing for FlashROM and USB module, LED1, LED2 and LED3 on RSK/RSSK blink. Even if USB flash drive device is connected when these LEDs blink, Internal Flash ROM rewrite program via USB Mass Storage does not work properly.

2. Connecting USB flash drive Device

Connect to the RSK/RSSK the USB flash drive device on which the user program (mot file) is stored. Internal Flash ROM rewrite program via USB Mass Storage starts programing the user program (mot file) to the flash ROM.

(1). Erasing the internal flash ROM.

During erasing the internal flash ROM, the following LEDs (LED0 and LED1) on the RSK/RSSK board light on.

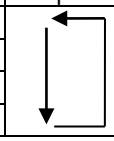
●: Off, ○: On

LED Display			
LED3	LED2	LED1	LED0
●	●	○	○

(2). During Programing of the User Program (mot File)

During the processing to program the flash ROM, the LEDs on the RSK/RSSK board illuminate in a shifting pattern.

●: Off, ○: On

LED Display				
LED3	LED2	LED1	LED0	Sequence
●	●	●	○	
●	●	○	●	
●	○	●	●	
○	●	●	●	

Note:

a. Don't detach the USB flash drive while programming the user program. If the USB flash drive was detached, you need to reset the RX MCU.

b. The LED shift illuminating does not start for about 15 seconds at maximum after connecting the USB flash drive since **Internal Flash ROM rewrite program via USB Mass Storage** does not process to the LED during erasing the Flash ROM

(2). Completion of Programming of User Program (mot File)

When processing to program the flash ROM completes (normal end), **Internal Flash ROM rewrite program via USB Mass Storage** starts up the user program written in the Flash ROM by the software reset.

Note:

The user program is started up by resetting RSK/RSSK board after completion of writing the user program (mot file) the since RX62N does not support the software reset when using RX62N. When processing to program the flash ROM completes (normal end). the LEDs on the RSK/RSSK board illuminate in a blinking pattern.

●: Off ○: On

LED Display				
LED3	LED2	LED1	LED0	Sequence
●	●	●	●	
○	○	○	○	

(3). User Program (mot File) Programming Error

If processing to program the flash ROM ends with an error, the LEDs flash on and off in one of the following patterns. If one of the patterns shown below is displayed, restart Internal Flash ROM rewrite program via USB Mass Storage.

●: Off ○: On

LED Display				Description
LED3	LED2	LED1	LED0	
●	●	●	○	No accessible drive detected.
●	●	○	●	Drive mount processing failed.
●	●	○	○	File open processing failed. (Note a)
●	○	●	●	File read processing failed. (Note d)
●	○	●	○	File close processing failed.
●	○	○	●	Checksum error
●	○	○	○	Format error
○	●	●	●	Flash ROM erase error
○	●	●	○	Flash ROM write error
○	●	○	●	Address error (Note b)
○	●	○	○	Write data verify error
○	○	●	●	File end error (Note c)
○	○	○	●	Other error (Note e)

Notes:

a. An error occurs if the file name of the user program (mot file) is in other than 8.3 format.

b. An error occurs if write data with a destination other than the write area is received.

c. An error occurs if no S type format end code is received, even through the FAT library has detected the file end.

d. An error occurs if the USB flash drive device is detached while processing to erase the flash ROM or write the user program to it is in progress.

e. One of the following two errors:

a) The security code contained in the user program (mot file) does not march the security code registered in Internal Flash ROM rewrite program via USB Mass Storage.

b) The option setting memory area is contained within the user program (mot file).

4.3 Cautions Regarding User Program Write Operation

1. If you write the user program to the area which already contains **Internal Flash ROM rewrite program via USB Mass Storage**, please start over by re-writing **Internal Flash ROM rewrite program via USB Mass Storage**.

   *Note that the ROM erase block unit differs depending on the MCU.

2. When a momentary power interruption or MCU reset occurs during erasing or writing to the interna flash ROM, the works of the section **4.1.1, Writing Internal Flash ROM rewrite program via USB Mass Storage to ROM** may be necessary again.

3. Be careful not to erase any block that includes fixed vectors. **Internal Flash ROM rewrite program via USB Mass Storage**

will not run if the fixed vectors have been erased.

5. Cautions Regarding Creating the User Program

This sections explains cautions that apply when creating the user program

5.1 UserApp Header Area (user application header)

When using this program to write a user program, you must include a UserApp Header (user application header) area in the user program. The size of the UserApp Header area should be a total of 8 bytes: 4 bytes for the user program start address storage area and 4 bytes for the security code storage area (see Figure 5-1).

Refer to section **6.1, User Program Setting**, for details on how to create the UserApp Header area.

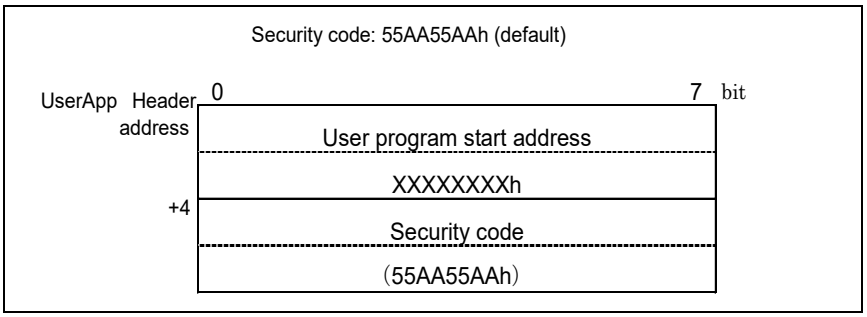


Figure 5-1 UserApp Header Area

This header information is read when **Internal Flash ROM rewrite program via USB Mass Storage** is started up and transitions to the UserApp startup sequence. For details, refer to section **8.2.1, Power On / Reset Operation Flow.**

5.2 Option-Setting Memory

Do not make any settings to the option setting memory in the user program (mot file). If a user program that makes settings to the option setting memory is written to the flash ROM, a write error occurs. The child write error can be confirmed from the illumination pattern of the LEDs on the RSK/RSSK board. For the LED illumination patterns, refer to **4.2, Execution of Internal Flash ROM rewrite program via USB Mass Storage (user program programming operation)**, Running **Internal Flash ROM rewrite program via USB Mass Storage** (Programming the mot File).

Note:

Make settings to option setting memory in Internal Flash ROM rewrite program via USB Mass Storage. For details, see **6.2, Internal Flash ROM rewrite program via USB Mass Storage Settings**.

5.3 Fixed Vectors

Do not include fixed vector area in the user program (mot file).

Note:

The fixed vectors of the **Internal Flash ROM rewrite program via USB Mass Storage** will be used.

5.4 File Name Of User Program (mot File)

Internal Flash ROM rewrite program via USB Mass Storage only supports user program (mot file) file names in 8.3 format. If the file name is not in 8.3 format, Internal Flash ROM rewrite program via USB Mass Storage illuminates the LEDs on the RSK/RSSK board to indicate an error, and write processing of the user program (mot file) does not take place.

Note:

For the LED illumination patterns indicating errors, refer to **4.2, Execution of Internal Flash ROM rewrite program via USB Mass Storage (user program programming operation)**.

5.5 Section Setting When Using Backup Function

The user program is run from Area 1, so Area 1 should be specified in the section settings for the user program code attribute and romdata attribute when you build your project.

code attribute : Stored execution code.
romdata attribute : Stored rom data

Note:
For the buckup function, refer to section **8.1, Backup Function.**

5.6 File Format

The only file format supported by **Internal Flash ROM rewrite program via USB Mass Storage** is Motorola type S (S3 format).

Note:

For the LED illumination patterns indicating errors, refer to refer to **4.2, Execution of Internal Flash ROM rewrite program via USB Mass Storage (user program programming operation)**.

5.6.1 Notes

1. Record Configuration

Figure 5-2 shows the record configuration supported by the Internal Flash ROM rewrite program via USB Mass Storage. S type format files configured using a sequence other than that shown in Figure 5-2 are not supported.

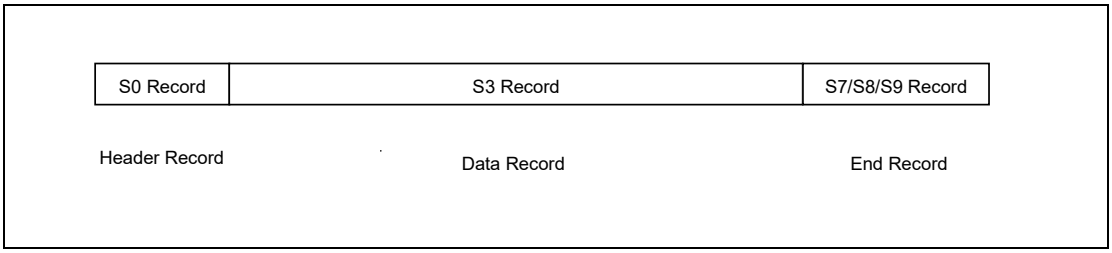


Figure 5-2 Record Configuration Supported by Internal Flash ROM rewrite program via USB Mass Storage

2. Load Addresses

Internal Flash ROM rewrite program via USB Mass Storage only supports S type format with the load addresses in ascending order. Do not use S type format files with addresses in descending order, or addresses before and after.

3. Error Detection

Internal Flash ROM rewrite program via USB Mass Storage detects an error if the received S type format is incorrect.

(1). Checksum Error

Checksum verification takes place for each record of the S type format, and a checksum error is detected if the verification result is incorrect.

(2). Format Error

A format error is detected if any of the following conditions apply to the S type format:

a. An unsupported record (S1, S2, S4, S5, S6) is detected.

b. The header record (S0) is detected twice.

c. A data record (S3) or end record (S7, S8, or S9) is detected before the header record (S0) is detected.

(3). Address Error

If write data with a destination outside the download area is received, an address error is detected.

6. Internal Flash ROM rewrite program via USB Mass Storage of and User Program Setting

6.1 User Program Setting

1. Setting Content 1

Create the UserApp Header area in the user program according to the sample in Figure 6-1. For more details about the UserApp Header, see section **5.1, UserApp Header Area (user application header)**.

2. Setting Content 2

Specify a section for the UserApp Header area created in step 1, above, and make sure to allocate this section to the start of the user program.

```
/*********************************************************************************
   APPLICATION INTERFACE HEADER
   The purpose of the header is for an external application to be able to read
   certain values from known addresses.
   - Start address of UserApp.
   - Security code must match what PCDC Flashloader expects.
   - For revision purposes of applications etc.
   - Do not change the order of these variables!
*********************************************************************************/
#pragma section C UserApp_Head_Sect

/* START ADDRESS of user application header data    - Appheader address + 0x00. */
const uint32_t  userapp_entry_addr =    (uint32_t)  userprog_start;

/*                                                  - Appheader address + 0x04. */
const uint32_t  userapp_sec_code =     (uint32_t)  USERAPP_SECURITY_CODE;
```

Section specification

Header information

Figure 6-1 UserApp Header Code Example

Sequence:

First select [Properties] → [C/C+ Build] → [Settings]. Next, select the Tool setting tab, and select [Linker] → [Section].

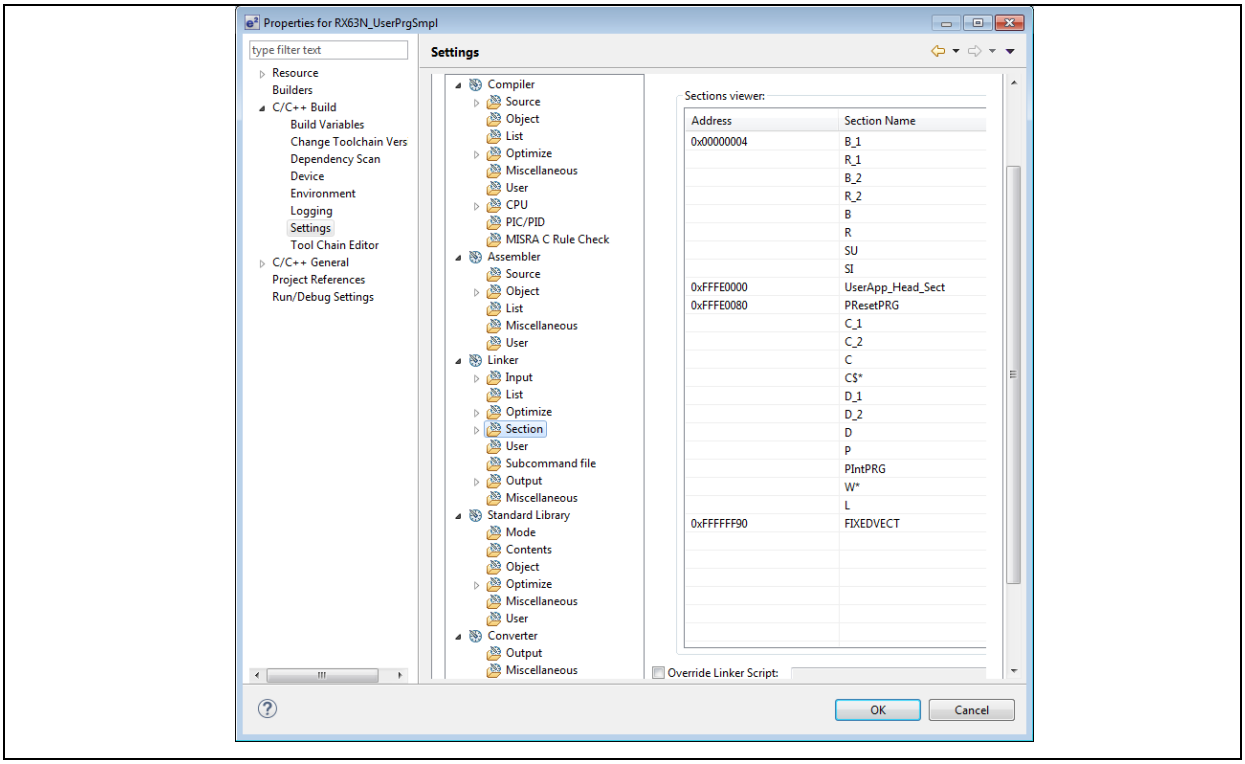


Figure 6-2 Allocation of UserApp Header Area

6.2 Internal Flash ROM rewrite program via USB Mass Storage Settings

1. Setting Content 1

Adjust the following definition setting to your system. The following definition is described in "r_config¥r_usb_updater_config.h" file.

(1). Specifying the Name of the User Program (mot File)

For the definition *USB_CFG_USER_PROGRAMNAME*, specify the name of the user program (mot file) to be written to the flash ROM area.

```
#define        USB_CFG_USER_PROGRAMNAME      "sample.mot"          // User program name
```

Notes:

a. Enclose the name of the user program (mot file) in double quotes.

b. Only user program (mot file) file names in 8.3 format are supported.

(2). USB module setting

Specify the USB module number as the *USB_CFG_USE_USBIP* definition. When using the USB0 module, set *USB_CFG_IP0* as the *USB_CFG_USE_USBIP* definition; when using the USB1 module, set *USB_CFG_IP1*.

```
#define        USB_CFG_USE_USBIP     USB_CFG_IP0        // USB0 module using setting
#define        USB_CFG_USE_USBIP     USB_CFG_IP1        // USB1 module using setting
```

Note:

If the target MCU supports only one USB module, set *USB_CFG_IP0* as the *USB_CFG_USE_USBIP* definition.

(3). Backup Function Settings

Specify whether or not the backup function will be used as the *USB_CFG_BACKUP* definition. To use the backup function set the definition to *USB_CFG_ENABLE*; set the definition to *USB_CFG_DISABLE* if the backup function will not be used.

```
#define        USB_CFG_BACKUP     USB_CFG_ENABLE     // Backup function is used.
#define        USB_CFG_BACKUP     USB_CFG_DISABLE    // Backup function is not used.
```

Note:

For details of the backup function, refer to section **8.1, Backup Function.**

(4). Input System Clock Frequency setting

Specify 20MHz setting or 24MHz setting to the Input system clock frequency bit (*CLKSEL*) in *PHYSET* register.when using USBAa/USBA module.

```
#define        USB_CFG_CLKSEL        USB_CFG_24MHZ      // 24MHz setting
#define        USB_CFG_CLKSEL        USB_CFG_20MHZ      // 20MHz setting
```

Note:

This definition is ignored when using USB module except USBAa/USBA module supported by RX71M/RX64M.

(5). CPU buswait setting

Specify the value to the following definition (*USB_CFG_BUSWAIT*).

This value is set to BUSWAIT register in USBA/USBAa module.

```
#define        USB_CFG_BUSWAIT        7                  // 7 wait setting
```

Notes:

a. Refer to the RX71M/RX64M hardware manual about the value which is set to *USB_CFG_BUSWAIT* definition.

b. This definition is ignored when using USB module except USBAa/USBA module supported by RX71M/RX64M.

(6). USB regulator setting

Specify whether your system uses USB regulator function supported by RX231 or not.

```
#define     USB_CFG_REGULATOR     USB_CFG_OFF     // No use
#deefine    USB_CFG_REGULATOR     USB_CFG_ON      // Use
```

Note:

This definition is ignored when using MCU except RX231.

(7). Other setting

Internal Flash ROM rewrite program via USB Mass Storage references the UserApp Header area in the user program. Therefore, if you change the assigned address of the UserApp Header area, make sure you also change this program to reference the revised UserApp Header area. In the same manner, if you change the security code value, make sure you make the corresponding changes in this program. Refer to section 5.1, **UserApp Header Area (user application header)** about UserApp Header area.

a. **USB_CFG_USERAPP_HEADER_ADDR** definition setting

Set the assigned address of the UserApp Header area to the *USERAPP_HEADER_ADDR* definition in the main.c file.

```
#define     USB_CFG_USERAPP_HEADER_ADDR          Assigned address of UserApp Header area
```

b. **USB_CFG_USERAPP_SECURITY_CODE** definition setting

Set the security code (4-byte data) specified in the UserApp Header area to the *USERAPP_SECURITY_CODE* definition in the main.c file.

```
#define     USB_CFG_USERAPP_SECURITY_CODE        Security code
```

Note:

Specify the value other than 0xFFFFFFFF to the security code.

2. Setting Content 2

When using an MCU that supports dual mode, specify either 0 (dual mode) or 1 (linear mode) in the definition of *BSP_CFG_CODE_FLASH_BANK_MODE* in the r_config¥r_bsp_config.h file.

```
#define     BSP_CFG_CODE_FLASH_BANK_MODE     0     // Dual mode
#define     BSP_CFG_CODE_FLASH_BANK_MODE     1     // Linear mode
```

3. Setting Content 3

(1). This program jumps to Internal Flash ROM rewrite program via USB Mass Storage or the user program depending on the state of the switch (RSK: Switch3, RSSK: Switch2) on the RSK/RSSK. The process for determining the state of SW depends on the board specifications. Please adjust the determination process to meet the target board specifications. This determination process is performed in the main function.

```
void main(void)
{
  if (SW3 != SW_ACTIVE)
   {
     if(USER_PROG_WRITE_OK == fu_user_prog_start())
      {
         usb_cpu_int_disable();
         jump_to_userapp();
      }
   }
                :
}
```

Judging whether SW3 is pushed or not

Figure 6-3 main function processing

(2). Internal Flash ROM rewrite program via USB Mass Storage uses the LED on RSK/RSSK. If you mount Internal Flash ROM rewrite program via USB Mass Storage in your system, delete the following function and the calling processing of this function as necessary.

Function Name : USB_RSK_LED_SET

4. Setting Content 4 (option setting memory)

The option setting memory can only be used to set the following items. Set all other items to the default values.

(1). FASTSTUP bit

(2). LVDAS / STUPLVD1REN bit

(3). VDSEL / STUPLVD1LVL bit

(4). MDE bit

Note that the updater does not write operation to the ROM in the user program's option setting memory. Because the firmware update program option setting memory is also used by the user program, set the option setting memory in accordance with the firmware update program.

Notes:

a. The initial settings for the firmware update option setting memory are all the default values.

b. RX62N does not support the option setting memory.

c. For more details about the option setting memory, refer to the hardware version of the target MCU user's manual.

5. Setting Content 5 (compile option)

Set the following compile options.

(1). When assigning Internal Flash ROM rewrite program via USB Mass Storage to a ROM area other than the user boot area:

Select **Compiles within 24 bits** as **Branch width size** in the e^2 studio

(2). When assigning Internal Flash ROM rewrite program via USB Mass Storage to the user boot area:

Select **None** as **Branch width size** in the e^2 studio

Note:

To specify the [Branch width size], select [File] → [Properties] → [C/C+ Build] → [Settings], specify [Common] → [CPU].

6.3 USB Pin Setting for Internal Flash ROM rewrite program via USB Mass Storage

Make USB pin setting according to your system. USB pin setting processing is described in the following function.

Function Name	File Name
usb_pin_setting	demo_src¥main.c

6.4 User Program Allocation

Make sure you assign the user program to ROM area which does not overlap with the area written by Internal Flash ROM rewrite program via USB Mass Storage.

Notes:

1. Specify settings such that the user program will be placed in the ROM areas below. In addition, when dual mode is selected, specify settings such that the user program will be placed in the startup bank area.

Backup Function	User Program Allocation Area		
OFF	On-chip ROM Area (Program ROM) Start Address	-	0xFFFF7FFB
ON	Start Address of Program Execution Area	-	0xFFFF7FFB

Note:

For the backup function and the program execution area , refer to section **8.1, Backup Function.**

2. All Flash ROM area other than Internal Flash ROM rewrite program via USB Mass Storage is erased before the user program (mot file) is programmed in the Flash ROM.
3. Although the Flash self-programming library occupies part of the RAM area, it is only used when executing Internal Flash ROM rewrite program via USB Mass Storage and will not affect the user program operations.
4. The Flash ROM area (4 bytes) of 0xFFFF7FFC to 0xFFFF7FFF is used as the management area by the Internal Flash ROM rewrite program via USB Mass Storage

7. Writing the User Program to USB flash drive

Copy the user program (mot file) to the root folder of a USB flash drive device formatted in FAT32 format.

Notes:

1. Internal Flash ROM rewrite program via USB Mass Storage writes to the flash ROM the user program specified by the *USB_CFG_USER_PROGRAMNAME* definition in the r_config¥r_usb_updater_config.h file.
2. If the user program specified by the *USB_CFG_USER_PROGRAMNAME* definition is not present in the root folder of the USB flash drive device, programming of the flash ROM does not occur.
3. For information on the *USB_CFG_USER_PROGRAMNAME* definition, see **6.2, Internal Flash ROM rewrite program via USB Mass Storage Settings**
4. Make sure to use a USB flash drive device formatted in FAT32 format.
5. Only user program (mot file) file names in 8.3 format are supported.

8. Internal Flash ROM rewrite program via USB Mass Storage Explanation

8.1 Backup Function

Internal Flash ROM rewrite program via USB Mass Storage supports a backup function that launches the user program stored in the specific area if overwriting of the flash ROM fails, for example due to USB transfer failure etc while the overwriting of the flash ROM is in progress.

An outline of the flash ROM overwrite processing of the backup function is presented below.

1. **Internal Flash ROM rewrite program via USB Mass Storage** divides the on-chip flash ROM (program ROM area) into two areas and uses the first (Area 1) as a program execution area and the second (Area 2) as a user program storage area. The division between Area 1 and Area 2 is located at the center of the on-chip flash ROM area. These two ROM areas are the same size. In addition, Area 2 contains an unused area because **Internal Flash ROM rewrite program via USB Mass Storage**, which is present in Area 1, is not present in Area 2.

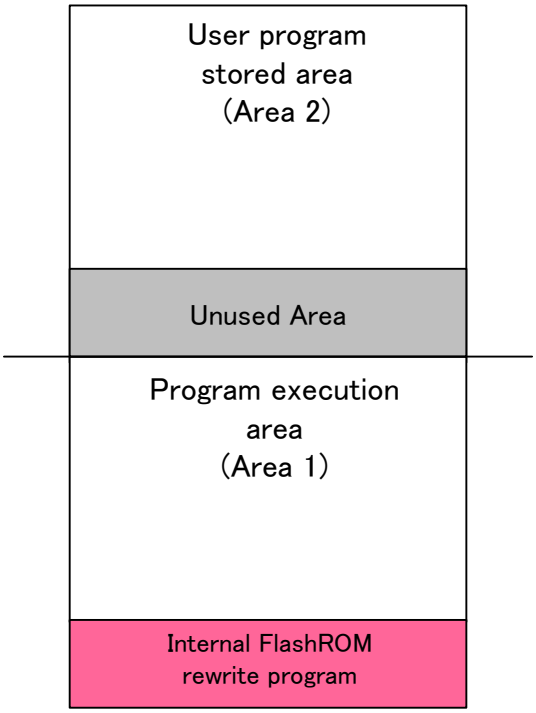


Figure 8-1 Flash ROM Area When Using Backup Area

2. When the backup function is enabled, **Internal Flash ROM rewrite program via USB Mass Storage** will always write the user program to Area 2.

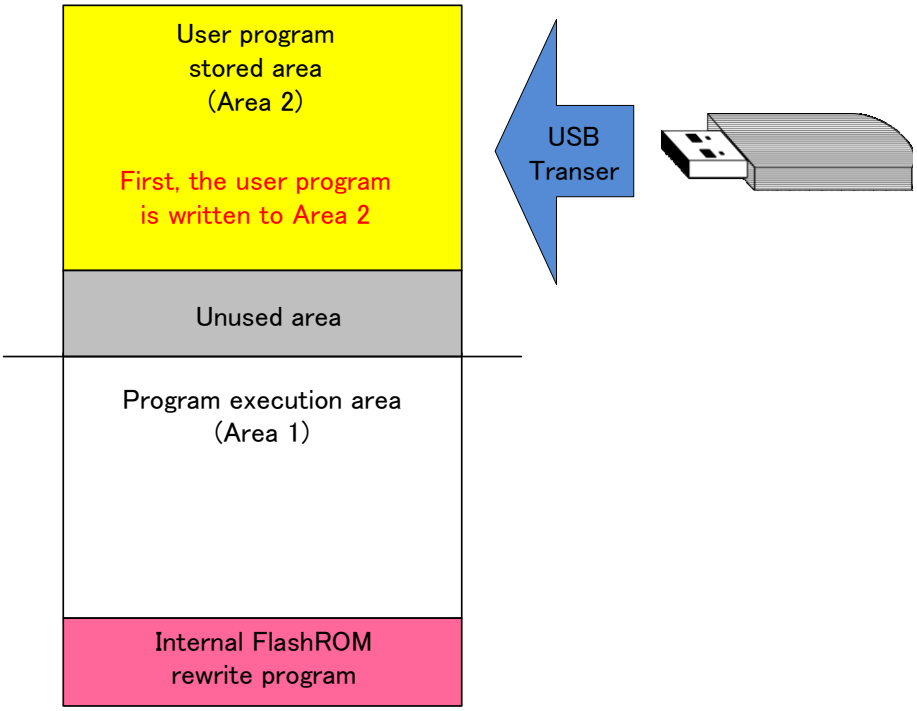


Figure 8-2 Writing of User Program to On-Chip Flash ROM (Area 2)

3. When the write finishes successfully, **Internal Flash ROM rewrite program via USB Mass Storage** is copied from Area 2 to Area 1. When copying to Area 1 finishes, the user program located in Area 1 is launched.

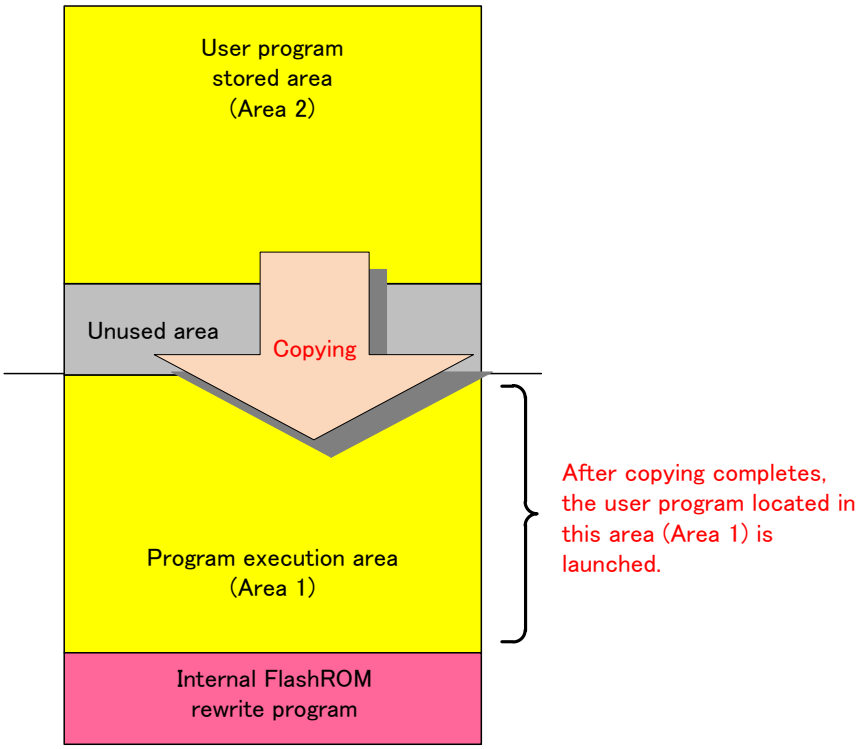


Figure 8-3 Copying of User Program

4. When the write finishes successfully, **Internal Flash ROM rewrite program via USB Mass Storage** is copied from Area 2 to Area 1. When copying to Area 1 finishes, the user program located in Area 1 is launched.

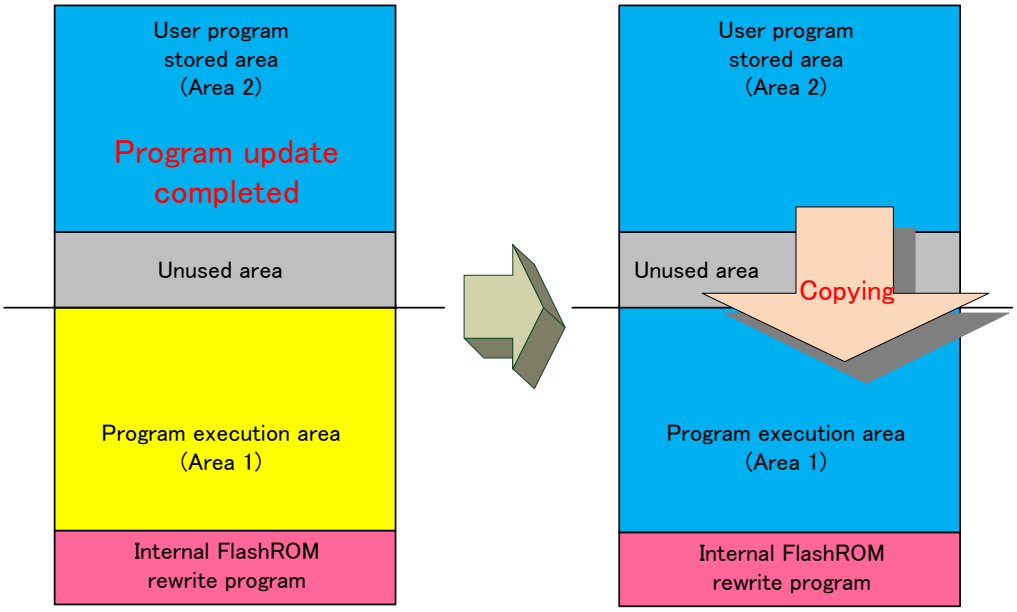


Figure 8-4 Updating the User Program

Note:

After the writing completes properly to Area 2, if this program can not be erased Area 1 by some reason, the user program previously written in Area 1 start up again since the user program is not updated to Area 1. If the phenomenon that Area 1 can not be erased occures, please do the writing processing again to Area 2. (Refer to the above step 2.)

5. If writing to Area 2 fails, for example due to USB transfer failure while the write to the flash ROM is in progress, the user program that was written to Area 1 in step 4, above, remains intact, so the user program previous to the failed write to the flash ROM can be launched.

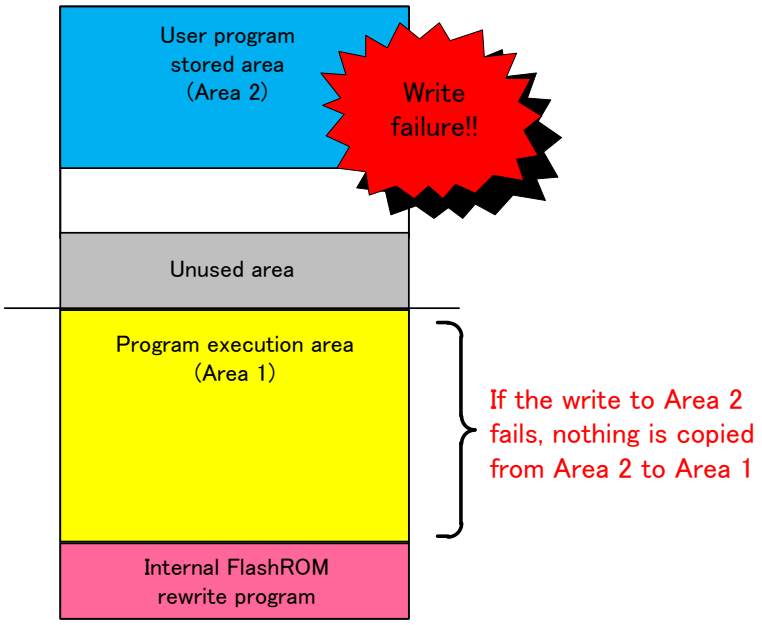


Figure 8-5 User Program Update Failure

Note:

1. While copying from Area 2 to Area 1, if the copying processing is failure by some reason, please reset or power on the RSK/RSSK. This program copies the user program again from Area 2 to Area 1. The user program is started up if the copy processing completes properly. This copy process requires a maximum of about 10 seconds after resetting or power on the RSK/RSSK.

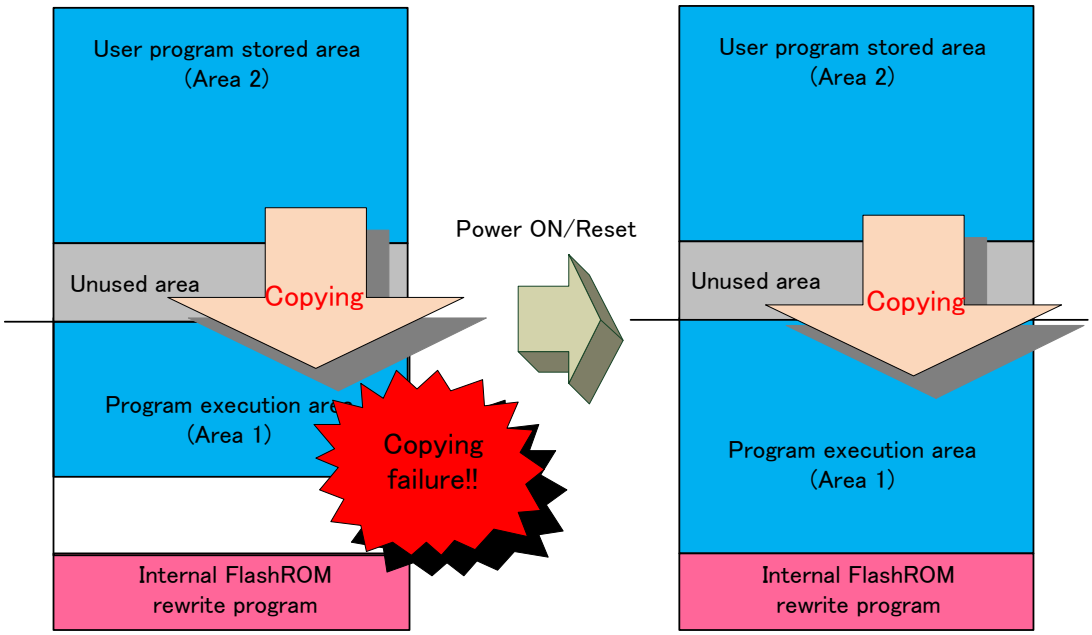


Figure 8-6 Failure to Copy from Area 2 to Area 1

2. The user program is run from Area 1, so Area 1 should be specified in the section settings for the user program code attribute and romdata attribute when you build your project.

   code attribute : Stored execution code.
   romdata attribute : Stored rom data

3. Whether or not the backup function is supported is specified by a macro definition in r_usb_fwupdater_config.h. For details of this setting, refer to **6.2, Internal Flash ROM rewrite program via USB Mass Storage Settings**.

4. Enable dual mode if the MCU you are using supports it.

8.2 Operation Immediately After Reset

Boot processing indicates the processing executed after the MCU is reset and before the main function (C language description: main()) is executed.

In RX MCUs, boot processing chiefly performs the following as initialization after reset:

1. Allocate stack area and set stack pointer
2. Allocate argument area for main function
3. Initialize data area and stack area
4. Branch to user program and initialize MCU peripheral devices in hdwinit function
5. Branch to main function

After reset, processing jumps from Internal Flash ROM rewrite program via USB Mass Storage to the user program. Therefore, make sure Internal Flash ROM rewrite program via USB Mass Storage is complete and the above-described MCU initializations are executed.

8.2.1 Power On / Reset Operation Flow

This section explains the operation flow after power on or reset for Internal Flash ROM rewrite program via USB Mass Storage.

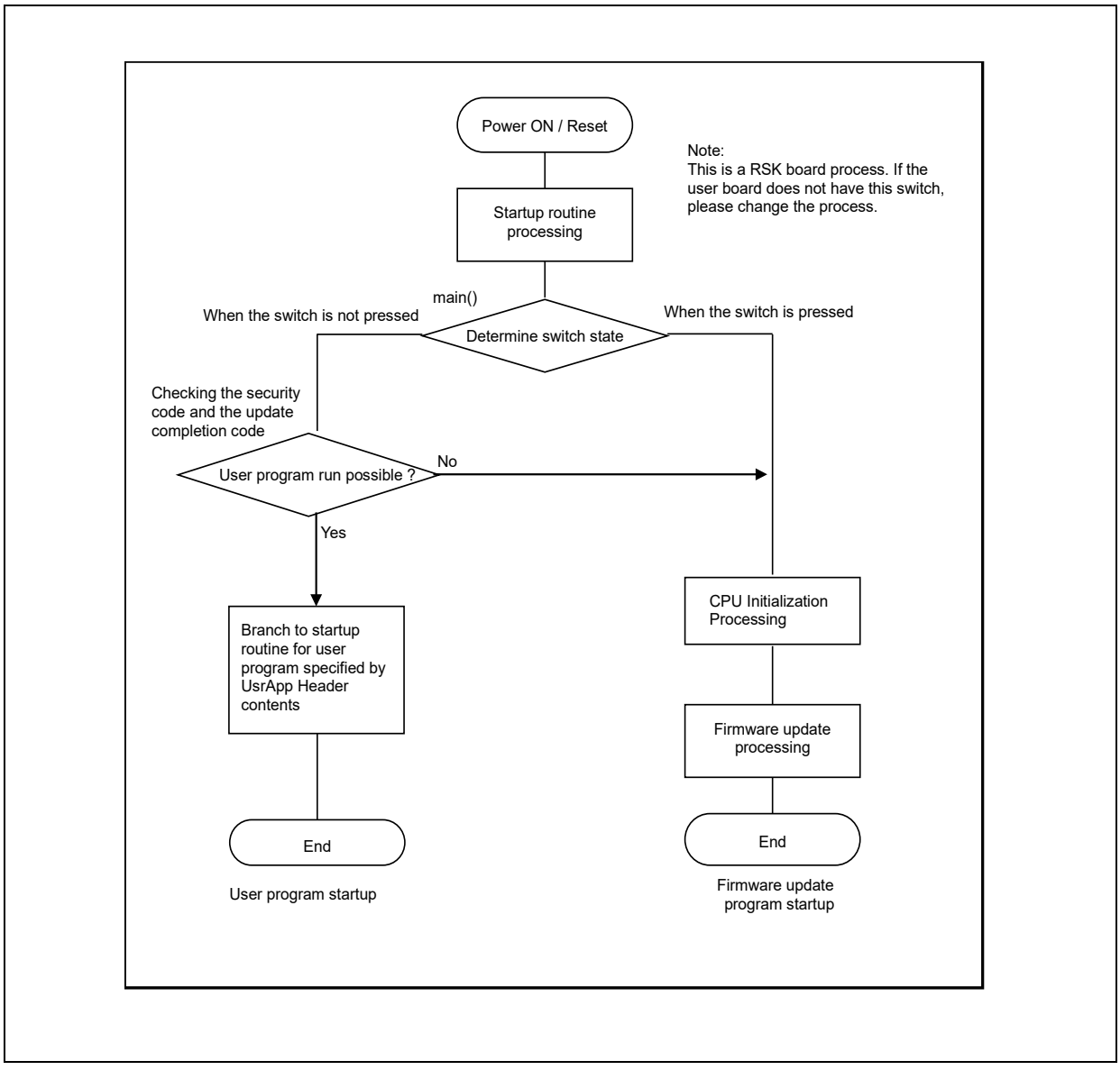


Figure 8-7 Power On / Reset Operation Flow

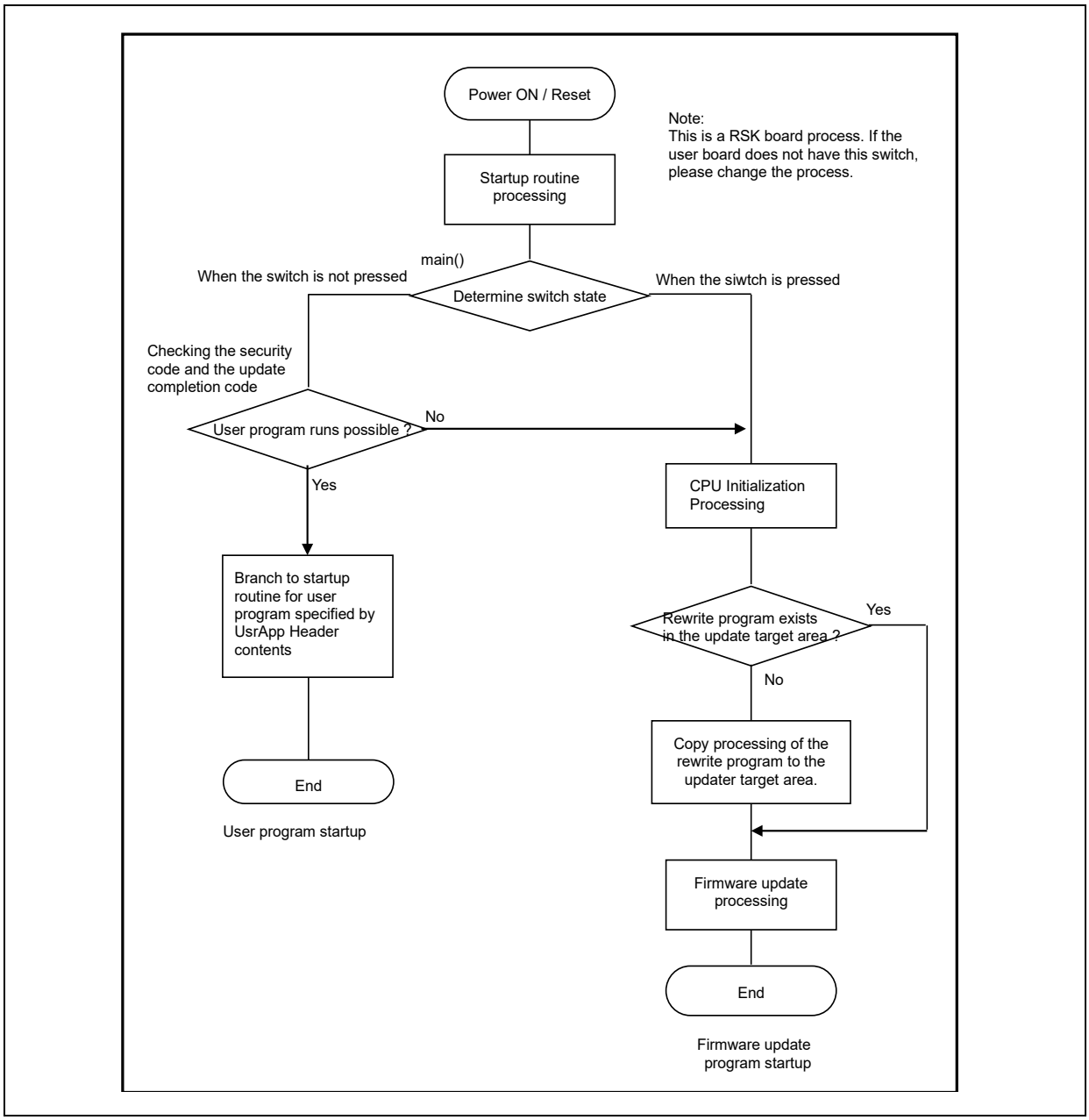


Figure 8-8 Power On / Reset Operation Flow (Using Dual Mode)

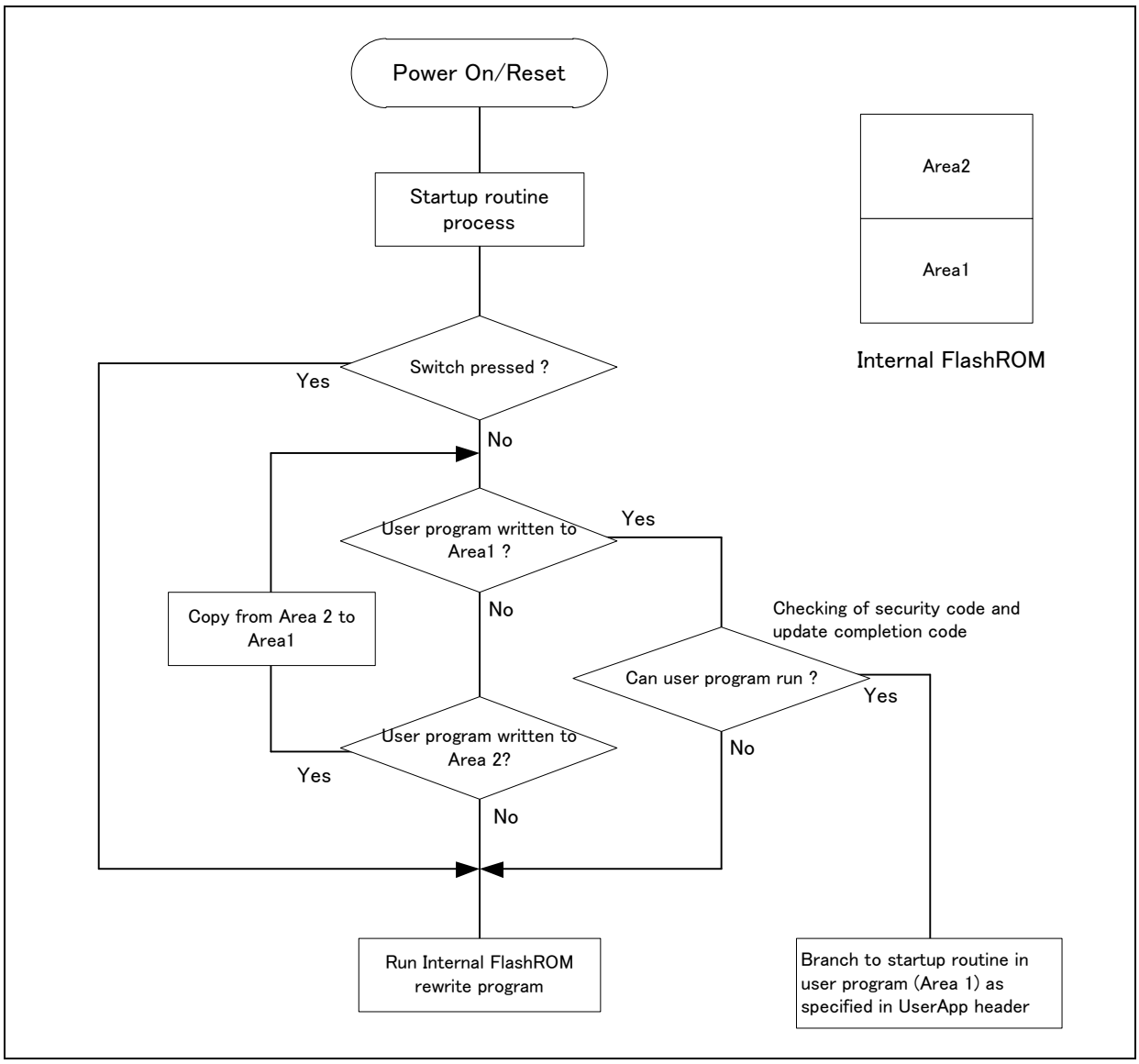


Figure 8-9 Power On / Reset Operation Flow (Using the backup function)

For information regarding branch address to security code and user program, refer to section **5.1, UserApp Header Area (user application header).**

Note that even if the security code in the UsrApp Header area is set correctly, if the start address of the user program is incorrect, the user program will not operate properly.

8.2.2 User program startup conditions

The user program set in the UsrApp Header area is started up when all of the following conditions are met:

1. Correct security code is set
2. Correct user program start address is set
3. The updating completion code is set.

   Internal Flash ROM rewrite program via USB Mass Storage writes the update completion code automatically after the user program is written properly to the Flash ROM

If the security code and the updating completion code do not match (is incorrect), only **Internal Flash ROM rewrite program via USB Mass Storage** will start up; the user program will not run.

8.2.3 Internal Flash ROM rewrite program via USB Mass Storage startup conditions

1. When user program has been written to ROM:

   Internal Flash ROM rewrite program via USB Mass Storage starts up when RESET is executed while the switch (RSK: Switch3, RSSK: Switch2) on the RSK/RSSK is pressed.

2. When user program has not been written to ROM:

   Internal Flash ROM rewrite program via USB Mass Storage starts up when power is turned on.

8.2.4 Branch to Internal Flash ROM rewrite program via USB Mass Storage

The main() function in Internal Flash ROM rewrite program via USB Mass Storage performs branch judgment to determine whether to jump to the user program or to continue with Internal Flash ROM rewrite program via USB Mass Storage.

After conditional branching is performed, the CPU build-in functions and peripheral circuits are initialized and Internal Flash ROM rewrite program via USB Mass Storage is executed.

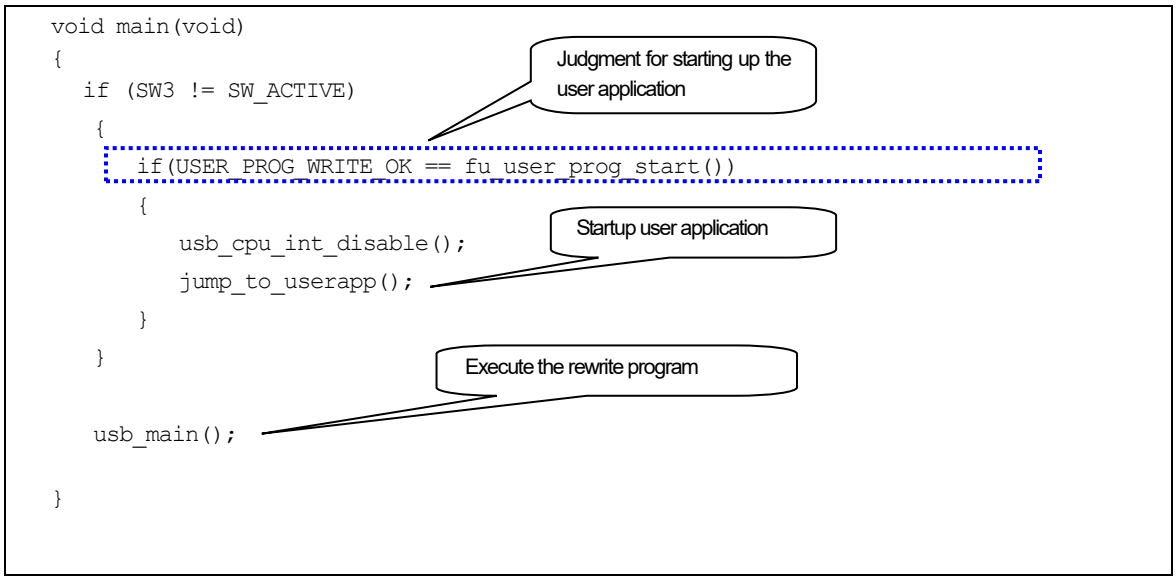


Figure 8-10 main() Function

8.2.5 Jump to user application

The processing to jump to the user program is performed in the jump_to_userapp( ) function. Refer to section **5.1, UserApp Header Area (user application header)** for details on specifying the start address of the user program jump destination.

8.3 File/Folder Configuration

The structure of the source files and folders of Internal Flash ROM rewrite program via USB Mass Storage is shown below.

```
(MCU name)
    ＋HardwareDebug          Build Result
    ＋src
      ＋———r_bsp             [Renesas Board Support Packcage]
      ＋———r_config          [Configuration File]
      ＋———r_flash_rx        [Flash Module]
      ＋———r_tfat_driver_rx  [FAT Driver]
      ＋———r_tfat            [FAT]
      ＋———r_usb_basic_mini  [USB Basic Driver]
      ＋———r_usb_hmsc_mini   [USB Host Mass Storage Class Driver]
      ＋———demo_src          [Application Program]
```

Figure 8-11 Internal Flash ROM rewrite program via USB Mass Storage Folder Configuration

8.3.1 *r_config* Folder

This folder contains header files for system settings.

8.3.2 *r_flash_rx* Folder

This folder contains source files and header files related to the flash module. For details, see the application note *Flash Module Using Firmware Integration Technology*.

When the microcontroller is selected using the board support package (r_bsp), the flash programming type is selected automatically.

8.3.3 *r_bsp* Folder

This folder contains source files and header files related to the BSP. For details, see the application note *RX Family Board Support Package Module* (document number: R01AN1685).

8.3.4 *r_tfat_rx* Folder

This folder contains source files and header files related to the FAT. For details, see the application note *RX Family Open Source FAT File System [M3S-TFAT-Tiny] Module* (document number: R20AN0038).

8.3.5 *r_tfat_driver_rx* Folder

This folder contains source files and header files related to the FAT driver. For details, see the application note *RX Family M3S-TFAT-Tiny Memory Driver Interface Module* (document number: R20AN0335).

8.3.6 *r_usb_basic_mini, r_usb_hmsc_mini* Folder

This folder contains source files and header files related to the USB driver. For details, see the application notes *Renesas USB MCU USB Basic Mini Host and Peripheral Driver* (document number: R01AN2166) and *Renesas USB MCU USB Host Mass Storage Class Driver* (document number: R01AN2169).

8.3.7 *demo_src* Folder

This folder contains the application program of Internal Flash ROM rewrite program via USB Mass Storage.

Table 8-1 Source file of Internal Flash ROM rewrite program via USB Mass Storage

File Name	Description
main.c	C language source code file of main function
r_flash_apl.c	Source code file of processing for programming ROM
r_flash_api.h	Header file of processing for programming ROM
r_flash_read_buff.c	Source code file of receive-related processing
r_flash_read_buff.h	Header file of receive-related processing
r_fwupdater_apl.c	Source code file of flash programming data processing
r_fwupdater_apl.h	Header file of flash programming data processing
Other	Application program file for USB driver Peripheral control files for RSK/RSSK

8.3.8 *HardwareDebug* Folder

This folder contains the executable object files generated at build time and the mot file of Internal Flash ROM rewrite program via USB Mass Storage.

8.4 Cautions

Internal Flash ROM rewrite program via USB Mass Storage determines whether to jump to Internal Flash ROM rewrite program via USB Mass Storage or to the user program based on the switch (RSK: Switch 3, RSSK: Switch 2) on the RSK/RSSK. The processing to make this determination is dependent on the specifications of the board, so modify the determination processing if necessary to match the board used. The determination processing is part of the main function of Internal Flash ROM rewrite program via USB Mass Storage.

9. Using the e² studio project with CS+

This package contains a project only for e² studio. When you use this project with CS+, import the project to CS+ by following procedures.

Note:

1. The name of the folder which stores *src* folder and *rcpc* file has to be "MCU name_FirmwareUpdater". For example, the folder name is "RX63N_FirmwareUpdater" when using RX63N.
2. Uncheck the checkbox *Backup the project composition files after conversion* in *Project Convert Settings* window.

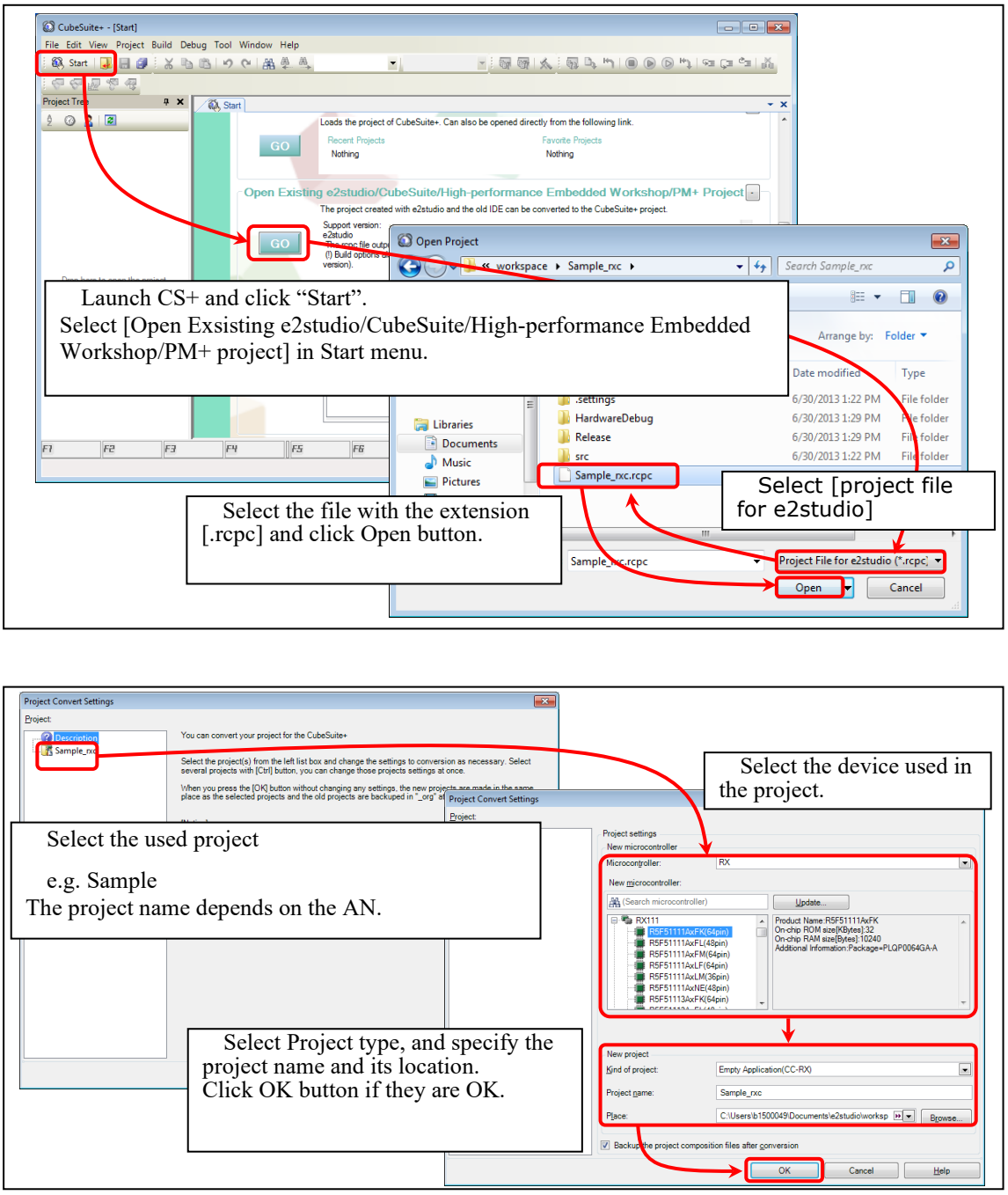


Figure 9-1 Using the e² studio project with CS+

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Mar 31, 2017	-	First edition issued.
1.01	Jan 31, 2018	-	1. Support for RX65N/RX651-2M 2. Support for dual mode.
1.02	Apr 16, 2018	-	Support the backup function
1.03	Apr 16, 2019	-	RX66T and RX72T are added in Target Device.
1.04	Mar 1, 2020	-	RX72M, RX72N, RX66N and RX23W are added in Target Device.
1.05	Mar 1, 2021	-	RX671 is added in Target Device.

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/.