APPLICATION NOTE

RX Family

Internal Flash ROM rewrite program via USB CDC

R01AN3294EJ0106
Rev.1.06
Mar 1, 2021

Overview

This application note explains Flahs ROM rewrite program, which uses USB peripheral controllers.

Target Devices

RX111, RX113, RX231, RX23W

RX62N/RX621, RX630, RX63N/RX631, RX63T

RX65N/RX651, RX64M, RX71M, 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

a. The operations described in this application note are not guaranteed. When using this application note for your system, conduct an extensive evaluation to ensure compatibility.

b. The program settings are based on Little Endian. If the user program is based on Big Endian, please modify this program to Big Endian as well. Please refer to **6.2 Internal Flash ROM rewrite program via USB CDC Settings** about the endian setting.

c. When implementing this program into your system, please refer to the contents of section **6 Internal Flash ROM rewrite program via USB CDC and User Program Settings** and **7.4 Cautions** section**.**

d. **Internal Flash ROM rewrite program via USB CDC** does not analyze the user program (mot/hex file). When you develop the file transfer application program (GUI tool) woking on PC, the GUI tool needs to analyze the user program. In addition, refer to the section **9, Data Transmission Specification** for the USB data transfer specification with RX device.

e. This program does not support USB Command Verifier (CV).

f. The operation is not checked when changing the header files except *r_usb_fwupdater_config.h* file in *r_config* folder.

g. 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.

h. Allocate the "FW_CODE" section at address 0xFFFFFF7C when using the dual mode.

i. It is necessary to move the following resistance on the RSSK board when using RSSK(RX23W).

R89 --> R90

R96	-->	R97
R112	-->	R113

j. 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
	RX630	USBa module
	RX63T	USBa module
	RX64M	USBb module
	RX71M	USBb module
	RX65N/RX651	USBb module
	RX66T/RX72T	USBb module
	RX72M	USBb module
	RX72N	USBb module
	RX66N	USBb module
	RX111	USBc module
	RX113	USBc module
	RX231	USBd module
	RX23W	USBc module
USB1 module (start address:0xA0200 / 0xD0400)	RX62N/RX621	USB module
	RX63N/RX631	USBa module
	RX64M	USBA module
	RX71M	USBAa 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
CDC	:	Communication Device Class
e^2 studio	:	Eclipse embedded studio (RX-supported)
H/W	:	Renesas USB device
MCU	:	Micro control Unit
P/E	:	Program / Erase
RSK	:	Renesas Starter Kit
RSSK	:	Renesas Solution Starter Kit
USB	:	Universal Serial Bus

2. Internal Flash ROM rewrite program via USB CDC Overview

The program transfers a specified user program from the file transfer application on the host machine (referred to as "PC" herein) to the evaluation board via a USB connection. The transferred user program is written to an address in the ROM using the Flash self-programming library

The program is configured as follows:

(1). Internal Flash ROM rewrite program via USB CDC

This is the program to be implemented in the evaluation board; performs serial transmission via USB and self programming.

(2). File transfer application

The application runs on the host machine (PC) and transfers specified files to the evaluation board in USB transmissions.

(3). User program

This file is written by Internal Flash ROM rewrite program via USB CDC for the operation confirming.

Program 1: LEDs on RSK/RSSK board light up in consecutive order.

Program 2: LEDs on RSK/RSSK board light up simultaneously.

The following shows the program's data flow.

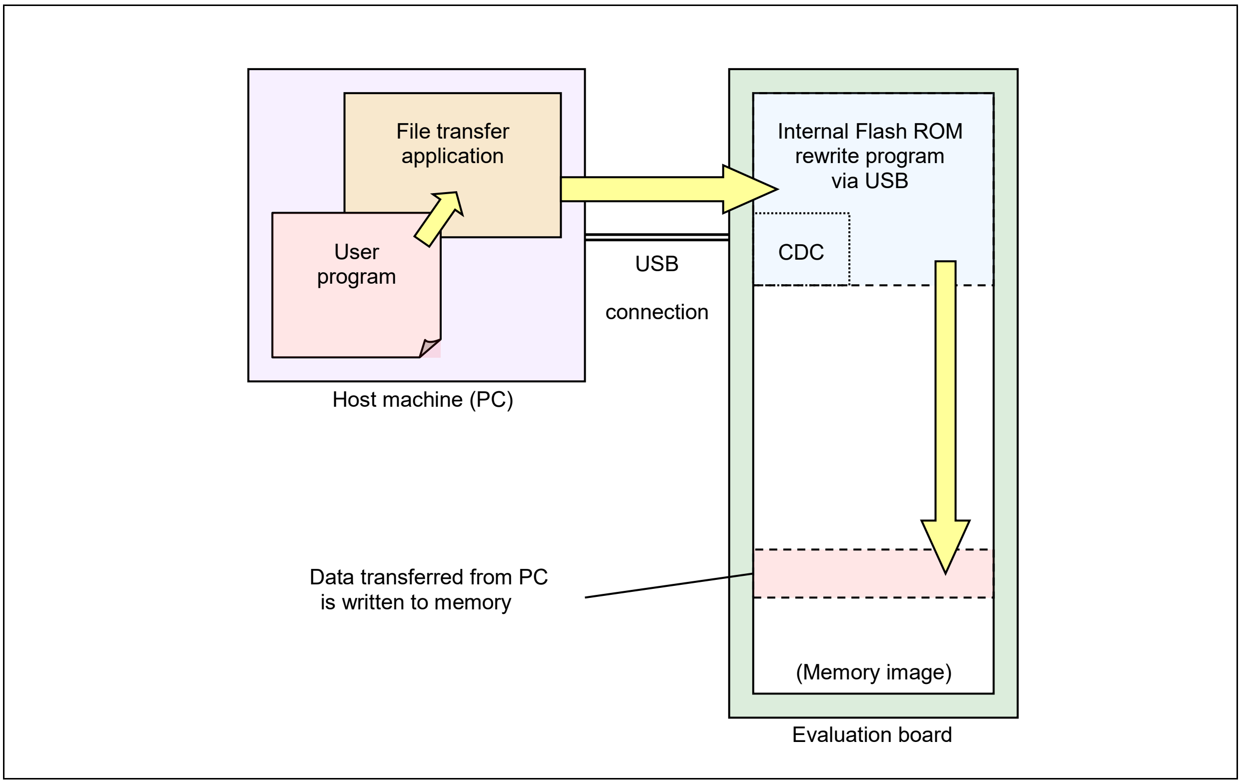

File transfer application
User program
Host machine (PC)
USB
connection
Internal Flash ROM rewrite program via USB
CDC
Data transferred from PC is written to memory
(Memory image)
Evaluation board

Figure 2-1 RX USB F/W Update Data Flow

The Internal Flash ROM rewrite program via USB CDC works when the evaluation board is started up in specified conditions, otherwise the user program works.

2.1 Features

This program offers the following features.

1. The program performs full-speed data transfers between the USB host and the evaluation board using CDC (Communication Deive Class).
2. The program occupies part of the internal flash memory. If your MCU supports user boot area, the Flash ROM rewite program can be assigned to the user boot area.
3. This program supports the Motorola S and Intel HEX formats as the user program format (.mot/.hex files).
4. The program supports writing and verification with respect to the Flash ROM.
5. The program supports dual mode. (For information regarding dual mode, see the hardware manual of an MCU that supports dual mode.)
6. A backup function is supported. For details of the backup function, refer to section **7.2, Backup Function**.
7. The user program can use all interrupt functions.

2.2 ROM Size

The following is ROM size used by this program.

ROM Size : 8K bytes

[Note]

The compiler uses CC-RX V.3.01/V.3.03 and the default option is specified for the optimization option.

2.3 Target Device & Flash Type

Four types of RX Flash are available. The following table shows which type of Flash is available according to MCU. For more details, please refer to the RX Family Flash Module Using Firmware Integration Technology Application Note.

Table 2-1 MCU Flash Programming Type

Flash Programming Type	Target Device
Flash Type1	RX111, RX113, RX231, RX23W
Flash Type2	RX62N/RX621, RX630, RX63N/RX631, RX63T
Flash Type3	RX64M, RX71M, RX66T/RX72T
Flash Type4	RX65N/RX651, RX72M, RX72N, RX66N, RX671

2.4 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, RX651, RX62N, RX63T, RX630, RX111, RX113, RX231, RX72T, RX72M, RX72N, RX66N, RX23W
(3).	Emulator	E2 Lite
(4).	USB cable	USB communication between evaluation board and PC
(5).	PC	PC running on Window® 8.1/ Window® 10 (32bit/64bit)

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).	Flash programming tool	Renesas Flash Programmer V.3.03.00
(4).	USB F/W Update sample/program set	
		Internal Flash ROM rewrite program via USB CDC File transfer application Sample user program

Note:

(a). Operations for this program has not been confirmed when using USB1 module in RX62N.

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

2.5 Folder Configuration

The following is the folder configuration for this program.

```
 (Top Directory)
  +—reference
  |         +--cdc_inf
  |         |      CDC driver sample inf file (CDC_Demo.inf)
  |         +--FirmupdateGUI
  |         |      |  File transfer application (UsbfUpdater.exe / UsbfUpdater.ini)
  |         |      +---source
  |         |               File transfer application sources
  |         +--SampleProgram (Sample program for operation confirmation )
  |               +-- (MCU name)
  |                           +-- src (Sample program sources)
  |                           +-- mot (Sample user program)
  +—workspace (Internal Flash ROM rewrite program via USB CDC Sample projects)
           +-- (MCU name_FirmwareUpdater)
```

The following provides a description of each folder.

(1). reference¥cdc_inf

This folder stores the INF file to install the Windows ® CDC driver.

CDC_Demo.inf: Windows ® CDC driver (Windows® 32bit/64bit)

(2). reference¥FirmupdateGUI

This folder stores the file transfer application.

UsbfUpdater.exe: File transfer application execution file

UsbfUpdater.ini: File transfer application setting file

(3). reference¥FirmupdateGUI¥source

This folder stores the file transfer application source program. For more details, refer to section **8, File Transfer Application (RX USB Firmware Updater) Explanation**

(4). reference¥SampleProgram

This folder stores the sample user program.

sample1.mot: LEDs light up in consecutive order

sample2.mot: LEDs light up simultaneously

(5). workspace

This file stores Internal Flash ROM rewrite program via USB CDC for each MCU. For more details, refer to section **7 Internal Flash ROM rewrite program via USB CDC Explanation.**

3. Internal Flash ROM rewrite program via USB CDC Setup

This section explains the setup sequence for this program.

3.1 Project Setup

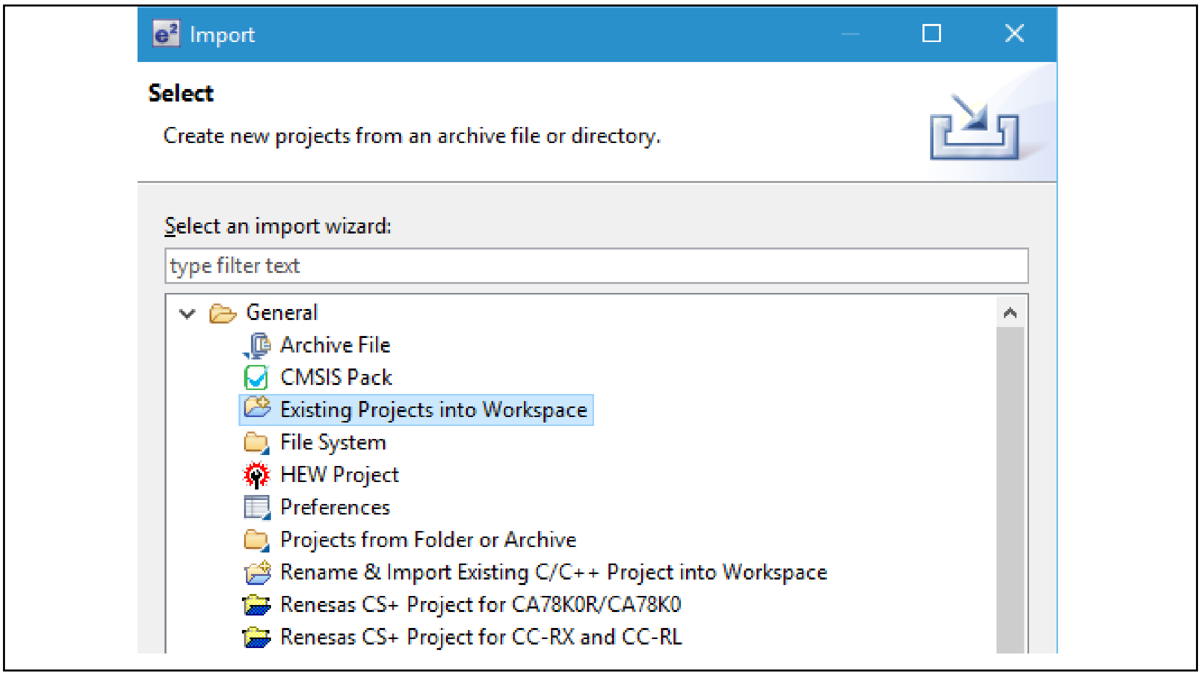
Select the folder with the name of the MCU you are using from the Workspace folder tab. Set up the project according to the following sequence. This sequence is for setting up with e^2 studio.

(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**].



Import

Select

Create new projects from an archive file or directory.

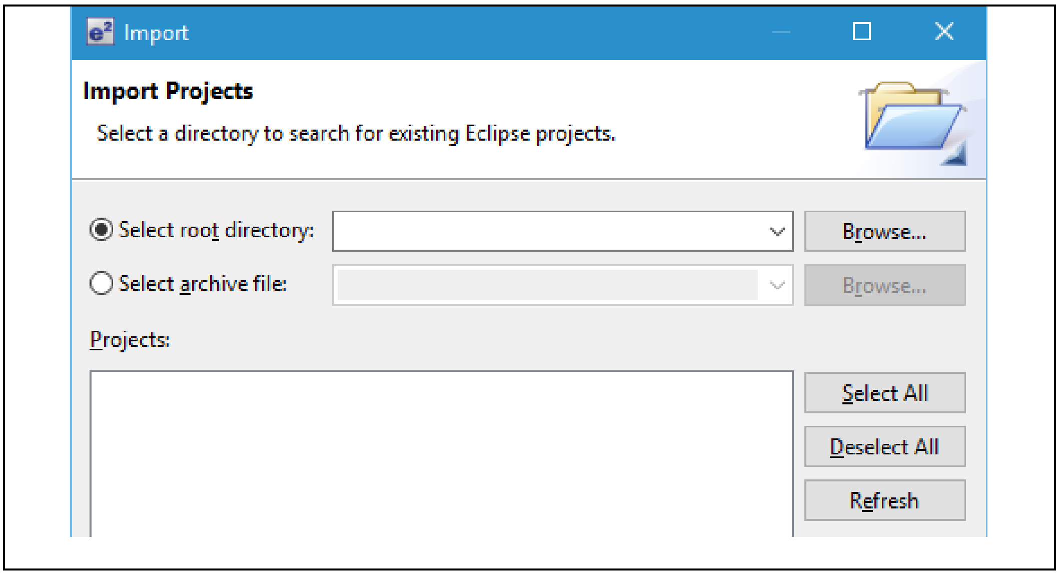
Select an import wizard:

type filter text

- General
  - Archive File
  - CMSIS Pack
  - Existing Projects into Workspace
  - File System
  - HEW Project
  - Preferences
  - Projects from Folder or Archive
  - Rename & Import Existing C/C++ Project into Workspace
  - Renesas CS+ Project for CA78K0R/CA78K0
  - Renesas CS+ Project for CC-RX and CC-RL

Figure 3-1 Select Import Source

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



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

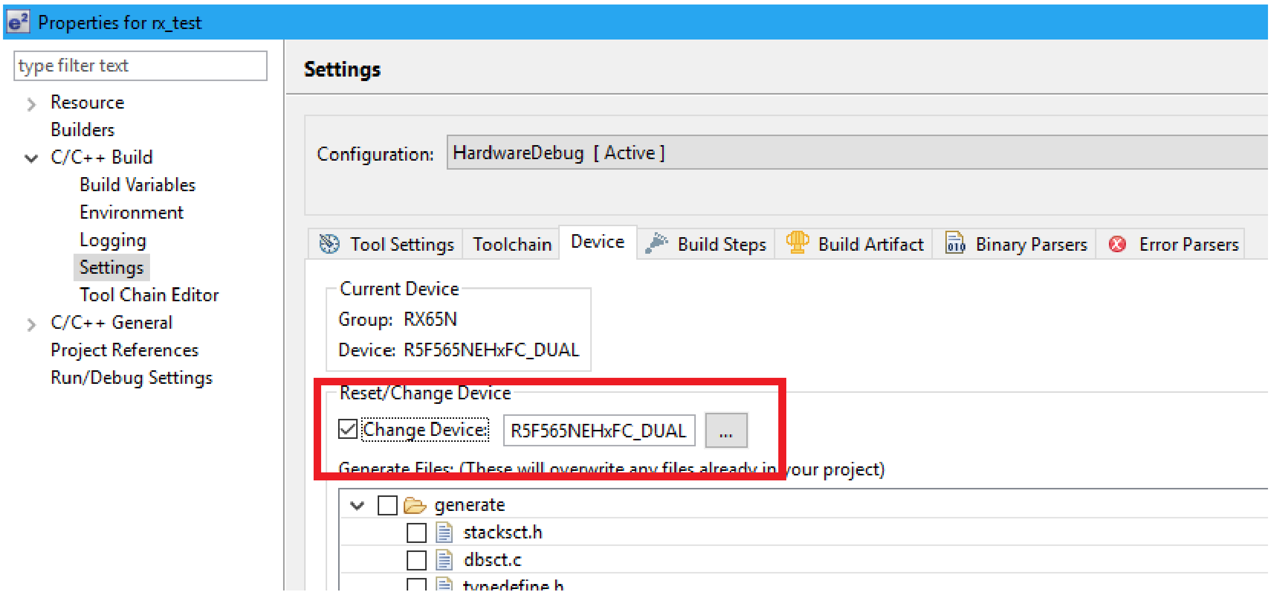
Figure 3-2 Project Import Dialog Box

(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)



Settings

Configuration: HardwareDebug [Active]

Current Device

Group: RX65N

Device: R5F565NEHxFC_DUAL

Reset/Change Device

☑ Change Device: R5F565NEHxFC_DUAL

Generate Files: (These will overwrite any files already in your project)

□ generate

□ stacksct.h

□ dbsct.c

□ typedefine.h

Figure 3-3 Change Device

4. Execute Internal Flash ROM rewrite program via USB CDC

This section describes how to execute this program.

This process uses the RSK/RSSK board to confirm operations of two different user programs.

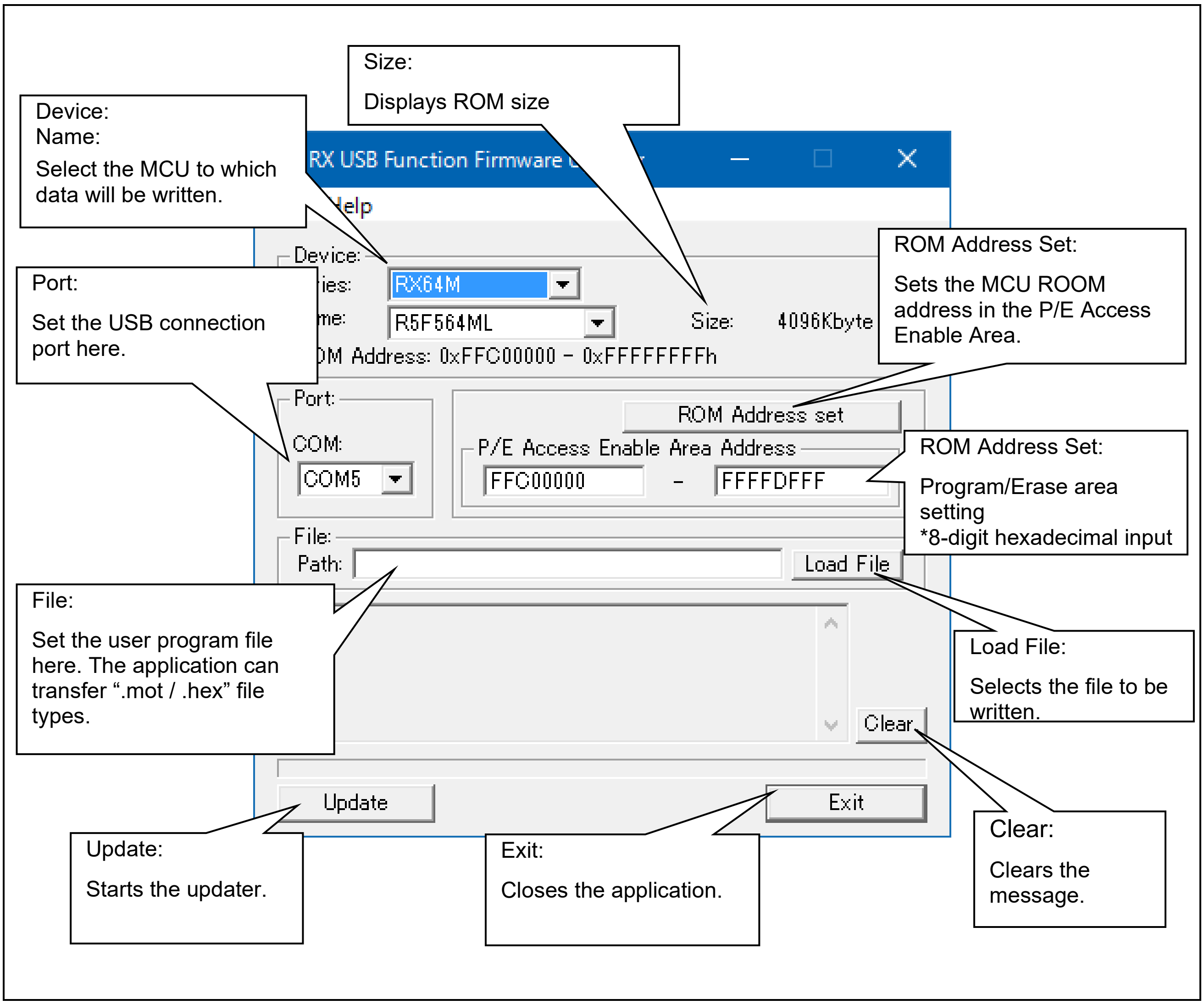
4.1 File Transfer Application (RX USB Function Firmware Updater) Startup

The File Transfer Application which transmits the user program starts up when the UsbfUpdater.exe file in the FirmupdateGUI folder is executed.

Figure 4-1 shows how to set the following file transfer application.

Notes:

If the file transfer application does not start up, make sure the folder that contains the exe file also contains the UsbfUpdater.ini, and then retry the process.



RX64M

R5F564ML

Size: 4096Kbyte

OM Address: 0xFFC00000 - 0xFFFFFFFFh

COM5

FFC00000 - FFFFDFFF

Size:

Displays ROM size

Device:
Name:
Select the MCU to which data will be written.

ROM Address Set:

Sets the MCU ROOM address in the P/E Access Enable Area.

Port:

Set the USB connection port here.

ROM Address Set:

Program/Erase area setting
*8-digit hexadecimal input

File:

Set the user program file here. The application can transfer “.mot / .hex” file types.

Load File:

Selects the file to be written.

Update:

Starts the updater.

Exit:

Closes the application.

Clear:

Clears the message.

Figure 4-1 RX USB Firmware Updater GUI Software

4.1.1 P/E Access Enable Area Address

Set the Program/Erase enable area so that this program area will not be written over when the user program is written to the MCU.

Note that this program does not allow access to the ROM block that includes the reset vector (Block 0 in the RX Series). Please use the settings listed in Table 4-1 to set the range for P/E Access Enable Area Address.

Table 4-1 P/E Access Enable Area Address Settings

Backup Function	P/E address Setting		
OFF	On-chip ROM Area (Program ROM) Start Address	-	0xFFFFDFFF
ON	Start Address of Program Execution Area	-	0xFFFFDFFF

Notes:

1. The block including the specified address will be erased during an erase operation. Be careful when setting the ROM block size. For more details on ROM block size, refer to the user's hardware manual corresponding to the target MCU.
2. When selecting dual mode, specify the startup bank area (and not the update target area).
3. Specify the start address (start address of the start Flash ROM block) and the end address (end address of the end Flash ROM block) for the user prograum in *P/E Access Enable Area Address.*
4. For Backup function and the program execution area, refer to section **7.2, Backup Function.**

4.2 Writing Internal Flash ROM rewrite program via USB CDC to Flash ROM write and execution

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

4.2.1 Writing Internal Flash ROM rewrite program via USB CDC to ROM

(1). Hardware setup

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

a. Using an emulator

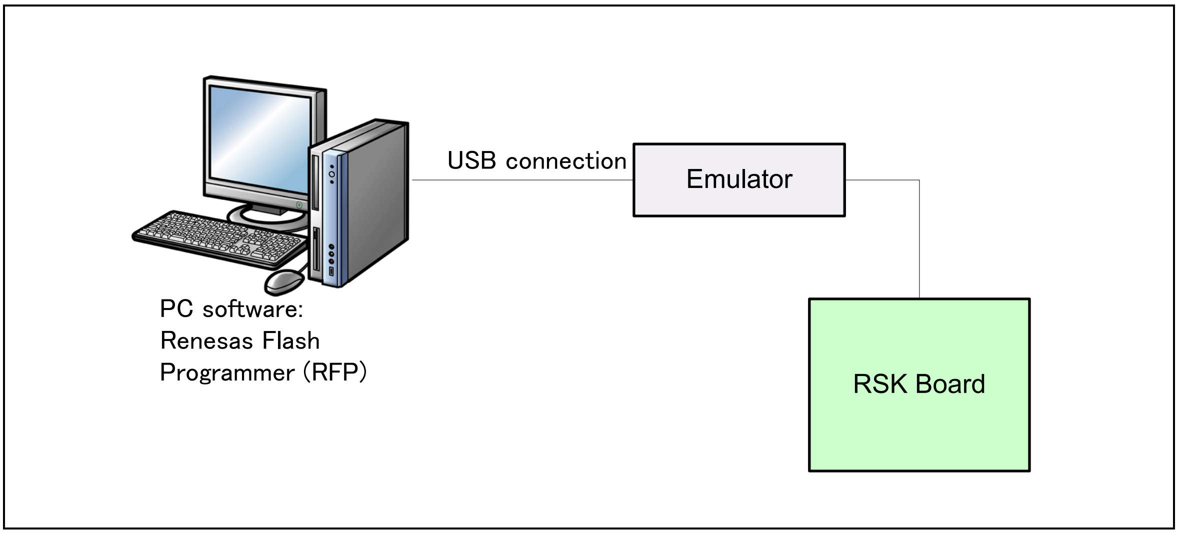


Figure 4-2 Connection Diagram Using an Emulator

b. Not using an emulator

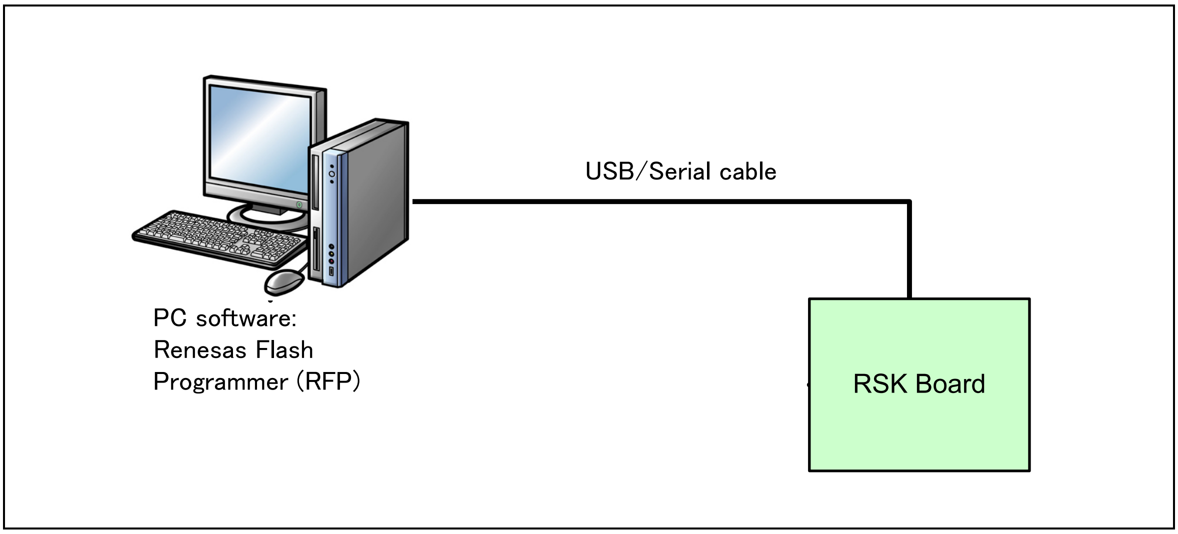


Figure 4-3 Connection Diagram with No Emulator

Notes:

a) Note that when writing this program to the user boot area in USB boot mode, the existing USB boot mode program in the user boot area will be overwritten.

b) When writing this program to the user boot area without use of an emulator (as in **Figure 4-3**), write to the ROM in boot mode. The user boot area cannot be programmed in USB boot mode.

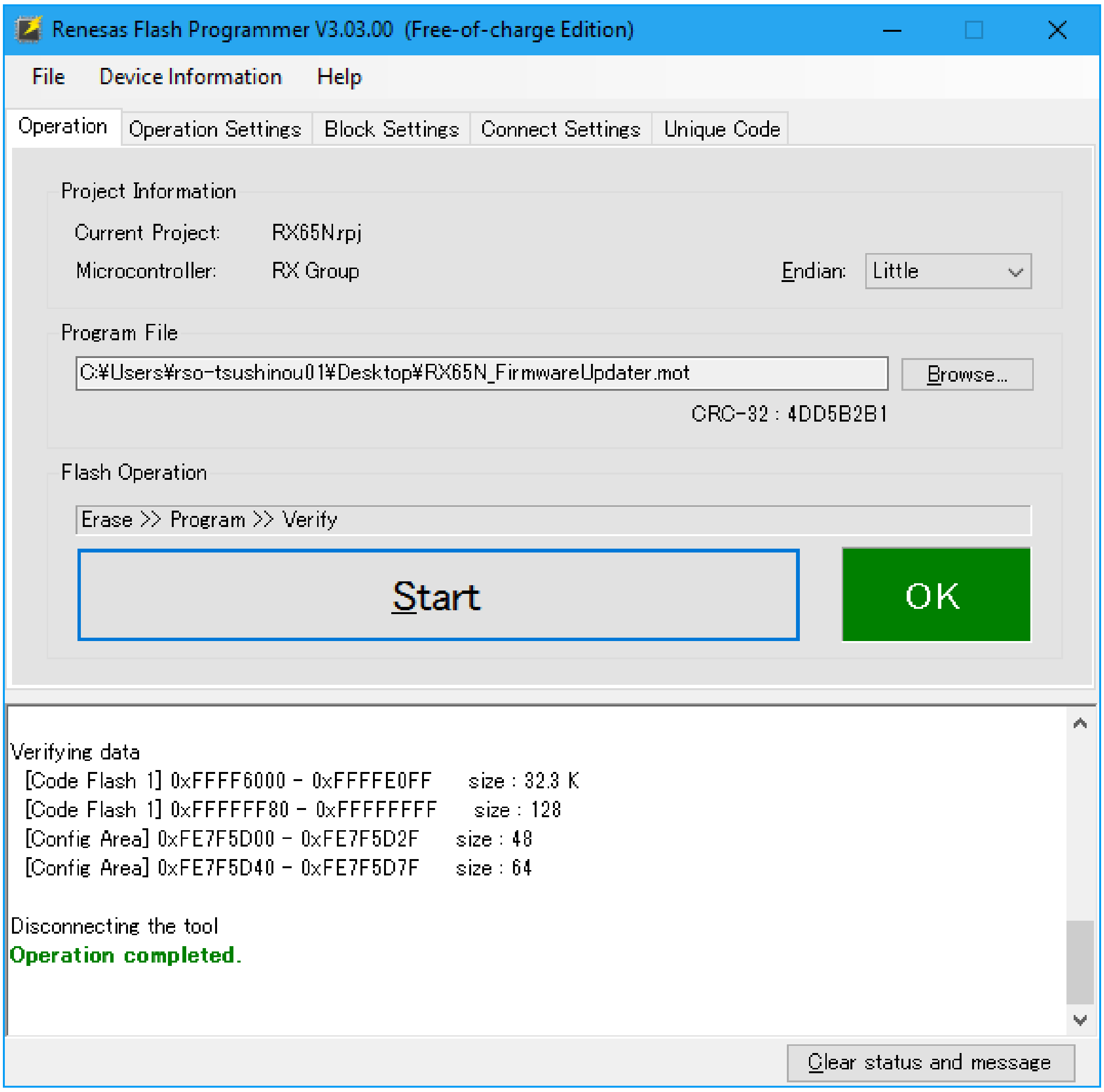
c) This program can be written to the user boot area when using an emulator (as in **Figure 4-2**).

d) When writing this program in USB boot mode, write the program to an area other than the user boot area.

e) Refer to the target MCU's user's hardware manual for more details on boot mode and USB boot mode.

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

Run the Renesas Flash Programmer (RFP) and, using the [**Browse]** for [Program File] button, select Internal Flash ROM rewrite program via USB CDC 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.



Project Information

Current Project: RX65N.rpj

Microcontroller: RX Group

Endian: Little

Program File

C:¥Users¥rso-tsushinou01¥Desktop¥RX65N_FirmwareUpdater.mot

CRC-32 : 4DD5B2B1

Flash Operation

Erase >> Program >> Verify

```
Verifying data
  [Code Flash 1] 0xFFFF6000 - 0xFFFFE0FF    size : 32.3 K
  [Code Flash 1] 0xFFFFFF80 - 0xFFFFFFFF    size : 128
  [Config Area] 0xFE7F5D00 - 0xFE7F5D2F    size : 48
  [Config Area] 0xFE7F5D40 - 0xFE7F5D7F    size : 64

Disconnecting the tool
Operation completed.
```

Figure 4-4 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 **4.2.2 Internal Flash ROM rewrite program via USB CDC address assignment** for more details concering positioning of Internal Flash ROM rewrite program via USB CDC.

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

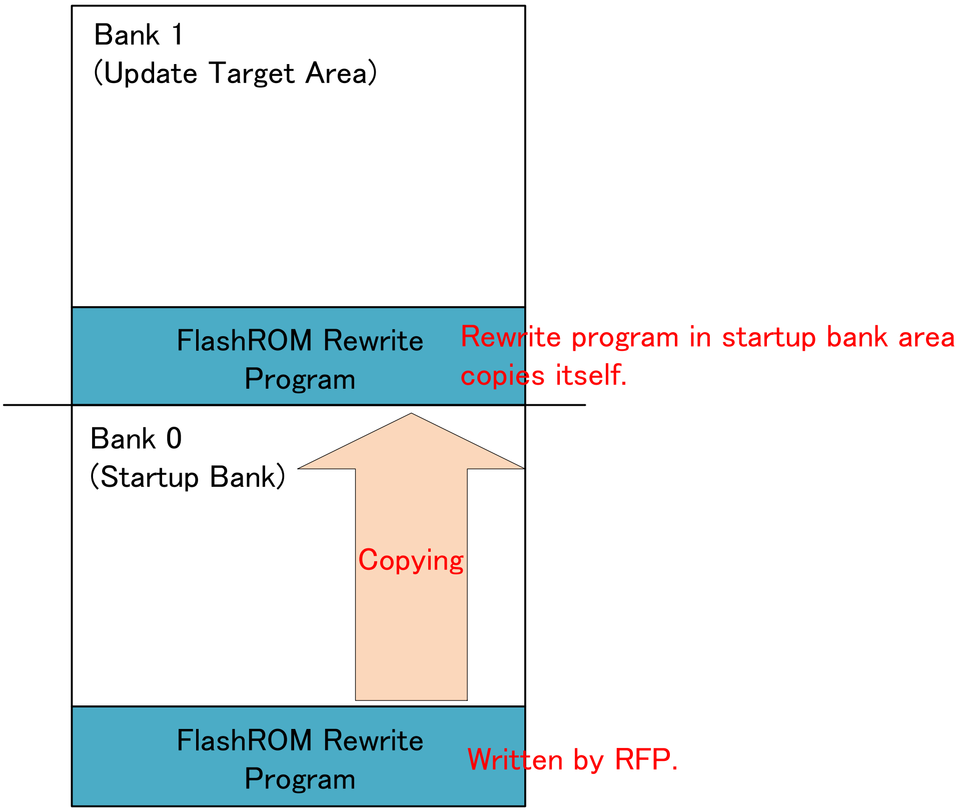


Figure 4-5 Placement of the Flash ROM Rewrite Program

Note:

The following message is displayed on the file transfer application (GUI tool) when the user program is written after the copying processing of FlashROM rewrite program is failure to the update target area.

ERR: Copying of Flash ROM rewrite program failed.

4.2.2 Internal Flash ROM rewrite program via USB CDC address assignment

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 CDC in the following area.

Alloation Areas for Internal Flash ROM rewrite program via USB CDC		
0xFFFFE000	-	0xFFFFFFFF

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

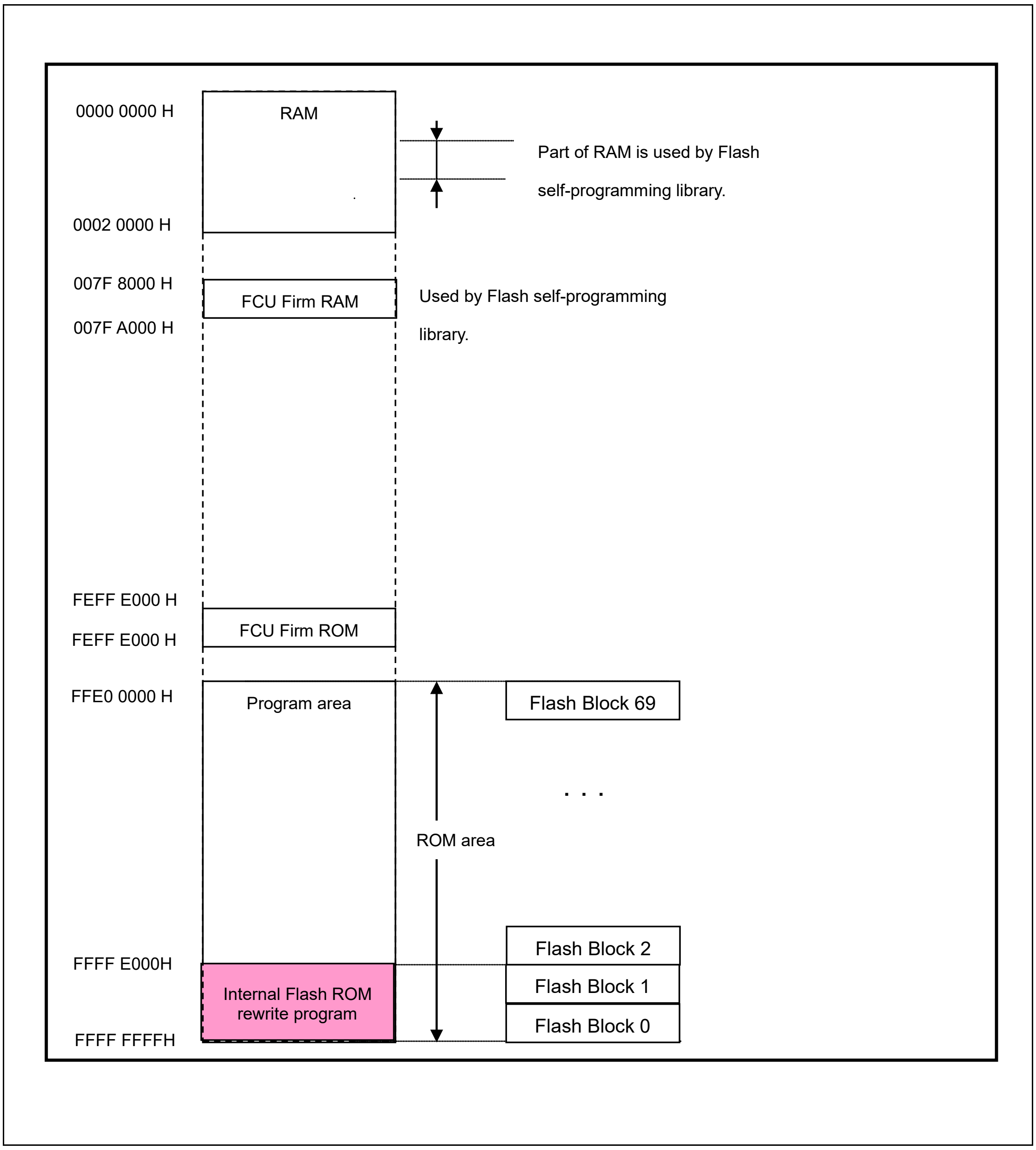


Figure 4-6 Memory Map (user boot area not used)

Notes:

When compiling **Internal Flash ROM rewrite program via USB CDC**, select **24 bits** 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). Assigning program to user boot area

Internal Flash ROM rewrite program via USB CDC can be assigned to the user boot area if it is supported by the target MCU. Table 4-2 provides user boot area information.

Table 4-2 MCU User Boot Area Information

MCU	User Boot Area	User Boot Address		
RX71M	32KB	0xFF7F8000	-	0xFF7FFFFF
RX64M	32KB	0xFF7F8000	-	0xFF7FFFFF
RX66T/RX72T	32KB	0xFF7F8000	-	0xFF7FFFFF
RX63T	16KB	0xFF7FC000	-	0xFF7FFFFF
RX63N/RX631	16KB	0xFF7FC000	-	0xFF7FFFFF
RX630	16KB	0xFF7FC000	-	0xFF7FFFFF
RX62N/RX621	16KB	0xFF7FC000	-	0xFF7FFFFF

Note:

When compiling **Internal Flash ROM rewrite program via USB CDC**, 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].

The following shows the memory map when **Internal Flash ROM rewrite program via USB CDC** is assigned to the user boot area in RX63N.

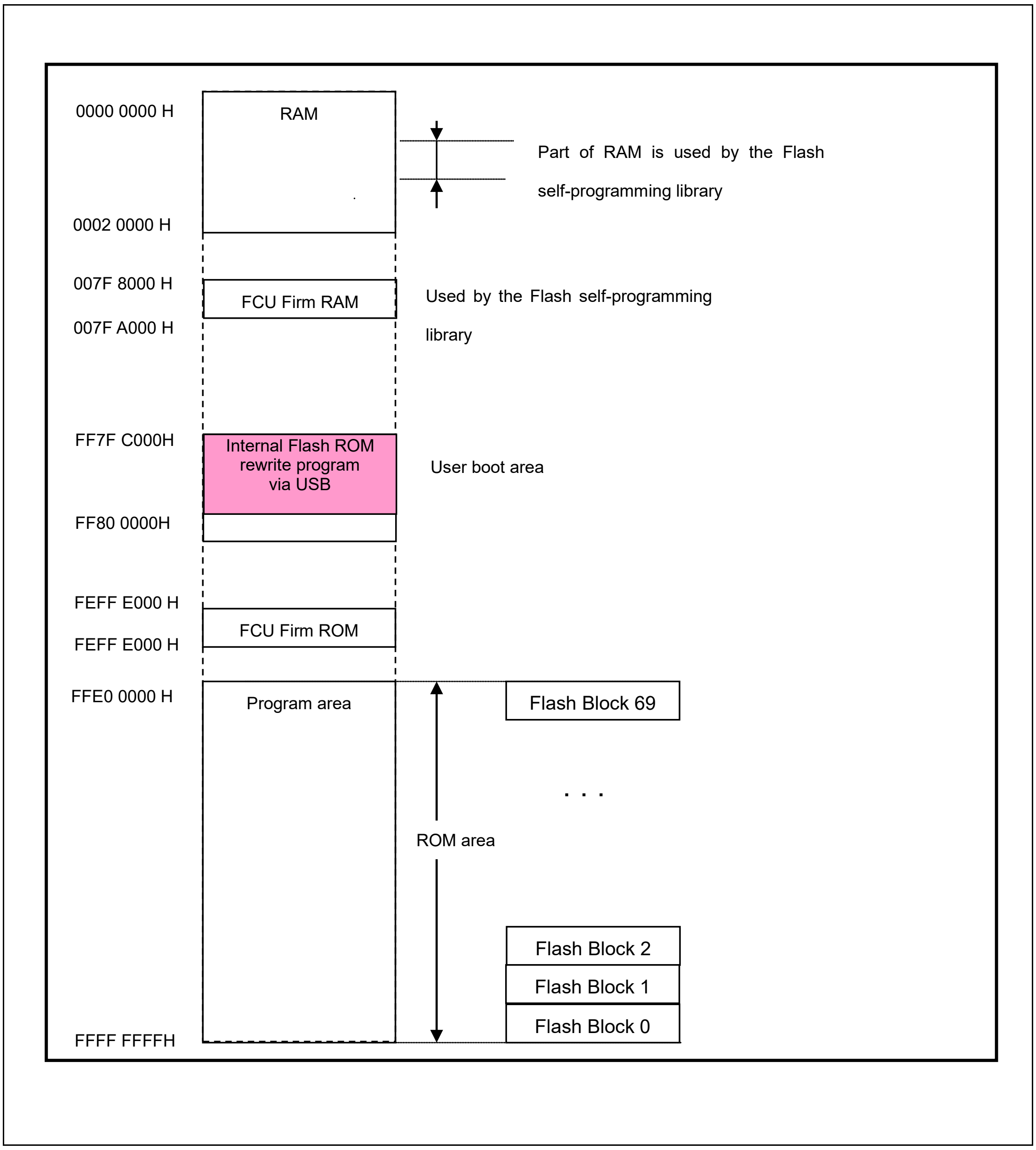


Figure 4-7 Memory Map (when using user boot area)

(3). When using dual mode

The momory map when using dual mode is shown below.

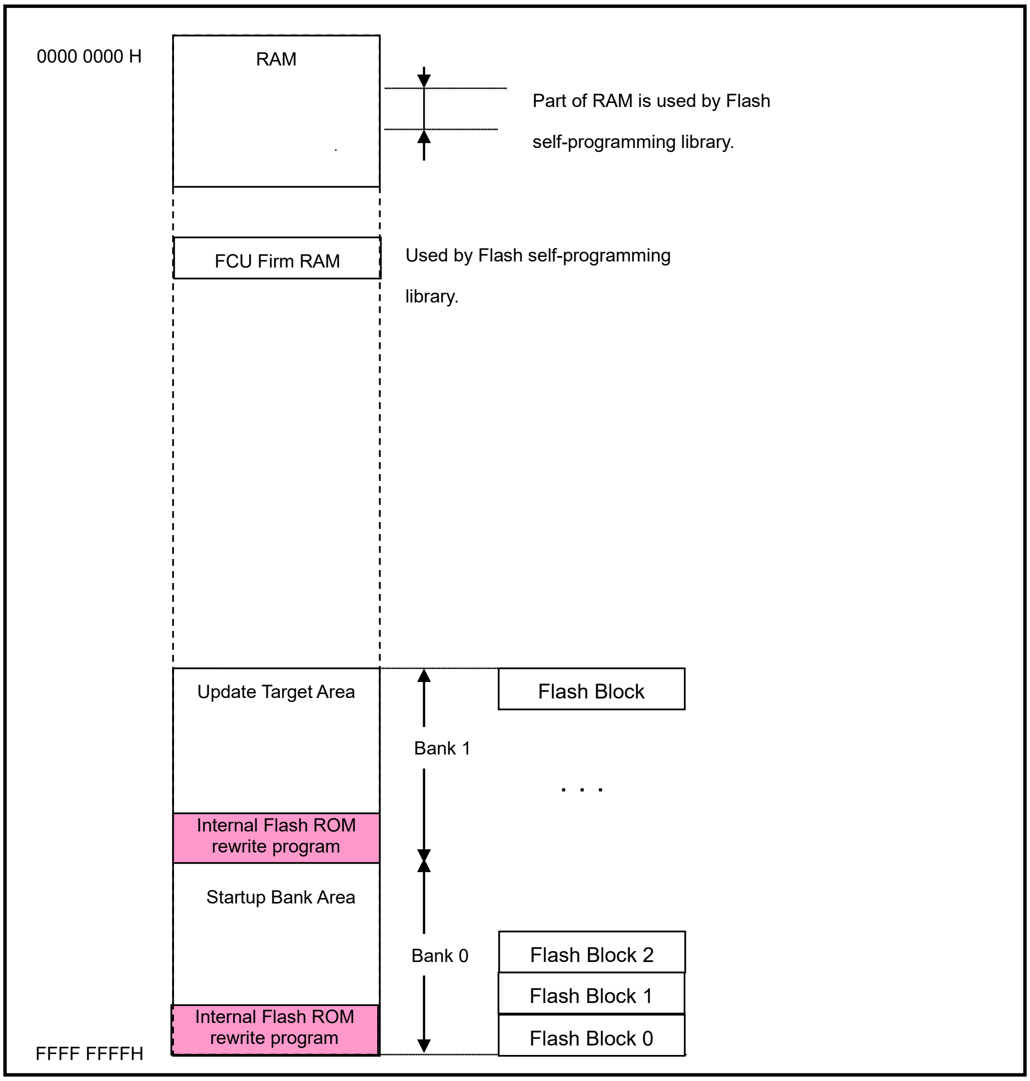


Figure 4-8 Memory Map (Dual Mode Used)

4.3 Execution of Internal Flash ROM rewrite program via USB CDC (user program write operation)

This section explains the sequence for **Internal Flash ROM rewrite program via USB CDC** execution and user program write operation.

(1). Hardware preparation

To execute the write operation, detach the emulator and connect the PC and evaluation board with the USB cable. **Figure 4-9** shows the connection diagram.

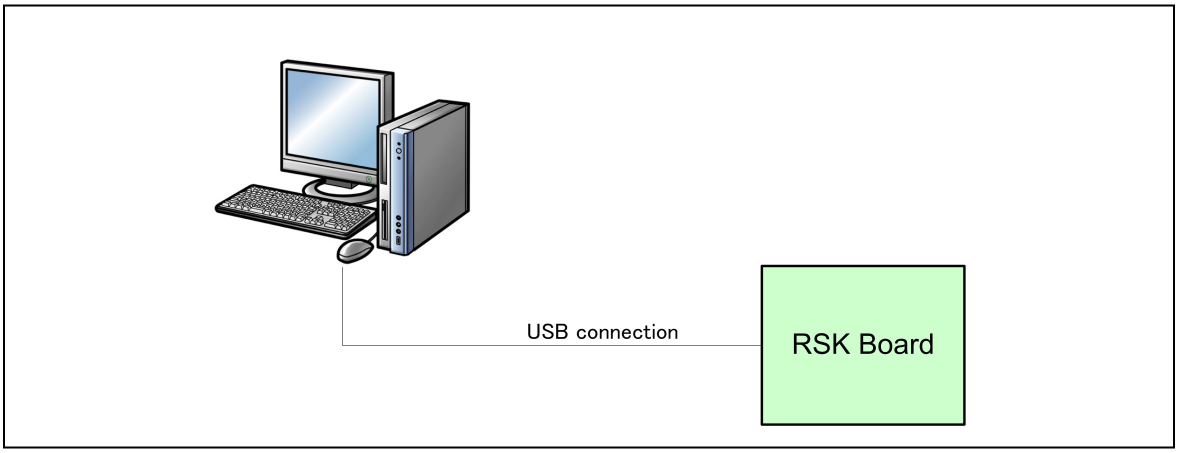


Figure 4-9 PC-Evaluation Board Connection Diagram

(2). Internal Flash ROM rewrite program via USB CDC startup

Press the RESET button while holding down switch SW3 on the evaluation board. After transitioning to program mode, the board will wait for transfer data from the PC.

Note:

a. Don't detach the USB cable while erasing or writing of the user program.

b. The PC used to run the file transfer application must be installed with a CDC driver. For details, refer to section **4.5 CDC Driver Installation.**

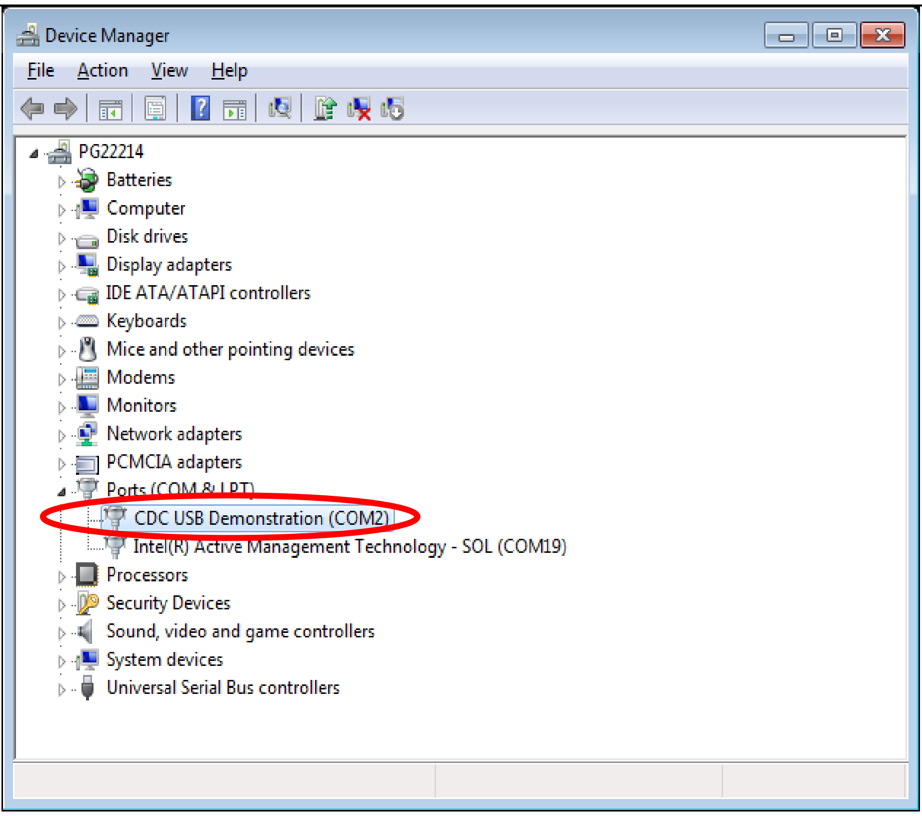
(3). File transfer preparation

Run the file transfer application (RX USB Function Firmware Updater: PC-side software). Refer to Figure 4-11 for image.

Confirm the Windows device manager under "COM:" in the updater window, and then select the assigned COM number.

Note:

The COM number varies according to environment. Numbers 1 to 9 can be used the COM number.

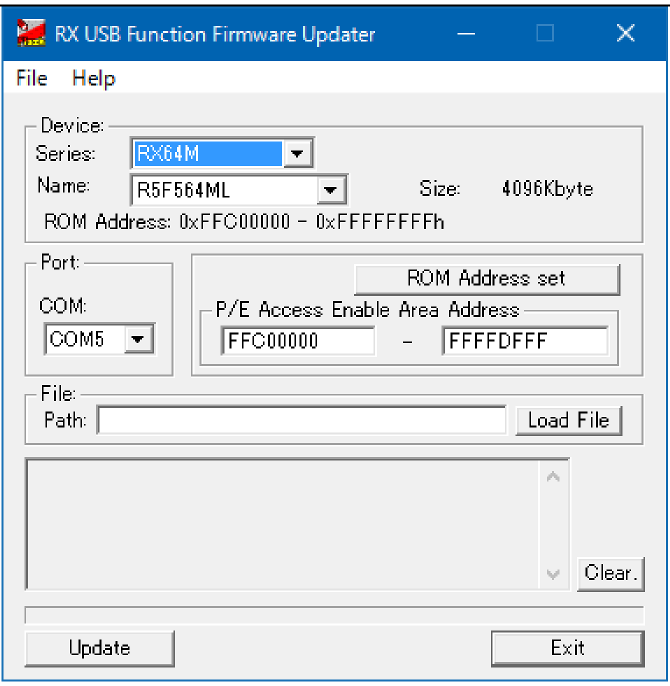


```
PG22214
  Batteries
  Computer
  Disk drives
  Display adapters
  IDE ATA/ATAPI controllers
  Keyboards
  Mice and other pointing devices
  Modems
  Monitors
  Network adapters
  PCMCIA adapters
  Ports (COM & LPT)
    CDC USB Demonstration (COM2)
    Intel(R) Active Management Technology - SOL (COM19)
  Processors
  Security Devices
  Sound, video and game controllers
  System devices
  Universal Serial Bus controllers
```

Figure 4-10 Device Manager Port Confirmation

(4). Transfer file selection

Click the **Load File** button in the file transfer application (RX USB Function Firmware Updater: PC-side software) and select the file to be written to the ROM. Then select the target MCU under **Device:**.



Device:
Series: RX64M
Name: R5F564ML Size: 4096Kbyte
ROM Address: 0xFFC00000 - 0xFFFFFFFFh

Port:
COM: COM5

P/E Access Enable Area Address
FFC00000 - FFFFDFFF

File:
Path:

Figure 4-11 RX USB Firmware Updater GUI Software (file transfer application)

For details on how to use the file transfer application, refer to section **4.1 File Transfer Application (RX USB Function Firmware Updater) Startup.**

(5). P/E limited area setting (P/E Enable Address setting)

Next, set the Program/Erase Enable Area within the ROM. For details, refer to section **4.1.1 P/E Access Enable Area Address**.

Sequence:

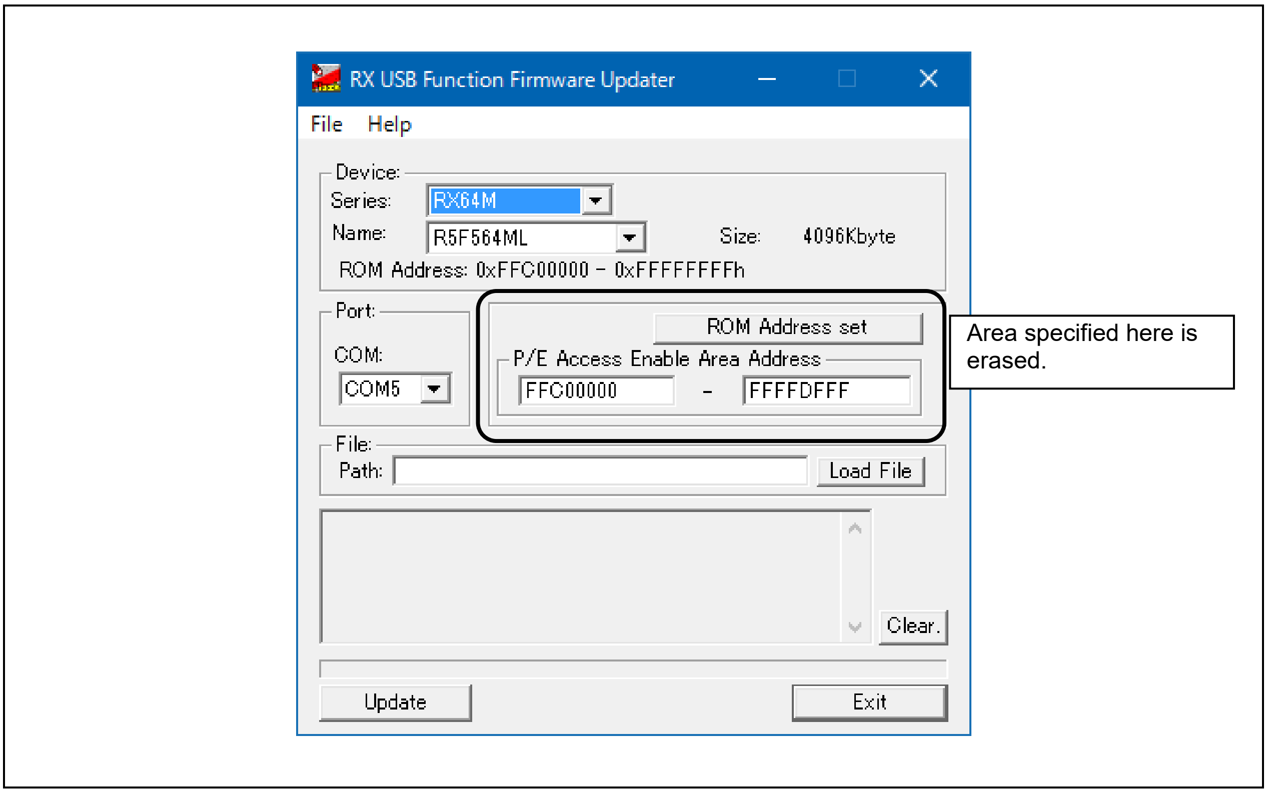


Figure 4-12 P/E Limited Area Setting

(6). User program transfer execution

Click the **Update** button in the file transfer application GUI window. This will display the start message and start the file transfer or write operation processing.

Note:

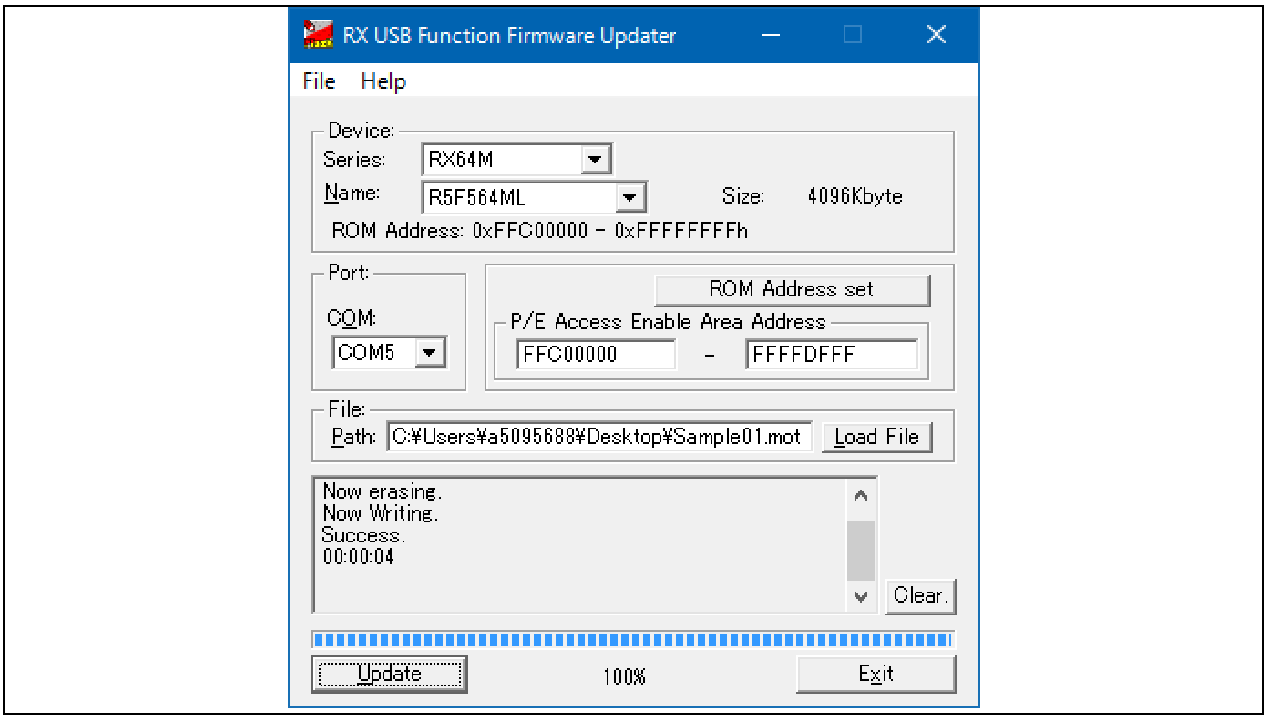
a. Don't detach the USB cable while programming the user program. If the USB cable was detached, you need to reset the RX MCU.

b. If the user program write operation fails, the file transfer application interface will show a corresponding message. See section **8.4, Application Messages** for detailed explanations.

c. In using dual mode, if the copying processing of **Internal Flash ROM rewrite program via USB CDC** to the update target area described in (3) in section **4.2.1** is failure, The message " ERR:Flash ROM rewrite program does not exist in update taget area." is displayed on the file transfer appication (PC tool)

(7). User program transfer complete

When the file transfer or write operation processing ends, the file transfer application interface will display "Success" to indicate the operation is complete. This ends the full write operation processing. Note that when dual mode is selected, bank switching will be performed by the **Internal Flash ROM rewrite program via USB CDC** if the writing of the user program to the update target area exits normally. Also note that bank switching will not be performed by the **Internal Flash ROM rewrite program via USB CDC** if the writing of the user program to the update target area fails.



Device:
Series: RX64M
Name: R5F564ML
Size: 4096Kbyte
ROM Address: 0xFFC00000 - 0xFFFFFFFFh

Port:
COM: COM5

P/E Access Enable Area Address
FFC00000 - FFFFDFFF

File:
Path: C:¥Users¥a5095688¥Desktop¥Sample01.mot

```
Now erasing.
Now Writing.
Success.
00:00:04
```

100%

Figure 4-13 Write Processing Complete

(8). User program startup

When the rewrite operation is completed, a software reset is executed automatically and the written user program is started.

When sample program 1 (user program) has been written to the MCU, the LEDs on the RSK/RSSK board light up in consecutive order. Note that when dual mode is selected, the user program that was written to the update target area will be launched if the writing of the user program in step (7) above exits normally. Also note that the user program that was previously present in the startup bank area will be launched if the writing of the user program fails.

(9). User program rewrite operation

This step rewrites the user program. Prepare sample program 2 (user program), restart the **Internal Flash ROM rewrite program via USB CDC**, and repeat the sequence from step (4).

(10). Rewrite complete

When the rewrite operation is complete, the evaluation board is reset, and the new user program is started. The RSK/RSSK board LEDs light up if sample program 2 (user program) is written.

4.4 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 CDC**, please start over by re-writing **Internal Flash ROM rewrite program via USB CDC**.

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

2. Be careful not to erase any block that includes the reset vector. **Internal Flash ROM rewrite program via USB CDC** will not run if the reset vector has been erased.

4.5 CDC Driver Installation

The PC used to run the file transfer application must be installed with a CDC driver. The wizard shown in Figure 4-14 will appear on your screen and prompt the CDC driver installation when you connect your PC to target board used to write **Internal Flash ROM rewrite program via USB CDC** to the MCU.

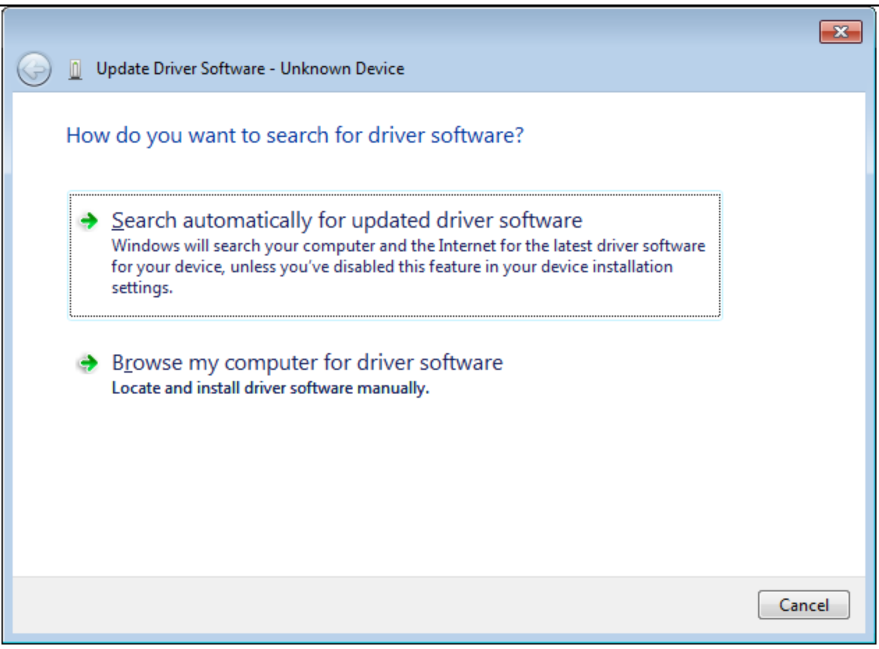
(1). Select **Update Driver Software** from the device manager.

(2). Select "**Browse my computer for driver software**".

Note:

a. It is not necessary the following installation work for CDC driver when using Window® 10.

b. The catalog file with the digital signature is required when using Windows® 8.1. The customer needs to create this catalog file.



Update Driver Software - Unknown Device

How do you want to search for driver software?

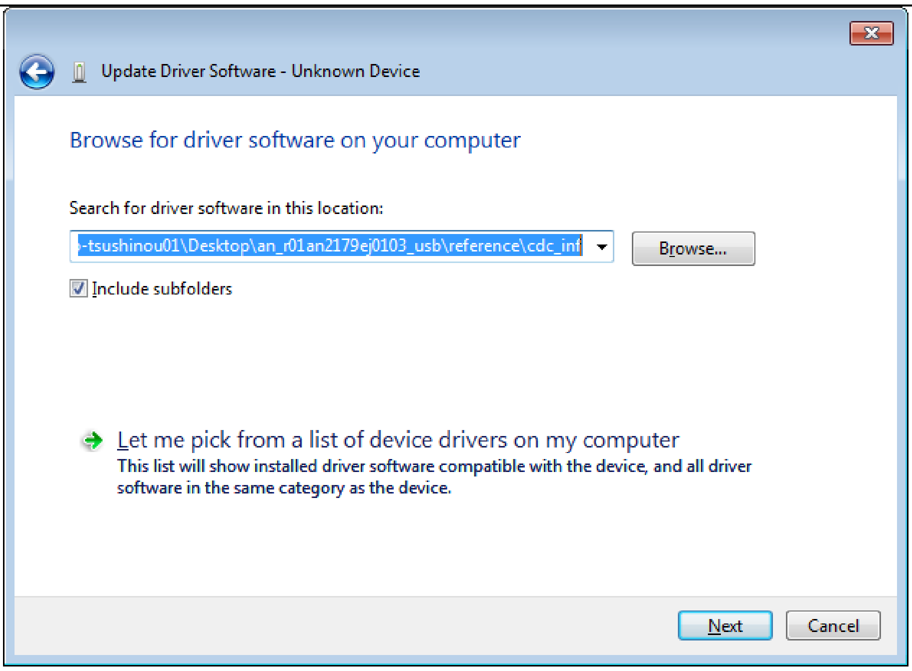
Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

Figure 4-14 New Hardware Search Wizard

(3). Select **“Browse for driver software on your computer"**

Click **Browse**, specify the folder in which the *CDC_Demo.inf* is stored, then click **“Next”**



Update Driver Software - Unknown Device

Browse for driver software on your computer

Search for driver software in this location:

-tsushinou01\Desktop\an_r01an2179ej0103_usb\reference\cdc_inf

☑ Include subfolders

Let me pick from a list of device drivers on my computer

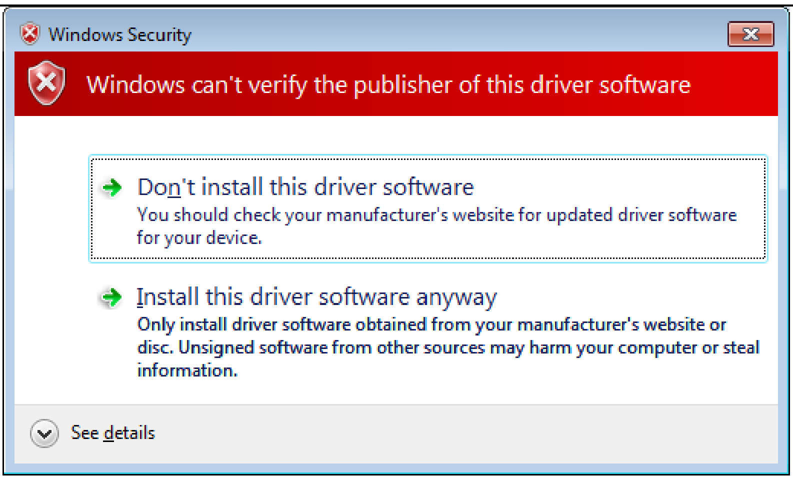
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Figure 4-15 Select Driver Location

Note:

The *CDC_Demo.inf* file is stored in "reference¥cdc_inf" in the package.

(4). If the following installation confirmation screen appears, click **“Browse for driver software on your computer”**



Windows Security

Windows can't verify the publisher of this driver software

Don't install this driver software

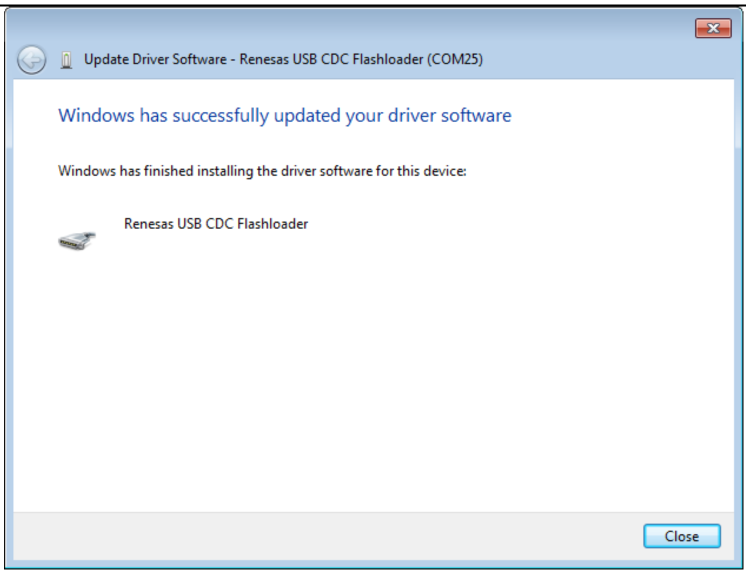
You should check your manufacturer's website for updated driver software for your device.

Install this driver software anyway

Only install driver software obtained from your manufacturer's website or disc. Unsigned software from other sources may harm your computer or steal information.

Figure 4-16 Installation Confirmation Screen

(5). When the following window appears, the CDC driver has been successfully installed. Click **“Close.”**



Update Driver Software - Renesas USB CDC Flashloader (COM25)

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

Renesas USB CDC Flashloader

Figure 4-17 Installation Complete

* An error may occur when installing the driver in the Windows 8.1 environment. In this case the installation confirmation screen will not appear.

5. Cautions Regarding Creating the User Program

This sections explains cautions that apply when creating the user program

5.1 File Format

The program supports the following file formats.

・Motorola S format

・Intel HEX format

Note:

This program only supports the file with the load addresses in ascending order and does not supports the file with addresses in descending order, or addresses before and after.

5.2 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 Settings** for details on how to create the UserApp Header area.

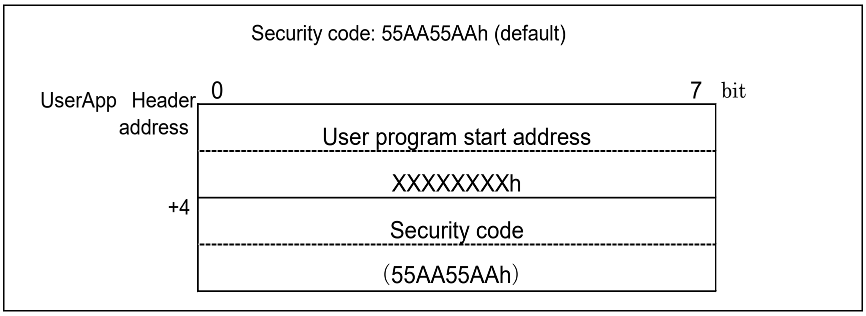


Figure 5-1 UserApp Header Area

This header information is read when **Internal Flash ROM rewrite program via USB CDC** is started up and transitions to the UserApp startup sequence. For details, refer to section **7.3.1 Power On / Reset Operation Flow.**

5.3 Fixed Vectors

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

Note:

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

5.4 Option-Setting Memory

Do not make any settings to the option setting memory in the user program when using the MCU with option-setting memory. If there is the setting to the option-setting memory in the user program (mot/hex file), this program does not work properly.

Note:

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

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 **7.2, Backup Function.**

6. Internal Flash ROM rewrite program via USB CDC and User Program Settings

This section provides the setting contents required for **Internal Flash ROM rewrite program via USB CDC** and the user program.

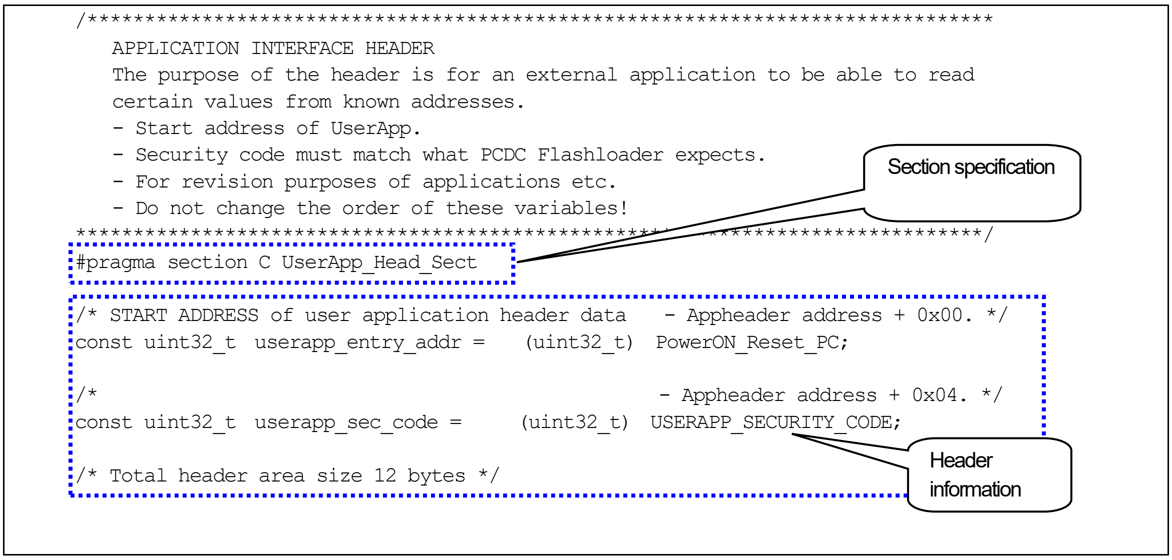
6.1 User Program Settings

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 **5.2 UserApp Header Area (user application header)**.

2. Setting Content 2

   Set the section for the UserApp Header area created in step 1 above, and make sure to allocate the section to the start of the user program. Specify the start address of the Flash ROM block address to the allocated address.

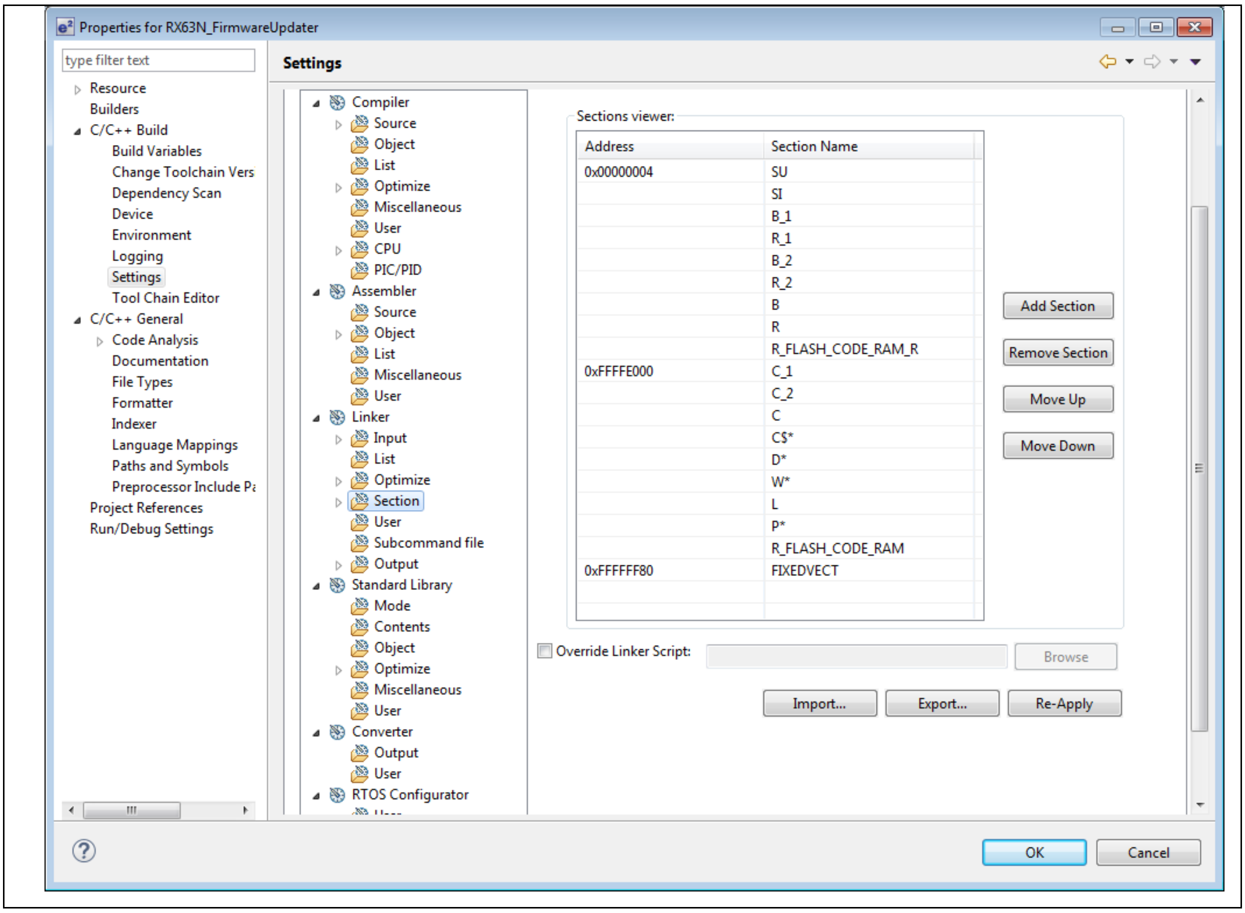


```
/*******************************************************************************
   APPLICATION INTERFACE HEADER
   The purpose of the header is for an external application to be able to read
   certain values from known addresses.
   - Start address of UserApp.
   - Security code must match what 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)  PowerON_Reset_PC;

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

/* Total header area size 12 bytes */
```

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].



Properties for RX63N_FirmwareUpdater

Settings

Sections viewer:

Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
	R_FLASH_CODE_RAM_R
0xFFFFE000	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	P*
	R_FLASH_CODE_RAM
0xFFFFFF80	FIXEDVECT

☐ Override Linker Script:

Figure 6-2 Example of Section Settings for Sample Program

6.2 Internal Flash ROM rewrite program via USB CDC Settings

1. Setting Content 1

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

(1). 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.

(2). Vendor ID and Product ID setting

Specify your vendor ID and product ID to the *USB_CFG_VENDOR_ID* and *USB_CFG_PRODUCT_ID* definition.

```
#define    USB_CFG_VENDOR_ID     0x0000    // Vendor ID setting
#define    USB_CFG_PRODUCT_ID    0x0002    // Product ID setting
```

Notes:

a. Be sure to set your vendor ID to *USB_CFG_VENDOR_ID* definition.

b. Be sure to set the setting value to the above macro defition to INF file (PC side).

(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 **7.2, Backup Function.**

(4). USB Pipe setting

Specify the pipe number to use for data transfer.

a. Bulk IN/OUT transfer

Set the pipe number (PIPE1 to PIPE5) to use for Bulk IN/OUT transfer. Do not set the same pipe number for the definitions of *USB_CFG_PCDC_BULK_IN* and *USB_CFG_PCDC_BULK_OUT*.

```
#define    USB_CFG_PCDC_BULK_IN     Pipe number (USB_PIPE1 to USB_PIPE5)
#define    USB_CFG_PCDC_BULK_OUT    Pipe number (USB_PIPE1 to USB_PIPE5)
```

b. Interrupt IN transfer

Set the pipe number (PIPE6 to PIPE9) to use for Interrupt IN transfer.

```
#define    USB_CFG_PCDC_INT_IN      Pipe number (USB_PIPE6 to USB_PIPE9)
```

(5). USB Power setting

Specify *USB_CFG_BUS* or *USB_CFG_SELF* to the following definition.

```
#define     USB_CFG_POWER          USB_CFG_BUS       // Bus Power Setting
#define     USB_CFG_POWER          USB_CFG_SELF      // Self Power Setting
```

(6). 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.

(7). 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.

(8). 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
#define     USB_CFG_REGULATOR      USB_CFG_ON        // Use
```

Note:

This definition is ignored when using MCU except RX231.

(9). Other setting

Internal Flash ROM rewrite program via USB CDC 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 **5.2 UserApp Header Area (user application header)** about UserApp Header area.

a. **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. **USERAPP_SECURITY_CODE** definition setting

Set the security code 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

This program jumps to **Internal Flash ROM rewrite program via USB CDC** or the user program depending on the state of SW (Switch) on the evaluation board. 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.

4. Setting Content 4 (option setting memory)

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

File Name	:	demo_src¥main.c
Function Name	:	usb_pin_setting()

5. Setting Content 5 (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.

6. Setting Content 6 (compile option)

Set the following compile options for the compile to be executed after steps 1 to 4 described above.

(1). When assigning the firmware update program 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 the firmware update program to the user boot area:

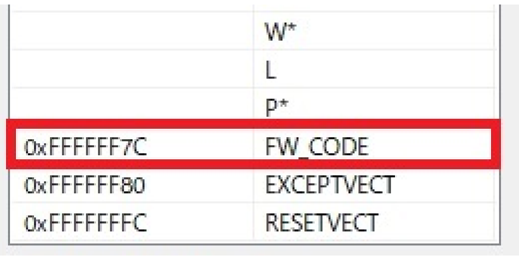
Select **None** as **Branch width size** in the e^2 studio

Note:

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

7. Setting Content 7 (Selecting Dual Mode)

If you specify the dual mode in the above "Setting Content 2", add *FW_CODE* section on the address 0xFFFFFF7C.



	W*
	L
	P*
0xFFFFFF7C	FW_CODE
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

6.3 User Program Position

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 CDC**. Assign the user program locations according to section settings.

Note:

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	P/E address Setting		
OFF	On-chip ROM Area (Program ROM) Start Address	-	0xFFFFDFFF
ON	Start Address of Program Execution Area	-	0xFFFFDFFF

Note:

For the backup function and the program execution area, refer to section **7.2, Backup Function.**.

2. The 4 bytes area from 0xFFFFDFFC to 0xFFFFDFFF is used as the management area by **Internal Flash ROM rewrite program via USB CDC.**
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 CDC** and will not affect the user program operations.

7. Internal Flash ROM rewrite program via USB CDC Explanation

This section explains each file used by **Internal Flash ROM rewrite program via USB CDC**.

7.1 File/Folder Configuration

The following shows the source file/folder configuration of this program.

```
(MCU name)
   +HardwareDebug                                          Build result
   +src
     +———r_config           [API setting file]
     |
     +———r_flash_rx         [Simple Flash API]
     |        +—— src                                       [FlashAPI driver]
     |               +—flash_type_1                        Flash write type 1 API
     |               +—flash_type_2                        Flash write type 2 API
     |               +—flash_type_3                        Flash write type 3 API
     |               +—flash_type_4                        Flash write type 4 API
     |               +—targets                             ROM information for each MCU
     +———r_bsp              [Renesas Board Support Package]
     |        +——— board                                   BSP setting for each RSK/RSSK
     |        +——— mcu                                     BSP setting for each MCU
     +———demo_src           [Sample application]
     |        +——— inc                                      Sample application program
     +———USB                [USB driver]
                +——— inc                                    USB driver common header file
                +——— src                                    USB driver
```

Figure 7-1 Internal Flash ROM rewrite program via USB CDC Folder Configuration

This program uses the following packages.

- r_bsp (Renesas board support package)
- r_flash_rx (RX family simple flash module)

7.1.1 src¥r_config Folder

This folder stores all the setting files for the target MCU.

Table 7-1 API Header Files

File Name	Description
r_bsp_config.h	BSP setting header file
r_flash_rx_config.h	Flash write setting file
r_usb_fwupdater_config.h	Flash ROM rewrite program setting file

7.1.2 src¥r_flash_rx Folder

This folder stores the simple flash API source files and header files. For more details, refer to the Flash Module Using Firmware Integration Technology application note.

The flash write type is automatically selected when the MCU is selected in the board support package (r_bsp).

7.1.3 src¥r_bsp Folder

This folder stores the Renesas Board support package module source files and header files. For more details, refer to the RX Family Support Package Module Application Note.

7.1.4 src¥demo_src Folder

This folder stores **Internal Flash ROM rewrite program via USB CDC** source files.

Table 7-2 Internal Flash ROM rewrite program via USB CDC Source Files

File Name	Description
main.c	C language main function description file
r_usb_pcdc_apl.c	USB data transfer processing file
r_fwupdater_apl.c	Flash ROM rewrite program processing file
r_flash_apl.c	Flash API calling processing file (Flash ROM rewriting processing)
r_usb_descriptor.c	USB descriptor definition file
inc¥r_usb_pcdc_apl.h	USB data transfer processing header file
inc¥r_fwupdater_apl.h	Flash ROM rewrite program processing header file
inc¥r_flash_apl.h	Flash ROM rewrite program header file

7.1.5 src¥USB Folder

This folder stores the CDC (USB) source files and header file.

Table 7-3 Internal Flash ROM rewrite program via USB CDC Source Files

File Name	Description
inc¥r_usb_reg.h	USB register initialization, setting definitions
inc¥r_usb_define.h	USB definition
inc¥r_usb_extern.h	Function Extern
src¥r_usb_api.c	USB transmit/receive, initialization processing file
src¥r_usb_driver.c	USB driver processing
src¥r_usb_classcdc.c	USB CDC processing
src¥r_usb_rx_mcu.c	USB interrupt initialization, port setting file
src¥r_usb_reg.c	USB register setting, etc.

7.1.6 Hardware Debug Folder

This folder stores object files and mot files of **Internal Flash ROM rewrite program via USB CDC** that can be executed during a build.

Table 7-4 Creating File

File Name	Description
*MCU name*_FirmwareUpdater.mot	mot format executable object file

7.2 Backup Function

Internal Flash ROM rewrite program via USB CDC 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 CDC** 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 CDC**, which is present in Area 1, is not present in Area 2.

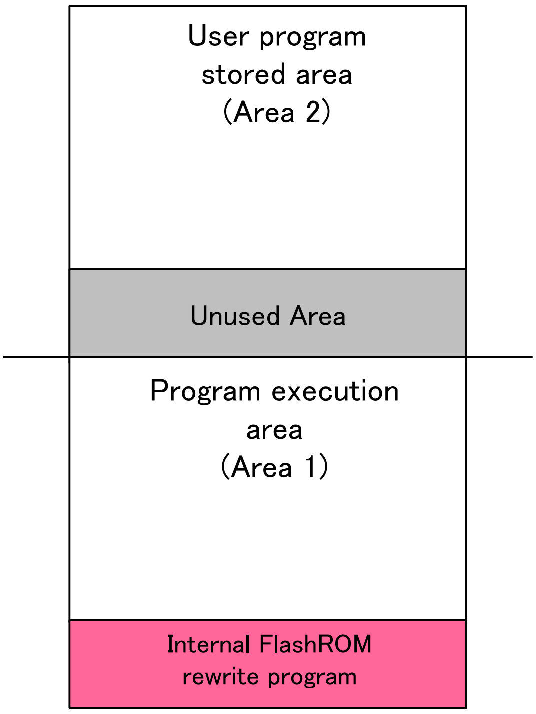
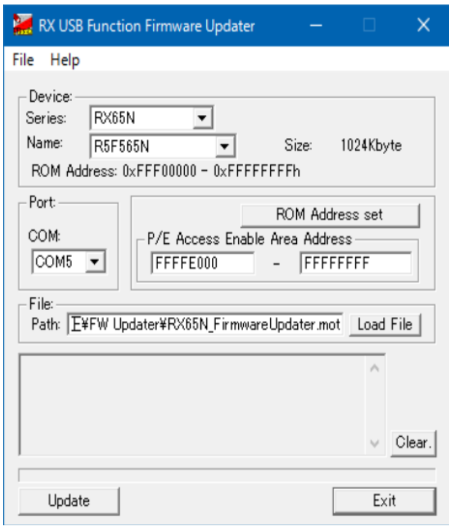


Figure 7-2 Flash ROM Area When Using Backup Area

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

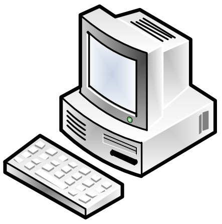

RX USB Function Firmware Updater

Device:
Series: RX65N
Name: R5F565N
Size: 1024Kbyte
ROM Address: 0xFFF00000 - 0xFFFFFFFFh

Port:
COM: COM5

P/E Access Enable Area Address
FFFFE000 - FFFFFFFF

File:
Path: E¥FW Updater¥RX65N_FirmwareUpdater.mot



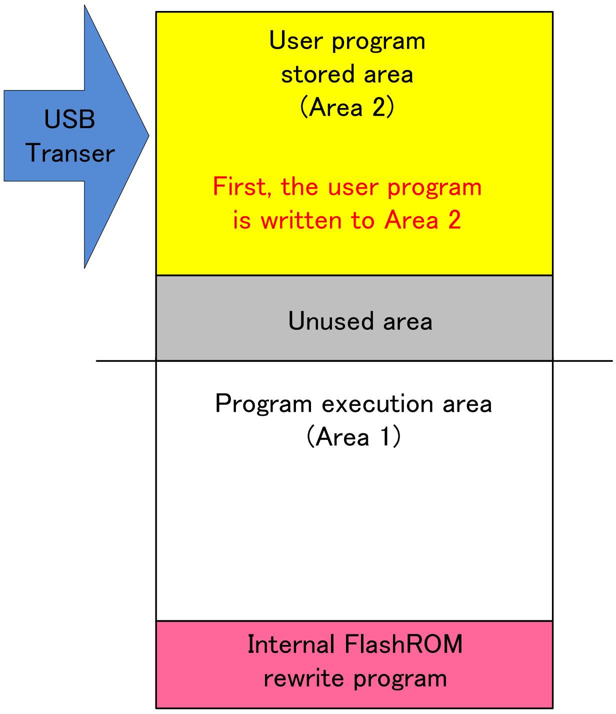


Figure 7-3 Writing of User Program to On-Chip Flash ROM (Area 2)

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

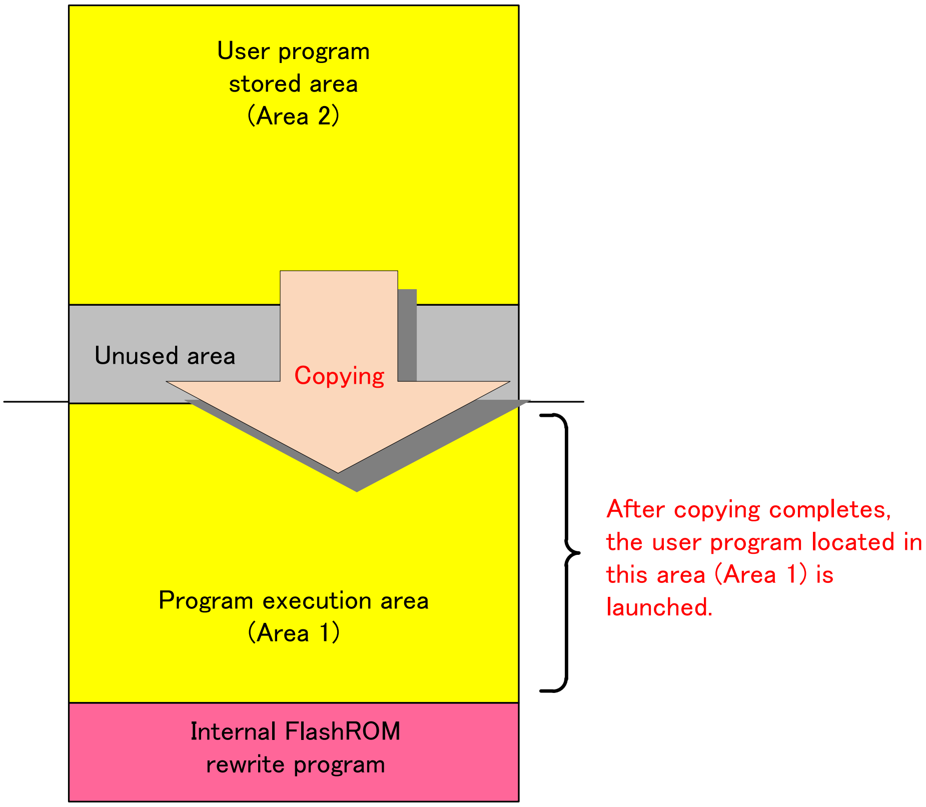


Figure 7-4 Copying of User Program

4. When **Internal Flash ROM rewrite program via USB CDC** is used to update the user program, it first erases Area 2 of the flash ROM, then writes the user program to Area 2, and finally, after writing completes, copies the user program to Area 1.

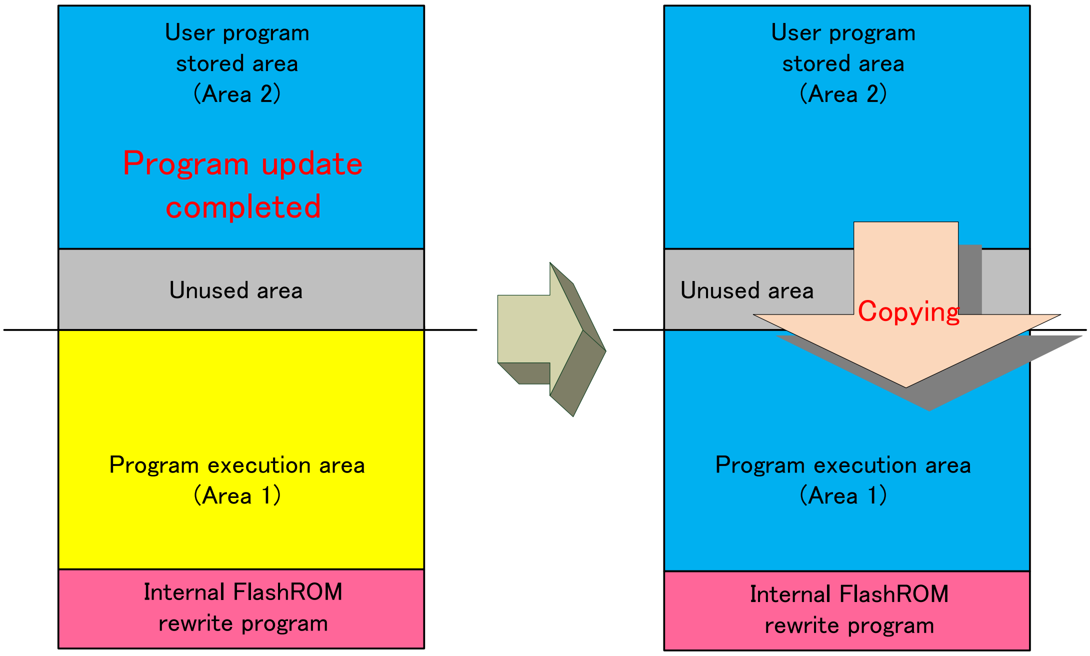


Figure 7-5 Updating the User Program

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.) When this program can not erase Area 1, The message "ERR: Writing process stop." or "ERR: Data reception error." is displayed on the file transfer application (PC tool).

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.

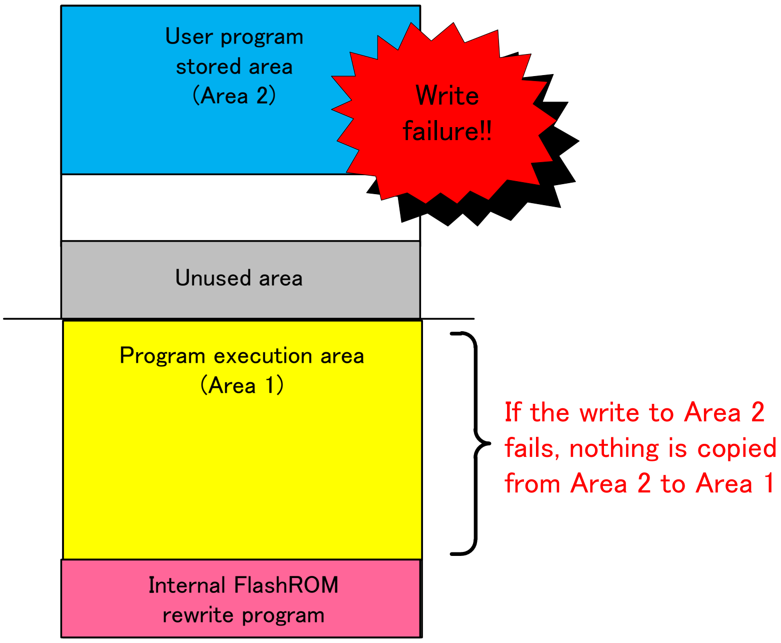


Figure 7-6 User Program Update Failure

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.

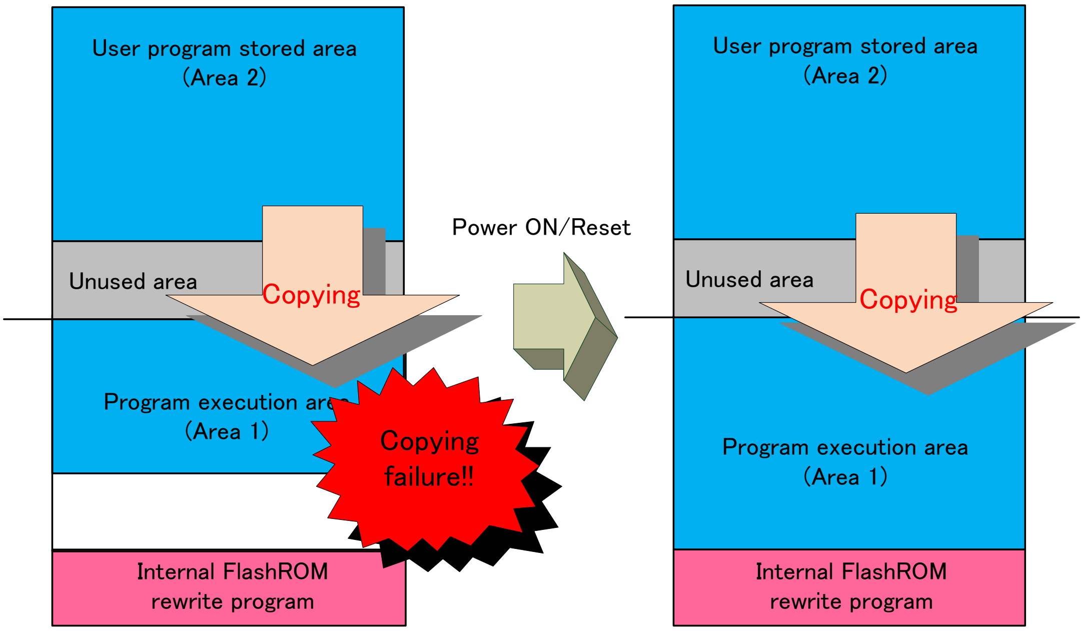


Figure 7-7 Failure to Copy from Area 2 to Area 1

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 CDC Settings**.

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

7.3 Boot Processing

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:

- Allocate stack area and set stack pointer
- Allocate argument area for main function
- Initialize data area and stack area
- Branch to user program and initialize MCU peripheral devices in hdwinit function
- Branch to main function

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

7.3.1 Power On / Reset Operation Flow

This section explains the operation flow after power is turned on for **Internal Flash ROM rewrite program via USB CDC**.

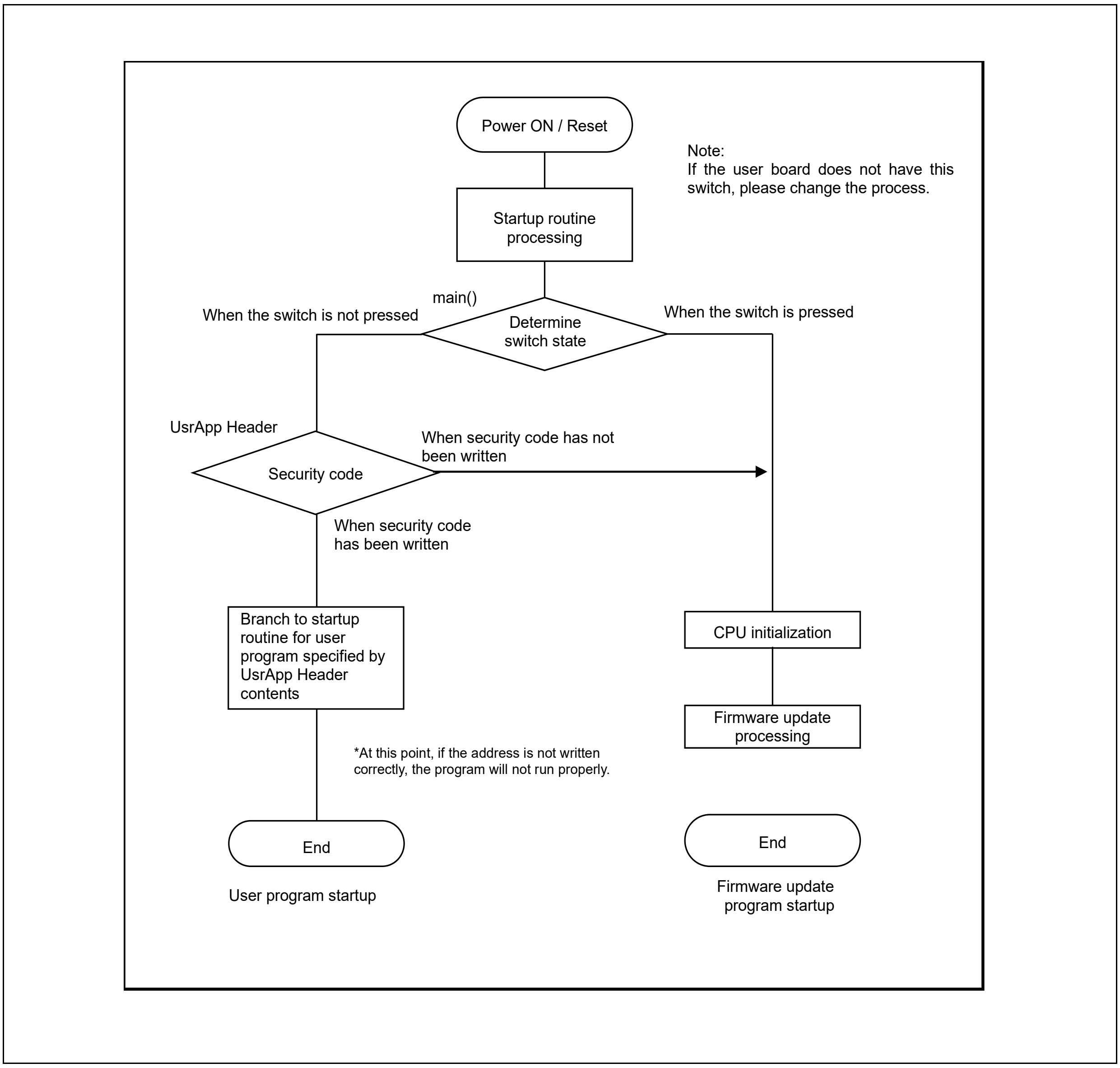

Power ON / Reset
Note:
If the user board does not have this switch, please change the process.
Startup routine processing
main()
When the switch is not pressed
Determine switch state
When the switch is pressed
UsrApp Header
Security code
When security code has not been written
When security code has been written
Branch to startup routine for user program specified by UsrApp Header contents
CPU initialization
Firmware update processing
*At this point, if the address is not written correctly, the program will not run properly.
End
End
User program startup
Firmware update program startup

Figure 7-8 Power On / Reset Operation Flow

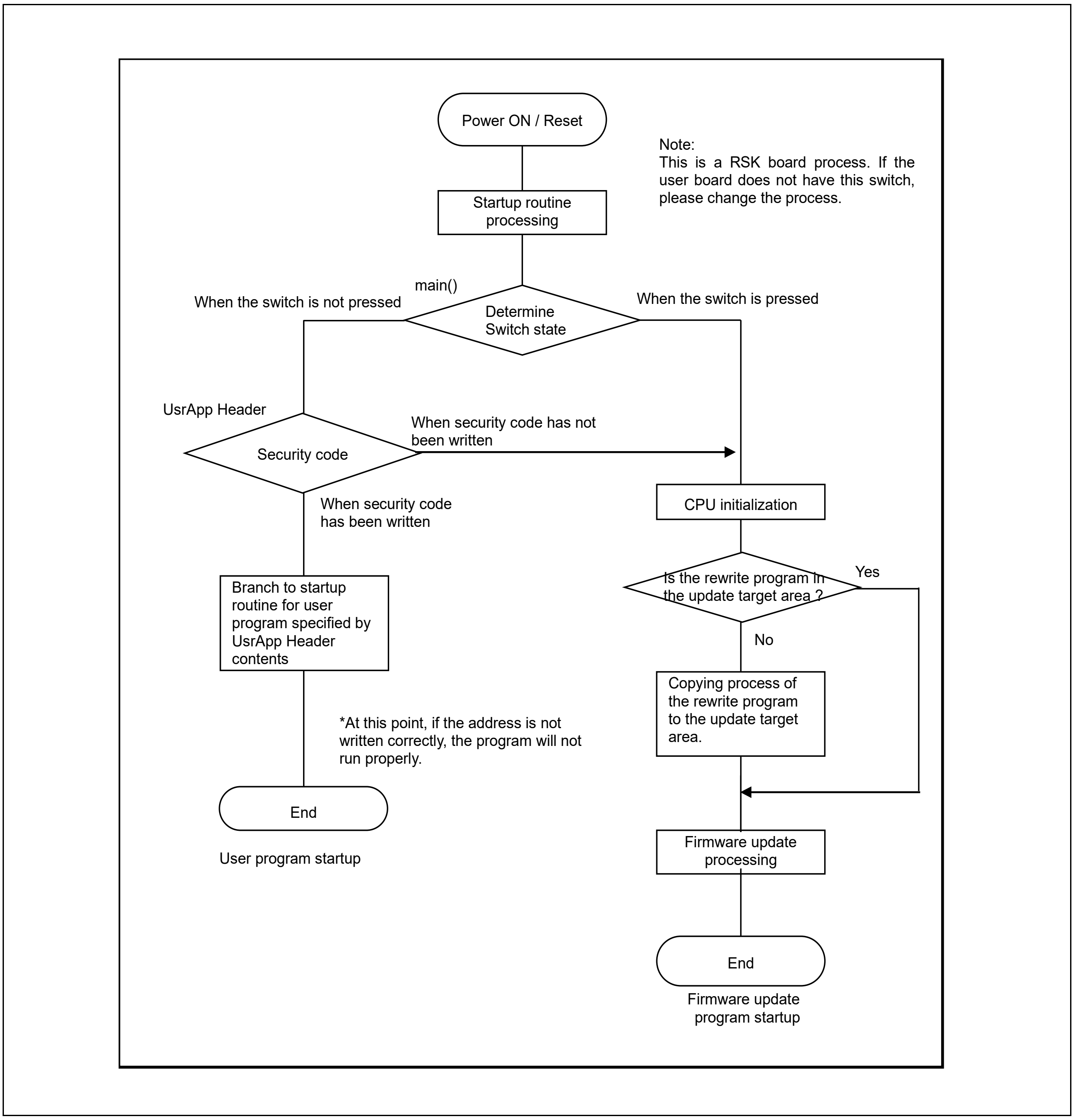

Power ON / Reset
Note:
This is a RSK board process. If the user board does not have this switch, please change the process.
Startup routine processing
main()
When the switch is not pressed
Determine Switch state
When the switch is pressed
UsrApp Header
Security code
When security code has not been written
When security code has been written
CPU initialization
Is the rewrite program in the update target area ?
Yes
No
Branch to startup routine for user program specified by UsrApp Header contents
Copying process of the rewrite program to the update target area.
*At this point, if the address is not written correctly, the program will not run properly.
End
User program startup
Firmware update processing
End
Firmware update program startup

Figure 7-9 Power On / Reset Operation Flow (Using Dual mode)

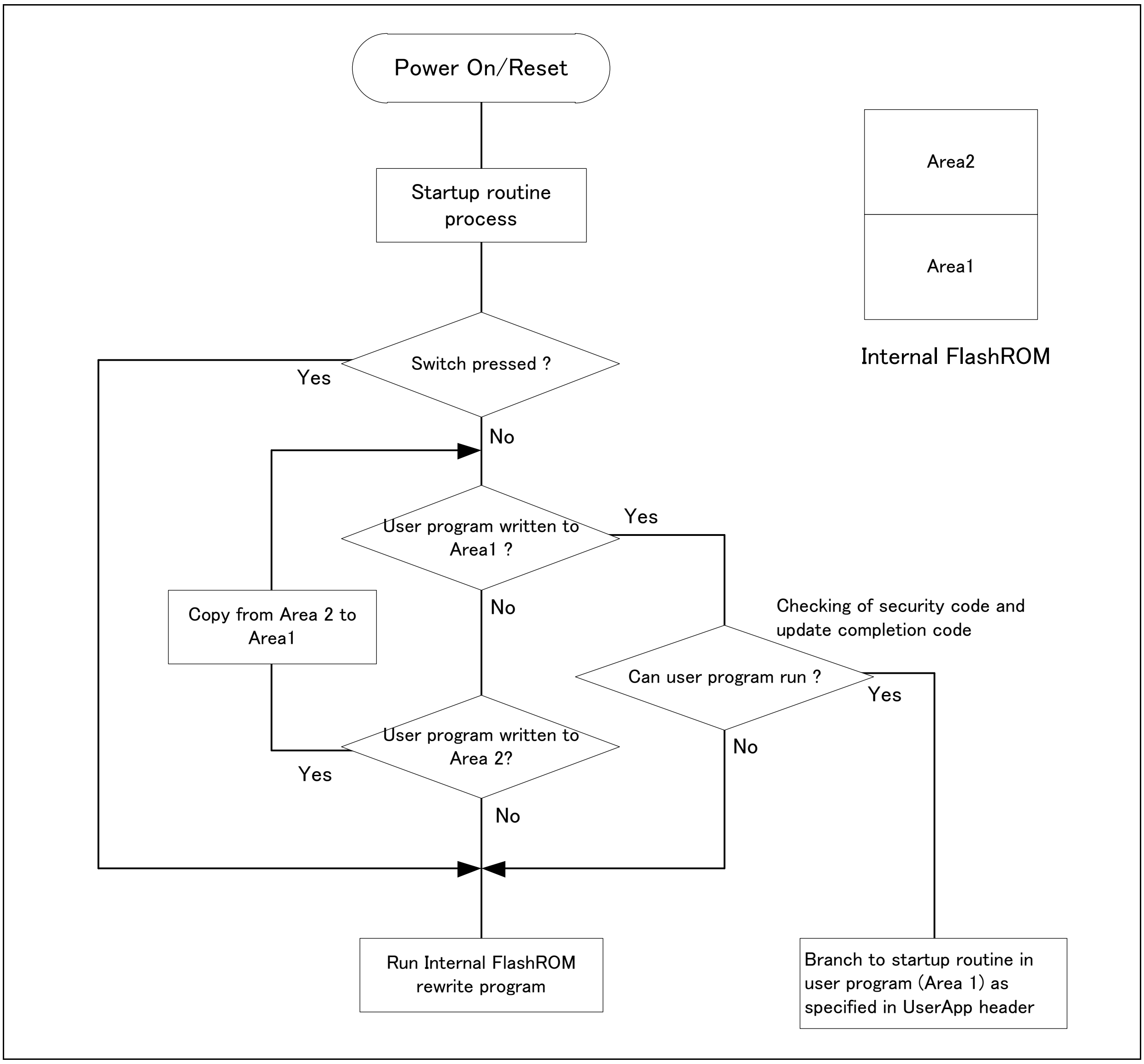

Power On/Reset
Startup routine process
Switch pressed ?
Yes
No
User program written to Area1 ?
Yes
No
Copy from Area 2 to Area1
Checking of security code and update completion code
Can user program run ?
Yes
No
User program written to Area 2?
Yes
No
Run Internal FlashROM rewrite program
Branch to startup routine in user program (Area 1) as specified in UserApp header
Area2
Area1
Internal FlashROM

Figure 7-10 Power On / Reset Operation Flow (Using the backup function)

For information regarding branch address to security code and user program, refer to section **5.2 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.

7.3.2 User program startup conditions

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

a. Correct security code is set

b. Correct user program start address is set

c. Update completion code is written properly

This rewrite program writes the update completion code automatically when the user program writing completes properly.

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

7.3.3 Internal Flash ROM rewrite program via USB CDC startup conditions

1. When user program has been written to ROM:

   The **Internal Flash ROM rewrite program via USB CDC** starts up when RESET is executed while the switch (RSK: Switch3, RSSK: Switch2) on the evaluation board is pressed.

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

   The **Internal Flash ROM rewrite program via USB CDC** starts up when power is turned on.

7.4 Cautions

1. **Internal Flash ROM rewrite program via USB CDC** determines whether to jump to the user program or continue on with the firmware update program by judging the state of the switch (RSK: Switch3, RSSK: Switch2) on the evaluation board. This judgment process is dependent on the board's specifications. Please change the judgment process to meet the specifications of your evaluation board. The judgment processing is performed in the main function of the **Internal Flash ROM rewrite program via USB CDC**.

2. Note that a check is not performed as to whether or not the addresses to which the **Internal Flash ROM rewrite program via USB CDC** is to be written is within the Flash ROM area.

7.5 Functions for Internal Flash ROM rewrite program via USB CDC

This section describes all functions used in the Updater other than BSP and simple Flash API-related functions.

7.5.1 Data Type

Data types applicable in **Internal Flash ROM rewrite program via USB CDC** are listed below.

Table 7-5 Data Type

Data Type	Specifier	Valid Range
int8_t	signed char	Signed 8-bit integer
int16_t	signed short	Signed 16-bit integer
int32_t	signed long	Signed 32-bit integer
uint8_t	unsigned char	Unsigned 8-bit integer
uint16_t	unsigned short	Unsigned 16-bit integer
uint32_t	unsigned long	Unsigned 32-bit integer

7.5.2 Structures

Table 7-6 *response_record_t* Structure Definition

Data Type	Variable Name	Description
uint32_t	record_type	Record type
uint8_t	record_len	Record length
uint8_t	response_type	Response type ACK/NAK
uint8_t	err_field	Error code
uint8_t	checksum	Check sum

Table 7-7 *rom_rewrite_buf_t* Strucure Definition

Data Type	Variable Name	Description
uint8_t	data[ROM_WRITE_SIZE]	ROM write buffer
uint32_t	dest_addr	Program destination address
uint32_t	data_flag	Data storage confirmation flag 0: None, 1: Data Storared

Table 7-8 *rom_erase_addr_t* Strucure Definition

Data Type	Variable Name	Description
uint32_t	start_addr	ROM erase start address
uint32_t	end_addr	ROM erase end address

7.5.3 Flash write main processing functions

Table 7-9 Main Processing Functions

File Name	Function Name	Processing Description
main.c	**main**	USB pin setting, judgment to jump to user program or Flash ROM rewrite program
r_usb_pcdc_apl.c	**usb_main**	Initialization, main processing
r_usb_pcdc_apl.c	**fu_cdc_read**	USB CDC data reception requirement processing
r_usb_pcdc_apl.c	**fu_main**	Flash ROM rewriting main processing
r_usb_pcdc_apl.c	**usb_send_response_record**	Data response processing to USB Host(GUI tool)
r_usb_pcdc_apl.c	**jump_to_userapp**	Jump processing to User program
r_usb_pcdc_apl.c	**usb_transfer_complete**	Transmission/Reception completion flag changing processing
r_fwupdater_apl.c	**fl_write_data_init**	Initialization processing to the variable for Flash programming
r_fwupdater_apl.c	**fl_erase_area**	Flash ROM erase processing
r_fwupdater_apl.c	**fl_write_data**	Flash ROM programming judgment, programming processing
r_fwupdater_apl.c	**fu_check_security_code**	Security code checking processing
r_fwupdater_apl.c	**fu_byte2num**	Convert4-byte address to address value in unsigned long
r_flash_apl.c	**fl_rom_write**	Calling processing the function for Flash ROM program API. Processing branches according to type.
r_flash_apl.c	**fl_rom_erase**	Calling processing the function for ROM erase API. Processing branch according to type.
r_flash_apl.c	**fl_set_access_window**	Flash ROM access enable setting processing. Flash Type 1 only.
r_flash_apl.c	**fl_get_blk_num**	Calculate number of blocks and block position information from ROM start and end addresses
r_flash_apl.c	**fl_get_blk_addr**	Calculate start address of ROM block from corresponding ROM address

Table 7-10 main()

Function Name		**main**
Description Format		void main (void)
Function		Entry function at start. Executes initialization processing and branching to **Internal Flash ROM rewrite program via USB CDC** or user program.
I/O	**Input**	None
	Output	None
Remarks		For operation details, refer to section 7.5.5 Branch to firmware update program.

Table 7-11 usb_main()

Function Name		**usb_main**
Description Format		void usb_main (void)
Function		Initialization, main processing
I/O	**Input**	None
	Output	None
Remarks		For operation details, refer to section 7.5.5, **Branch to Internal Flash ROM rewrite program via USB CDC**.

Table 7-12 fu_cdc_read()

Function Name		fu_cdc_read
Description Format		static uint16_t fu_cdc_read(void)
Function		CDC data reception detection
I/O	Input	None
	Output	uint16_t: read results
Remarks		CDC_BLK_OUT_OK: read complete CDC_NO_CONFIGUED: CDC not connected CDC_DETCH: CDC connection error CDC_BLK_OUT_ERR: read error

Table 7-13 fu_main()

Function Name		fu_main
Description Format		void fu_main (void)
Function		main processing for **Internal Flash ROM rewrite program via USB CDC**
I/O	Input	None
	Output	None
Remarks		--

Table 7-14 usb_send_response_record()

Function Name		usb_send_response_record
Description Format		static void usb_send_response_record (uint8_t response_type, uint8_t response_field)
Function		Data response processing to USB Host(GUI tool)
I/O	Input	None
	Output	None
Remarks		For details concerning communication protocol, refer to section 9 **Data Transmission Specification**.

Table 7-15 jump_to_userapp()

Function Name		jump_to_userapp
Description Format		static void jump_to_userapp (void)
Function		Jump processing to User program
I/O	Input	None
	Output	None
Remarks		For more information concerning the jump destination address, refer to section 5.2 **UserApp Header Area (user application header)**.

Table 7-16 usb_transfer_complete()

Function Name		**usb_transfer_complete**
Description Format		void usb_transfer_complete(void)
Function		Transmission/Reception completion flag changing processing
I/O	**Input**	None
	Output	None
Remarks		None

Table 7-17 fl_write_data_init()

Function Name		**fl_write_data_init**
Description Format		void fl_write_data_init(void)
Function		Initialization processing to the variable for Flash programming
I/O	**Input**	None
	Output	None
Remarks		None

Table 7-18 fl_erase_area()

Function Name		**fl_erase_area**
Description Format		flash_err_t fl_erase_area(void)
Function		Flash ROM erase processing
I/O	**Input**	None
	Output	Result of Flash ROM erasing
Remarks		None

Table 7-19 fl_write_data()

Function Name		**fl_write_data**
Description Format		flash_err_t fl_write_data(void)
Function		Flash ROM programming judgment, programming processing
I/O	**Input**	None
	Output	Result of Flash ROM programming
Remarks		None

Table 7-20 fu_check_security_code()

Function Name		**fu_check_security_code**
Description Format		flash_err_t fu_check_security_code(void)
Function		Security code checking processing

I/O	Input	None
	Output	Result of the security code checking and ROM erasing
Remarks		None

Table 7-21 fu_byte2num()

Function Name		**fu_byte2num**
Description Format		static uint32_t fu_byte2num(uint8_t * dat, uint16_t size)
Function		Convert4-byte address to address value in unsigned long
I/O	Input	Dat: byte row Size: size to be connected
	Output	Calculated results
Remarks		None

Table 7-22 fl_rom_write()

Function Name		**fl_rom_write**
Description Format		flash_err_t fl_rom_write(void)
Function		Calling processing the function for Flash ROM program API. Processing branches according to type.
I/O	Input	None
	Output	Processing result
Remarks		None

Table 7-23 fl_rom_erase()

Function Name		**fl_rom_erase**
Description Format		flash_err_t fl_rom_erase(const uint32_t start_addr, const uint32_t end_addr)
Function		Calling processing the function for Flash ROM program API. Processing branches according to type.
I/O	Input	start_addr: erase start address (erase block that includes address) end_addr: erase end address (erase block that includes address)
	Output	flash_err_t: proccessing result
Remarks		Although types 1 and 3 allow bulk erase specification, with type 2 the area limitations are judged in the API side processing and prevent the user from specifying an area that exceeds those limits for one erase. As a result, the erase operation must be specified in single blocks.

Table 7-24 fl_set_access_window()

Function Name		**fl_set_access_window**
Description Format		flash_err_t fl_set_access_window (const uint32_t start_addr, const uint32_t end_addr)
Function		Call function for ROM access enable API. Type 1 only.

I/O	Input	start_addr: ROM access enable start address end_addr ROM access enable end address
	Output	flash_err_t: processing result
Remarks		This process is only performed for Flash type 1. The access-enabled address is set assuming the end address will be truncated by 10-bits because it is retained after a 10-bit shift. This will become an access enabled area, so there will be no problems in processing a large area.

Table 7-25 fl_get_blk_num()

Function Name		**fl_get_blk_num**
Description Format		static uint32_t fl_get_blk_num(const uint32_t iaddr_start, const uint32_t iaddr_end, uint16_t *start_blk, uint16_t *end_blk)
Function		Calculate number of blocks and block position information from ROM start and end addresses
I/O	Input	iaddr_sta: start address specification iaddr_end: end address specification sta_blk: start block number sta_end: end block number
	Output	uint32_t: block count between start and end addresses
Remarks		The definition used for this function is dependent on ROM information definition of the Flash API. Please note that block numbers are assigned from the back of the ROM forward, so StartAddress=EndBlock and EndAddress=StartBlock.

Table 7-26 fl_get_blk_addr()

Function Name		**fl_get_blk_addr**
Description Format		static flash_block_address_t fl_get_blk_addr(const uint32_t iaddr)
Function		Calculate start address of ROM block from corresponding ROM address
I/O	Input	iaddr: ROM address for calculating block start address
	Output	flash_block_address_t: block start address
Remarks		The definition used for this function is dependent on ROM information definition of the Flash API.

7.5.4 USB Driver Functions

Table 7-27 lists the USB driver functions.

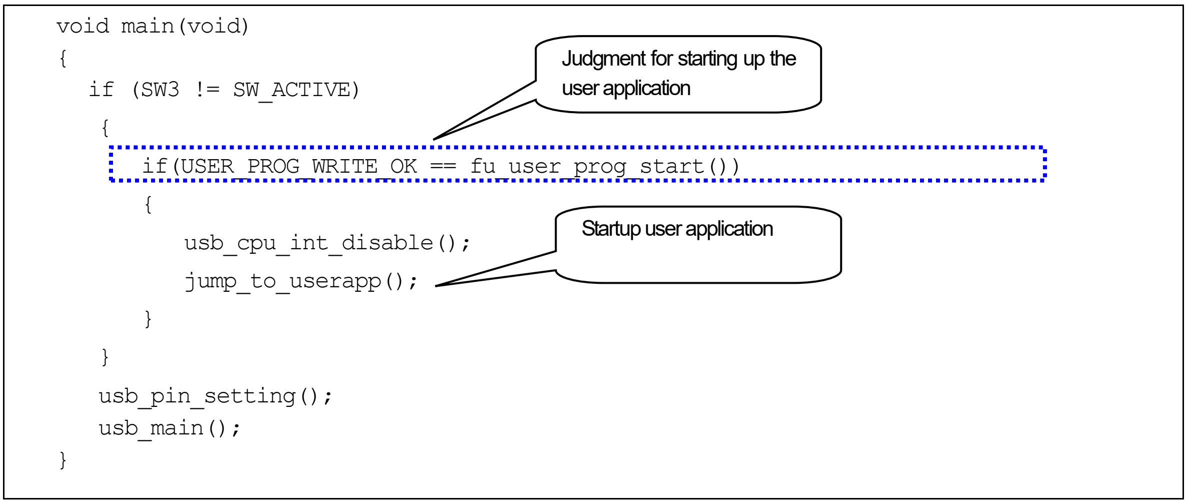
Table 7-27 USB Module Functions

File Name	Function Name	Processing Description
r_usb_api.c	**usb_bulk_in_start**	Bulk data receive request
r_usb_api.c	**usb_bulk_out_start**	Bulk data send request
r_usb_api.c	**usb_driver_init**	USB initialization processing
r_usb_driver.c	**usb_int_isr**	USB interrupt processing
r_usb_driver.c	**usb_save_request**	Get request information
r_usb_driver.c	**usb_ctrl_read_data_stage**	Control read data stage processing
r_usb_driver.c	**usb_ctrl_write_nodata_stage**	Control no-data status stage processing
r_usb_driver.c	**usb_intr_int_pipe0**	USB BRDY interrupt processing for PIPE0
r_usb_driver.c	**usb_bemp_int_pipe0**	USB BEMP interrupt processing for PIPE0
r_usb_driver.c	**usb_intr_int**	Bulk data send and receive processing
r_usb_driver.c	**usb_intr_int_read**	Bulk data receive
r_usb_driver.c	**usb_intr_int_write**	Bulk data send
r_usb_driver.c	**usb_ctr_read_start**	Control data send request
r_usb_driver.c	**usb_ctr_write_start**	Control data receive request
r_usb_driver.c	**usb_write_fifo**	Data writing to USB FIFO
r_usb_driver.c	**usb_read_fifo**	Data reading from USB FIFO
r_usb_driver.c	**usb_chk_frdy**	Checking FRDY bit in USB module
r_usb_driver.c	**usb_chg_port**	USB pipe switching processing
r_usb_driver.c	**usb_req_get_descriptor**	Standard request processing
r_usb_driver.c	**usb_req_set_configuration**	Standard request processing
r_usb_classcdc.c	**usb_reset_ep**	USB pipe configuration processing
r_usb_classcdc.c	**usb_cdc_init**	Serial initialize
r_usb_classcdc.c	**usb_class_write_data_stage**	Class request write data stage processing
r_usb_classcdc.c	**usb_class_read_data_stage**	Class request read data stage processing
r_usb_classcdc.c	**usb_class_write_nodata_stage**	Class request no-data status stage processing
r_usb_rx_mcu.c	**usb_cpu_mcu_initialize**	MCU initialization
r_usb_rx_mcu.c	**usb_int_init**	USB interrupt initialization
r_usb_rx_mcu.c	**usb_cpu_delay_1us**	Software waiting processing (us)
r_usb_rx_mcu.c	**usb_cpu_delay_1ms**	Software waiting processing (ms)
r_usb_rx_mcu.c	**usb_cpu_int_disable**	USB interrupt disable
r_usb_rx_mcu.c	**usb_cpu_usbint_init**	USB interrupt initialization

7.5.5 Branch to Internal Flash ROM rewrite program via USB CDC

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

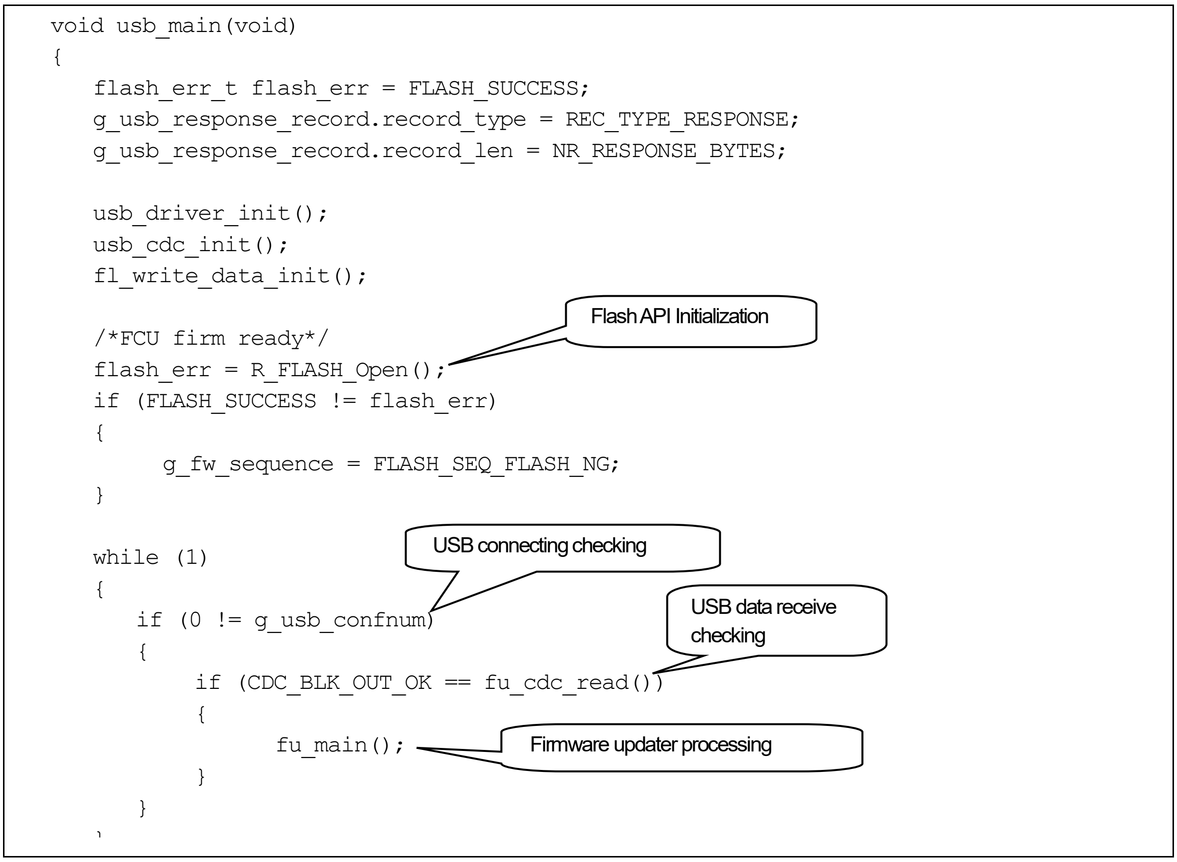
After conditional branching is performed, the CPU build-in functions and peripheral circuits are initialized and **Internal Flash ROM rewrite program via USB CDC** is executed.



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

Judgment for starting up the user application
Startup user application

Figure 7-11 main() Function



```
void usb_main(void)
{
    flash_err_t flash_err = FLASH_SUCCESS;
    g_usb_response_record.record_type = REC_TYPE_RESPONSE;
    g_usb_response_record.record_len = NR_RESPONSE_BYTES;

    usb_driver_init();
    usb_cdc_init();
    fl_write_data_init();

    /*FCU firm ready*/
    flash_err = R_FLASH_Open();
    if (FLASH_SUCCESS != flash_err)
    {
          g_fw_sequence = FLASH_SEQ_FLASH_NG;
    }

    while (1)
    {
        if (0 != g_usb_confnum)
        {
             if (CDC_BLK_OUT_OK == fu_cdc_read())
             {
                   fu_main();
             }
        }
    }
```

Flash API Initialization
USB connecting checking
USB data receive checking
Firmware updater processing

Figure 7-12 usb_main()関数

7.5.6 Jump to user application

The processing to jump to the user program is performed in the jump_to_userapp( ) function. Refer to section 5.2 **UserApp Header Area (user application header)** for details on specifying the start address of the user program jump destination.

8. File Transfer Application (RX USB Firmware Updater) Explanation

This section explains how the file transfer application performs on the host PC.

8.1 Development Environment

The file transfer application is configured with the following environment:

OS: Windows 8.1, Windows 10

Development language: Visual Studio 2017

8.2 Operation Overview

The file transfer application transitions to the direct re-write processing when it receives the name (or option) of a target re-write file name as an argument at startup. If a file has not been specified, the setting dialog is displayed.

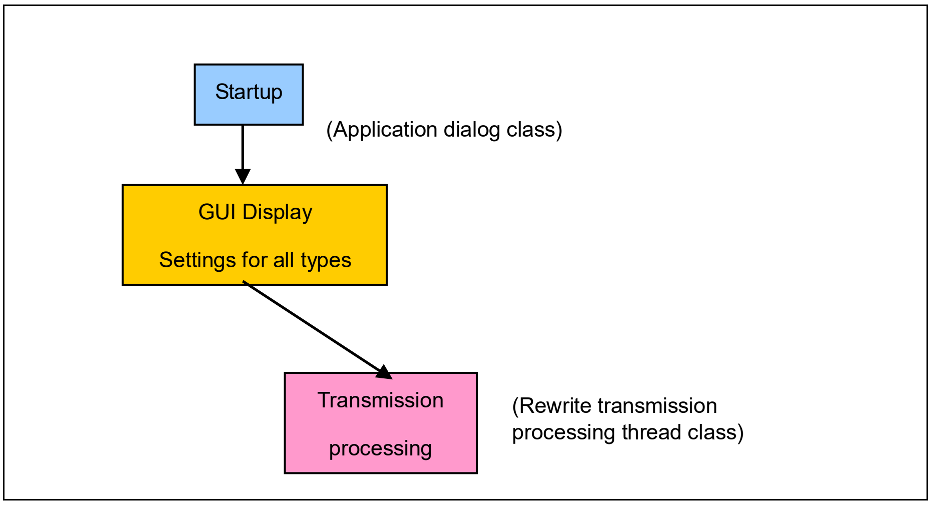


Figure 8-1 File Transfer Application Operation Overview

8.3 File Configuration

The following lists the file transfer application files (only key files are listed).

Table 8-1 File Transfer Application Files

File Name	Description
FlashSelfRewriteGUI.sin	Solution file
FlashSelfRewriteGUI.rc	Resource file
FlashSelfRewriteGUI.cpp	Application class processing file
FlashSelfRewriteGUI.h	Application class definition file
FlashSelfRewriteGUIDlg.cpp	Application dialog class processing file
FlashSelfRewriteGUIDlg.h	Application dialog class definition file
CommandThread.cpp	Rewrite transmission processing thread class processing file
CommandThread.h	Rewrite transmission processing thread class definition file
CommonProc.cpp	Common processing class processing file
CommonProc.h	Common processing class definition file
SerialPort.cpp	Serial COM port transmission class processing file
SerialPort.h	Serial COM port transmission class definition file
Resource.h	Resource header file
UsbfUpdater.ini	Application operation setting file

8.3.1 Application Class (FlashSelfRewriteGUI)

This processing checks the arguments (options) at the initial startup, then calls the dialog class.

The following lists the application startup options.

Table 8-2 Application Startup Options

Option	Description
/S nnnnnn	Specify the write start address as a hexadecimal number
/C nn	Specify the connection COM port number
Filename	Target rewrite file path

8.3.2 Application Dialog Class (FlashSelfRewriteGUIDlg)

This processing displays the rewrite specification dialog screen (refer to section 4 **Execute Internal Flash ROM rewrite program via USB CDC** for details). This screen allows the user to specify operation mode, rewrite address, rewrite file, and connection COM port. In addition, if these items are already specified when the screen is displayed, the function reads the application operation setting file and reflects the settings as default values.

Click the Update button to call the rewrite transmission processing thread class.

Added member variables are shown below.

Table 8-3 Application Dialog Class Member Variables

Member Variable		Description
Type	Name	
Int	m_nCOM	Number of COM port to be connected
TCHAR	m_tcAppDir[_MAX_PATH]	Application execution directory
CString	m_strCurTargetSeries	Current target series
CString	m_strCurTarget	Current target name
CString	m_strCurDevice	Current device
CStringArray	m_arDeviceSeries	Device series list
CStringArray	m_arDeviceVal	Device list
CStringArray	m_arDeviceText	Device name list
Int	m_nDevSize	Current device ROM size
CWinThrread*	m_pCommandThread	Thread class pointer
BOOL	m_bExistThread	Thread operation status
BOOL	m_bStartUp	Display initial startup
DWORD	m_dwROMStartAddress	ROM area start address
DWORD	m_dwROMEndAddress	ROM area end address
DWORD	m_dwEnROMStartAddress	ROM P/E access enabled start address
DWORD	m_dwEnROMEndAddress	ROM P/E access enabled end address
COleDateTime	m_dtStart	Rewrite processing start date and time
COleDateTime	m_dtEnd	Rewrite processing end date and time

Member functions are described below.

Table 8-4 Read_DeviceInfo Function

Function name		Read_DeviceInfo
Description		bool Read_DeviceInfo (void)
Function		Get information from application operation setting file
I/O	**Input**	None
	Output	TRUE(SUCCESS) / FALSE(FAILURE)

Table 8-5 Write_DeviceInfo Function

Function Name		Write_DeviceInfo
Description Format		bool Write_DeviceInfo (void)
Function		Update application operation setting file
I/O	Input	None
	Output	TRUE(success)/FALSE(fail)

Table 8-6 Update_Message Function

Function Name		Update_Message
Description Format		void Update_Message (LPCTSTR)
Function		Display message in message display column
I/O	Input	Message character string pointer
	Output	None

Table 8-7 Initialize_Device Function

Function Name		Initialize_Device
Description Format		void Initialize_Device (void)
Function		Initialization processing
I/O	Input	None
	Output	None

Table 8-8 DeviceListRefresh Function

Function Name		DeviceListRefresh
Description Format		void DeviceListRefresh (void)
Function		Create Device list
I/O	Input	None
	Output	None

Table 8-9 DeviceInfoRefresh Function

Function Name		DeviceInfoRefresh
Description Format		void DeviceInfoRefresh (void)
Function		Update device combo box
I/O	Input	None
	Output	None

Table 8-10 AppStatus Function

Function Name		AppStatus
Description Format		void AppStatus(bool stu)
Function		Set status at rewrite operation
I/O	Input	stu: TRUE (enable screen controls) FALSE (disable screen controls)
	Output	None

8.3.3 Rewrite Transmission Processing Thread Class (CommandThread)

This processing uses the serial COM port transmission class to send and receive the specified file based on the interface specifications when connected to the target evaluation board. If the file is a HEX file, analysis is also performed.

Added member variables are shown below (files listed under application dialog class are not repeated here).

Table 8-11 Rewrite Transmission Processing Thread Class Member Variables

Member Variable		Description
Type	Member Name	
CDialog*	m_pAppDlg	Dialog class of call origin pointer
CString	m_strAppDir	Directory in application
BOOL*	m_pbExistThread	Thread operation status pointer
CSerialPort	m_Serial	Serial COM port transmission class
int	m_nCOM	Connection COM port number
CString	m_strFileName	Target file path
EnMode	m_enMode	Rewrite mode
DWORD	m_dwStartAddress	Rewrite start address
DWORD	m_dwROMStartAddress	ROM start address
DWORD	m_dwROMEndAddress	ROM end address

Added member functions are listed below.

Table 8-12 Cal_CheckSum Function

Function Name		Cal_CheckSum
Description Format		BYTE Cal_CheckSum(LPBYTE bytes, LONG size)
Function		Calculate check sum
I/O	**Input**	Bytes: data string pointer Size: data string length
	Output	Calculated check sum value

Table 8-13 Change_strHex2Bibary Function

Function Name		Change_strHex2Binary
Description Format		VOID Change_strHex2Binary (LPCSTR strHex, LPBYTE pbytes, LONG size)
Function		Convert string displayed in hex to binary data string
I/O	**Input**	strHex: pointer to character string displayed in hexidecimal notation pbyte: data string start pointer size: number of conversion data
	Output	None

Table 8-14 Upsets_DWORD Function

Function Name		Upsets_DWORD
Description Format		DWORD Upsets_DWORD(DWORD dwVal)
Function		Invert DWORD type values by byte (ex.) 0xaabbccdd -> 0xddccbbaa
I/O	**Input**	dwVal: value of DWROD to be inverted
	Output	Inverted value

Table 8-15 SET_StartRecord Function

Function Name		SET_StartRecord
Description Format		VOID SET_StartRecord (LPVOID lpRecord)
Function		Creates rewrite start record
I/O	**Input**	lpRecord: record storage pointer
	Output	None

Table 8-16 SET_EndRecord Function

Function Name		SET_EndRecord
Description Format		VOID SET_EndRecord (LPVOID lpRecord)
Function		Creates rewrite end record
I/O	**Input**	lpRecord: record storage pointer
	Output	None

8.3.4 Common Processing Class (CommonProc)

Processes that are shared in the File Transfer Application are defined in this section. Added member functions are described below.

Table 8-17 GetAppDir Function

Function Name		GetAppDir
Description Format		static VOID GetAppDir(LPTSTR path, int sw = 0)
Function		Get the application execution address
I/O	**Input**	Path: target character string pointer sw: 0 Get path as is 1 Get shortened path
	Output	None

Table 8-18 Change_Hex2Val Function

Function Name		Change_Hex2Val
Description Format		static DWORD Change_Hex2Val(LPCSTR pHex)
Function		Convert character string displayed in 1 byte (2 hex digits) to a numerical value
I/O	**Input**	pHex: pointer for character string displayed in 2 hex digits
	Output	Converted value

Table 8-19 IsNumeric Function

Function Name		IsNumeric
Description Format		static BOOL IsNumeric(LPCTSTR lpNum, LONG size, int type)
Function		Numerical value check processing
I/O	**Input**	lpNum: pointer of character string expressed in numerical value size: number of digits of checked value type : 10 Check as a decimal number 16 Check as a hex number
	Output	TRUE (indicates a numerical value) /FALSE (indicates a non-numerical value)

Table 8-20 IsExistFile Function

Function Name		IsExistFile
Description Format		static BOOL IsExistFile(LPCTSTR lpszFileName, BOOL bDirectory = FALSE)
Function		Check for existing file
I/O	Input	lpszFileName: file path to be confirmed bDirectory: FALSE (check file) TRUE (check directory)
	Output	TRUE (file exists) / FALSE (no file)

8.3.5 Serial COM Port Transmission Class (SerialPort)

This class is used for serial transmission via the COM port.

Added member variables are list below.

Table 8-21 Serial COM Port Transmission Class Member Variables

Member Variable		Description
Type	Member Name	
HANDLE	m_hCom	Handle that is received when connection is made
DCB	m_Dcb	Device control block structure
COMMTIMEOUTS	m_TimeoutSts	Time out setting structure
INT	m_nCOM	Number of port to be connected

Member functions are described below.

Table 8-22 Port_Open Function

Function Name		Port_Open
Description Format		LONG Port_Open(INT com)
Function		Connect to specified COM port.
I/O	**Input**	Com: COM port number
	Output	0 Successful connection −1 Failed connection

Table 8-23 Port_Close Function

Function Name		Port_Close
Description Format		VOID Port_Close(VOID)
Function		Disconnect the connected port.
I/O	**Input**	None
	Output	None

Table 8-24 Port_Write Function

Function Name		Port_Write
Description Format		LONG Port_Write(LPCVOID buf, LONG cnt)
Function		Transmit data in serial transmission
I/O	Input	Buf: transmit data string pointer Cnt: transmit data length (bytes)
	Output	Number of transmitted bytes, “-1” indicates transmit failure.

Table 8-25 Port_Read Function

Function Name		Port_Read
Description Format		LONG Port_Read(LPVOID buf, LONG cnt)
Function		Receive data in serial transmission.
I/O	Input	Buf: pointer of data string that stores receive data cnt: receive data length (bytes)
	Output	Number of received bytes. “-1” indicated receive failure.

Table 8-26 Get_PortNumber Function

Function Name		Get_PortNumber
Description Format		INT Get_PortNumber(VOID)
Function		Get number of connected port.
I/O	Input	None
	Output	Number of currently connected port

Table 8-27 AutoScanCom Function

Function Name		AutoScanCom
Description Format		INT AutoScanCom (LPCTSTR pszService, LPCTSTR pszInterface, INT nNo = 0)
Function		Detect connectable COM ports.
I/O	Input	pszService: Name of service run by COM port pszInterface: interface name nNo: search beyond this number
	Output	Detected COM port number. If not found, return 0.

8.3.6 Application Operation Setting File (UsbfUpdater.ini)

The application operation setting file is ini file format and retains setting values and device information. Please keep this file in the folder that stores the exe file. Note that the application will not run normally without the ini file.

Definitions for the ini file are provided below.

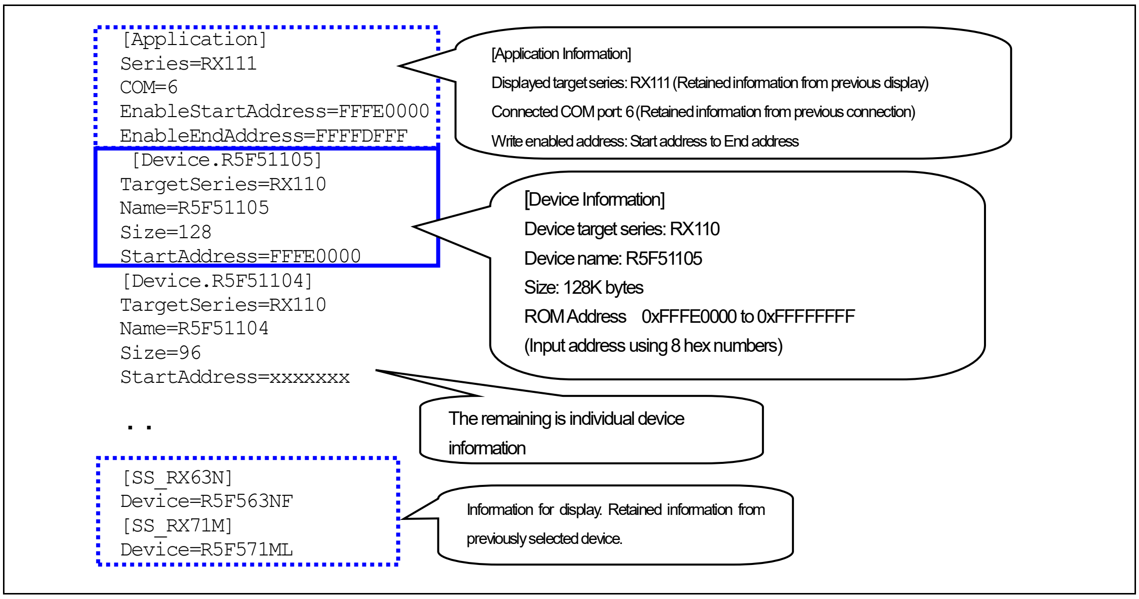
Table 8-28 Application Operation Setting File Description (sections)

Section	Description
Application	Display values currently set in the application. This is information to be written by the application.
SS_xxx	Retain previously displayed device information. This is information to be written by the application.
Device. XXXXXXXX	Display device information (multiple settings possible), This is information that can be added by user.

Table 8-29 Application Operation Setting File Contents

Section	Key	Value	Description
Application	Series	XXX	Series of specified target
	COM	1 to 20	The number of the COM port that is currently or will be connected Note: Can be set but not used in OS versions later than Windows 10.
	EnableStartAddress	FFFFFFFF	Write enabled start address
	EnableEndAddress	FFFFFFFF	Write enabled end address
SS_XXX	Device	XXX	Device specified by target
Device. XXX	TargetSeries	XXX	Series of this device
	Name	XXX	Name of this device
	Size	1 to 999	ROM size (Kbytes) of this device
	StartAddress	FFFFFFFF	ROM start address for this device

Items other than the device information are stored as display information and will be updated automatically when the GUI software is closed.



```
[Application]
Series=RX111
COM=6
EnableStartAddress=FFFE0000
EnableEndAddress=FFFFDFFF
 [Device.R5F51105]
TargetSeries=RX110
Name=R5F51105
Size=128
StartAddress=FFFE0000
[Device.R5F51104]
TargetSeries=RX110
Name=R5F51104
Size=96
StartAddress=xxxxxxx

 . .

[SS_RX63N]
Device=R5F563NF
[SS_RX71M]
Device=R5F571ML
```

[Application Information]
Displayed target series: RX111 (Retained information from previous display)
Connected COM port: 6 (Retained information from previous connection)
Write enabled address: Start address to End address

[Device Information]
Device target series: RX110
Device name: R5F51105
Size: 128K bytes
ROM Address 0xFFFE0000 to 0xFFFFFFFF
(Input address using 8 hex numbers)

The remaining is individual device information

Information for display. Retained information from previously selected device.

Figure 8-2 Application Operation Setting ini File

8.4 Application Messages

The following lists the messages displayed by the application in the message column and the timing in which they are displayed.

Table 8-30 Application Messages

Message	Display Timing
Start upload file.	At start of rewrite processing
Now erasing	Easing Flash ROM
Now writing	Writing Flash ROM
Now copying	Copying Flash ROM (Using the backup funcion only)
Please input file.	At rewrite processing when specified file is not specified. Also when specified file is not found.
Please set the address correctly.	When address is not specified correctly
Please set COM port.	When COM port is not specified correctly
ERR: file open error.	Failure in opening file
ERR: file format error.	When a file in other than Motorola S format or Intel HEX format is specified
ERR: Unable to connect to the COM port n.	Failed connection to COM port n
ERR: Flash ROM Initialization error	Flash ROM initializing error
ERR: Data transmission error.	Failed data transmission
ERR: Data reception error.	Failed data reception (failed for 3 retries)
ERR: Verify error	A verification error occurred.
ERR: Copying of Flash ROM rewrite program failed.	The copying of Internal FlashROM rewrite program is failure. (Using Dual mode only)
ERR: Unused area writing error	Unused area writing error (Using the backup function only)
ERR: Option-Setting Memory writing error	Option-Setting Memory writing error occured.
ERR: Writing process stop.	Received NAK (error) in response record from board side
ERR: Write Enable Area Address is ROM area over, or illegal value.	The specified P/E Access Enable Area exceeds ROM area or an illegal value (only when Use P/E Access Enable is selected).
ERR: Address is ROM area over. Process stop.	Programming address exceeds ROM area
ERR: file size error.	When file size check shows data size exceeds ROM area
ERR: Security code of Updater and User program do not match.	Security code of Firmware Updater and User program do not match.
ERR: Get ROM Address Error. <Device: xxxx >	When the ini file ROM information is incorrect
ERR: Get ROM Address Error. Update process stop.	When a write operation is executed and the ROM information read in the ini file is incorrect

9. Data Transmission Specification

9.1 Rewrite Transmission Interface Specification

This section describes transmission between the PC which the file transfer application works on and the evaluation board.

9.1.1 Transmission data configuration

The PC transmits the start record and end record. The write data is sent to the Flash memory in data record format.

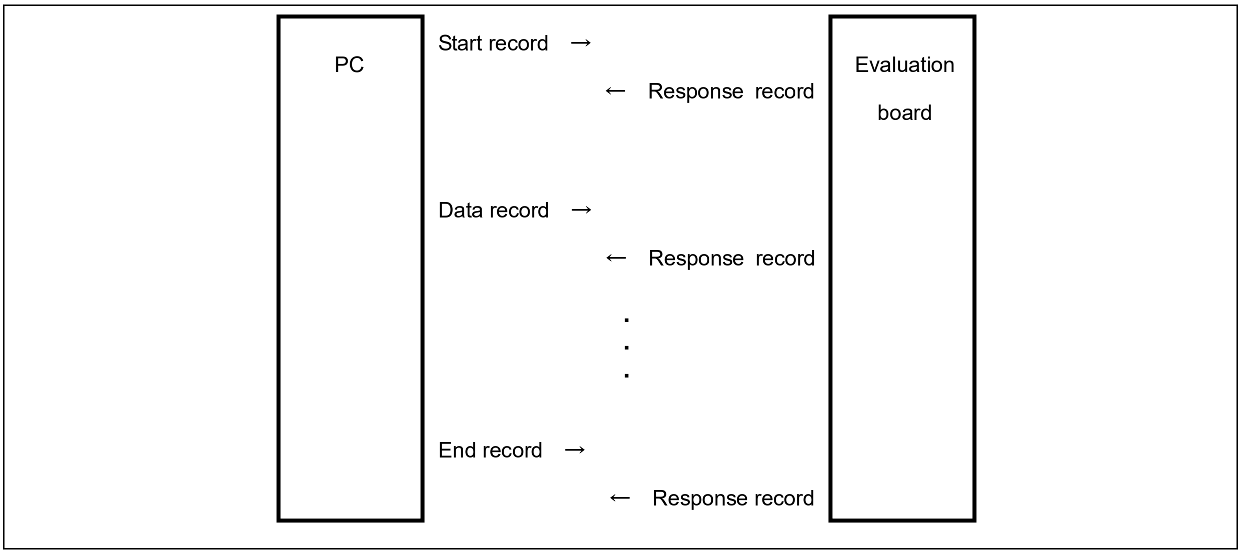


Figure 9-1 Transmission Data Sequence

9.1.2 PC-side transmission data

The PC side sends the start record, data record, and end record.

Each record is transmitted one at a time and the next record is not sent until a response for the previously sent record is received.

(1). Start record

The start record is the first record to be transmitted when executing a rewrite: 14 bytes.

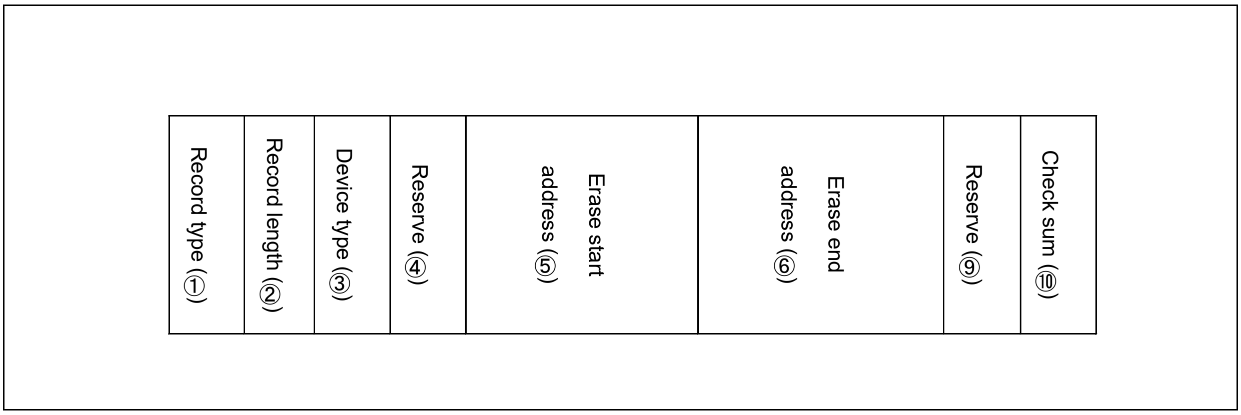


Figure 9-2 Start Record Format

① Record type: 1 byte

Record type

The start record record type is 0x00.

② Record length: 1 byte

Number of bytes after the device type

③ Device type: 1 byte

Device type (currently unused, therefore fixed as 0x00)

④ Reserve: 1 byte

0x00 fixed

⑤ Erase start address: 4 bytes

ROM erase start address setting. The address is a 32-bit numerical value in Little Endian format.

⑥ Erase end address: 4 bytes

ROM end address specification. The address is a 32-bit numerical value in Little Endian format.

⑦ Reserve:1 byte

0x00 fixed

⑧ Check sum: 1 byte

Record check sum.

Check sum of the record length, device type, and date and time.

The lower 8 bits of the complement 1 of the sum of all the bytes received.

(2). Data record

Write data record: (7+number of data) bytes (MAX 64 bytes)

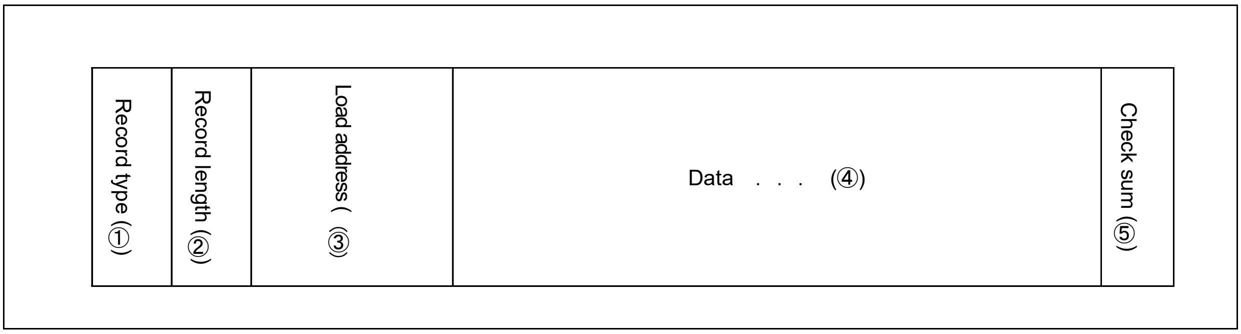


Figure 9-3 Data Record Format

① Record type: 1 byte

Record type

The data record record type is 0x0f.

① Record length: 1 byte

Number of bytes after the load address.

② Load address: 4 bytes

Flash memory address

Data is written from this address.

The load address is a 32-bit numerical value in Little Endian format.

③ Data: 1 to 57 bytes

Data to be written to the Flash memory

1 record is a maximum of 57 bytes.

④ Check sum: 1 byte

Record check sum.

Check sum of the record length and address data.

The lower 8 bits of the complement 1 of the sum of all the bytes received.

(3). End record

The end record is sent after all data is transmitted: 4 bytes.

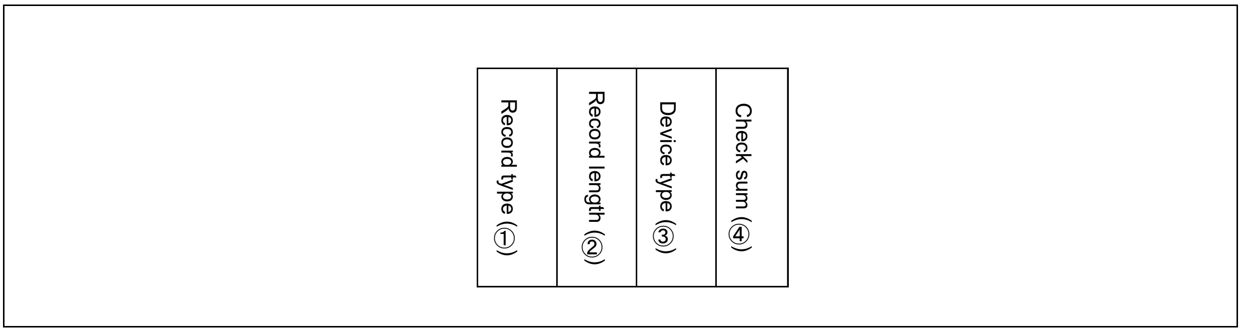


Figure 9-4 End Record Format

① Record type: 1 byte

Record type

The end record record type is 0xf0.

⑨ Record length: 1 byte

Number of bytes after the device type

⑩ Device type: 1 byte

Device type (currently unused, therefore fixed as 0x00)

⑪ Check sum: 1 byte

Record check sum.

Check sum of the record length and device type.

The lower 8 bits of the complement 1 of the sum of all the bytes received.

9.1.3 Evaluation board-side transmission data

The evaluation board sends a record in response to the record received from the PC: 5 to 8 bytes

(1). Response record

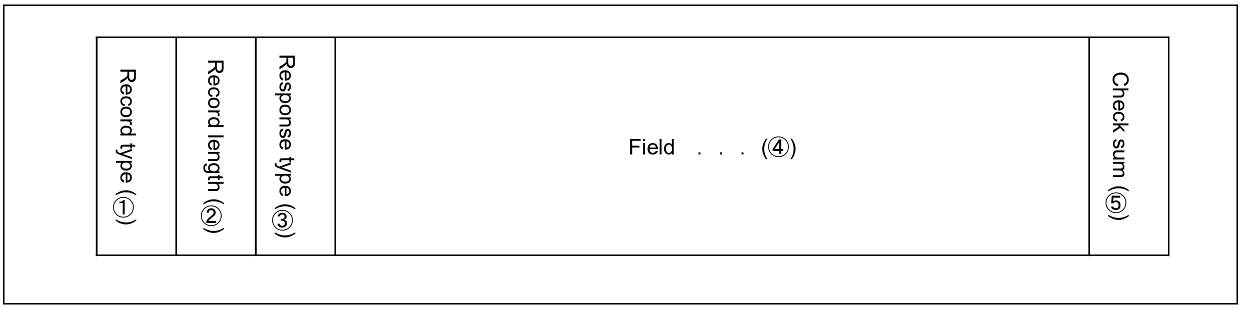


Figure 9-5 Response Record Format

① Record type: 1 byte

Record type

Type of record to which a response is being sent.

The response record type is 0xFF

② Record length: 1 byte

Number of bytes after the response type

③ Response type: 1 byte

Response type

One of the following 3 types

—0x00 : ACK

—0x0f : NAK (re-transmit/receive request)

—0xf0 : NAK (error end)

④ Field: 1 to 4 bytes

a. In the start record, the code to indicate the enable or disable of the backup function is returned.

Backup Function Enable	:	0xB0
Backup Function Disable	:	0xB1

b. In the data record or the end record, the following status code or error code is returned.

(a). Status Code

Flash ROM erasing	:	0x01
Flash ROM writing	:	0x03

(b). Error Code

Flash ROM initialization error	:	0xE1
Security code unmatching error	:	0xE2
Flash ROM erasing error	:	0xE3
Parameter error	:	0xE4
Verify error	:	0xE5
Option-Setting memory writing error	:	0xE6
Copying of Internal FlashROM rewrite program to the update target are is failure (Using dual mode only)	:	0xE7
Unused writing error (Using backup function only)	:	0xE8

If not an error, this driver returns the following status code.

⑫ Check sum: 1 byte

Record check sum.

Check sum of the record length, response type, and field.

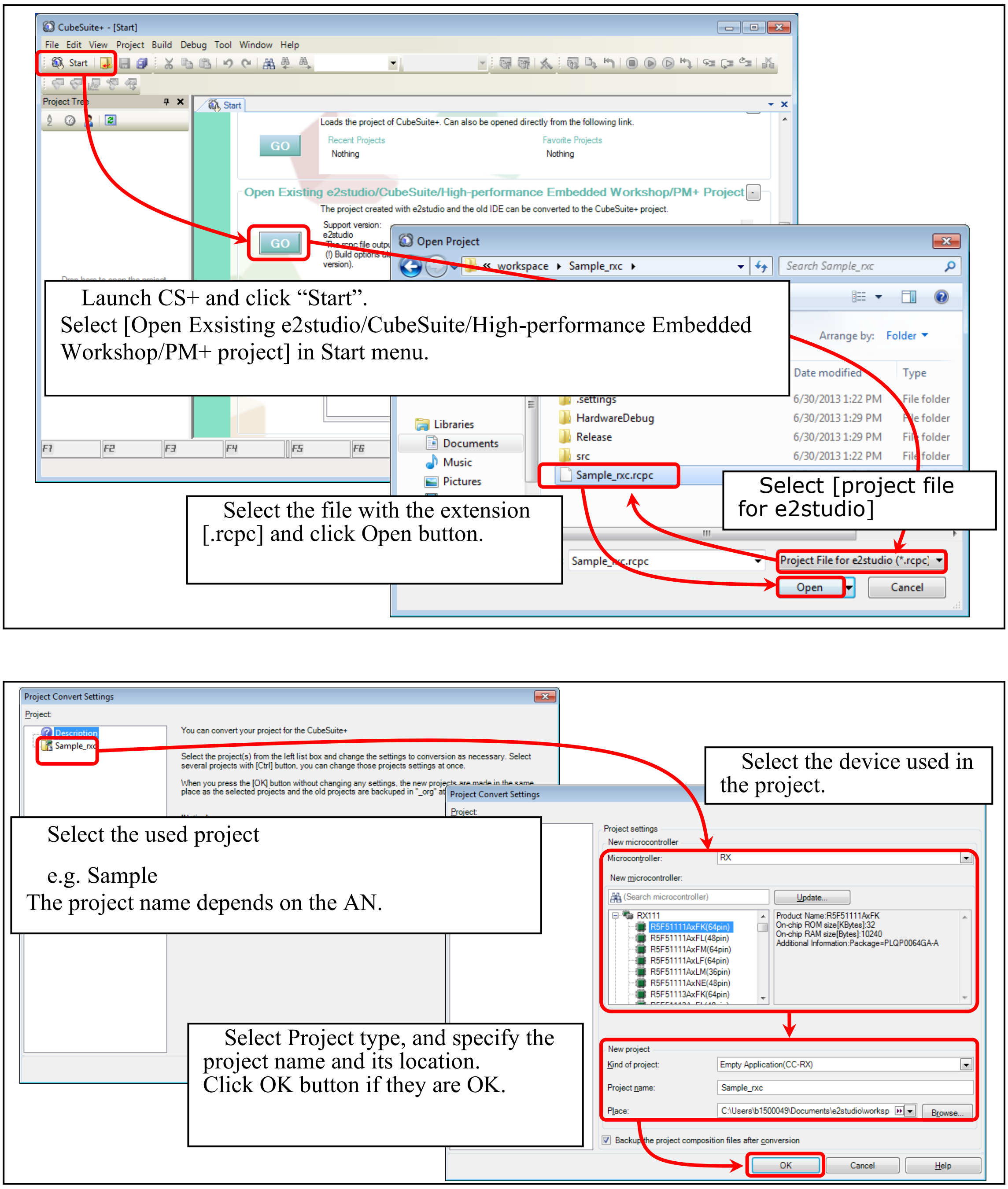
The lower 8 bits of the complement 1 of the sum of all the bytes received.

10. Using the e^2 studio project with CS+

This package contains a project only for e^2 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.



Launch CS+ and click "Start".
Select [Open Exsisting e2studio/CubeSuite/High-performance Embedded Workshop/PM+ project] in Start menu.

Select the file with the extension [.rcpc] and click Open button.

Select [project file for e2studio]

Loads the project of CubeSuite+. Can also be opened directly from the following link.

Open Existing e2studio/CubeSuite/High-performance Embedded Workshop/PM+ Project

The project created with e2studio and the old IDE can be converted to the CubeSuite+ project.

Select the used project

e.g. Sample
The project name depends on the AN.

Select the device used in the project.

Select Project type, and specify the project name and its location.
Click OK button if they are OK.

You can convert your project for the CubeSuite+

Select the project(s) from the left list box and change the settings to conversion as necessary. Select several projects with [Ctrl] button, you can change those projects settings at once.

Microcontroller: RX

Product Name:R5F51111AxFK
On-chip ROM size[KBytes]:32
On-chip RAM size[Bytes]:10240
Additional Information:Package=PLQP0064GA-A

Kind of project: Empty Application(CC-RX)

Project name: Sample_rxc

Place: C:\Users\b1500049\Documents\e2studio\worksp

☑ Backup the project composition files after conversion

Figure 10-1 Using the e^2 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	Jun 30, 2016	-	First edition issued.
1.01	Jun 30, 2017	-	RX65N and RX651 are added in Target Device.
1.02	Sep 30, 2017	-	1. Support for RX65N/RX651-2M 2. Support for dual mode. 3. Support for writing and verification
1.03	Feb 16, 2018	-	Support the backup function
1.04	Apr 16, 2019	-	RX66T and RX72T are added in Target Device.
1.05	Mar 1, 2020	-	RX72M, RX72N, RX66N and RX23W are added in Target Device.
1.06	Mar 1, 2021	-	RX671 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/.