RX100/RX200 Series

Updating Firmware Using Start-Up Program Protection and Serial Communication

Introduction

This application note describes updating the on-chip code flash memory using the start-up program protection in the RX100/RX200 Series. Serial communication is used for sample programs control and data transfer.

In this application note, “firmware program” and “firmware update program” are defined as follows:

- Firmware program: Program to be written in the user area of the code flash memory
- Firmware update program: Program to rewrite the firmware program

This application note includes the following sample programs: the firmware update program and the firmware to verify the firmware update program operation.

In this application note, the firmware update program is placed in the default area of the start-up program protection. If you plan to place the firmware update program in the alternate area, translate “default area” into “alternate area” in this document and vice versa.

Target Device

- RX230 Group and RX231 Group
- RX110 Group, RX111 Group, RX113 Group, RX130 Group, and RX140 Group

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

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)
- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- RX Family Flash Module Using Firmware Integration Technology (R01AN2184)
- RX Family SCI Multi-Mode Module Using Firmware Integration Technology (R01AN1815)
- RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)

Contents

1. Overview

1.1 About This Application Note

This application note describes the method to safely update the code flash memory using the start-up protection.

The firmware update program is placed into the default area of the start-up program protection and the constants data area. The firmware update program is controlled from the host PC through serial communication to update the code flash memory. The MCU operates in single-chip mode and the Motorola S-record data is used as data for reprogramming. The XMODEM/SUM is used as the data transfer protocol. Therefore, the terminal software on the host PC must be capable of XMODEM/SUM transfer.

Table 1.1 lists the Peripheral Functions Used and Their Applications, and Figure 1.1 shows the Operation Overview.

Table 1.1 Peripheral Functions Used and Their Applications

Peripheral Function	Application
Flash memory	Reprogramming the code flash memory
Serial communication interface	Asynchronous serial communication with the host PC

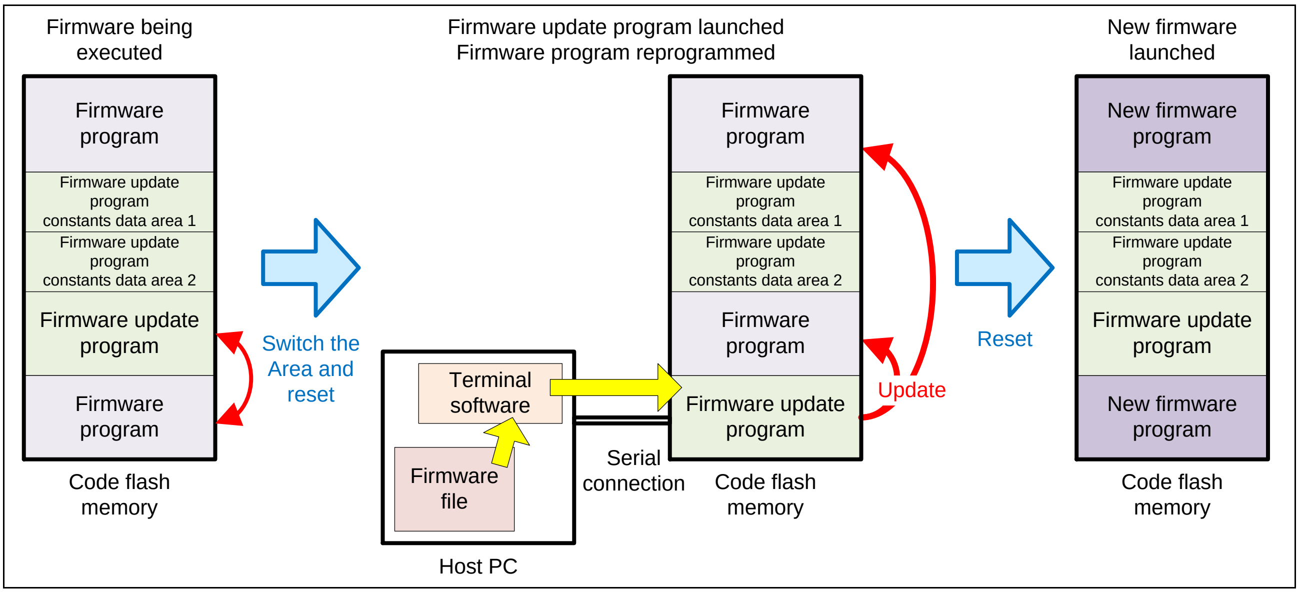

Firmware being executed
Firmware program
Firmware update program constants data area 1
Firmware update program constants data area 2
Firmware update program
Firmware program
Code flash memory
Switch the Area and reset
Firmware update program launched
Firmware program reprogrammed
Terminal software
Firmware file
Host PC
Serial connection
Firmware program
Firmware update program constants data area 1
Firmware update program constants data area 2
Firmware program
Firmware update program
Code flash memory
Update
Reset
New firmware launched
New firmware program
Firmware update program constants data area 1
Firmware update program constants data area 2
Firmware update program
New firmware program
Code flash memory

Figure 1.1 Operation Overview

The sample programs in this application note use the Firmware Integration Technology (FIT) modules to control peripheral modules. The FIT modules used in this application note are as follows:

- Board Support Package Module Using Firmware Integration Technology (BSP)
- Flash Module Using Firmware Integration Technology (Flash FIT module)
- SCI Multi-Mode Module Using Firmware Integration Technology (SCI FIT module)
- BYTEQ Module Using Firmware Integration Technology

1.2 Operation Confirmation Environment

The operation of the sample programs in this application note have been confirmed under the following conditions.

Table 1.2 Operation Confirmation Conditions

Item	Contents
MCU used	R5F51305ADFN (RX130 group) R5F51406BDFN (RX140 group) R5F52318ADFP (RX231 group)
Board used	Renesas Starter Kit for RX130 (product No.: RTK5005130C00000BE) Renesas Starter Kit for RX140 Renesas Starter Kit for RX231 (product No.: R0K505231C000BE)
Integrated development environment	Renesas Electronics e^2 studio Version 2022-04
	Renesas Electronics CS+ V.8.07.00
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V.3.04.00
	Compiler option -lang = c99
Flash programmer	Renesas Flash Programmer V.3.09.00
Emulator	E2 Lite
Endian	Little endian

1.3 Module Configuration

Figure 1.2 shows the module configuration of the sample program and Table 1.3 lists the FIT modules implemented in the sample program.

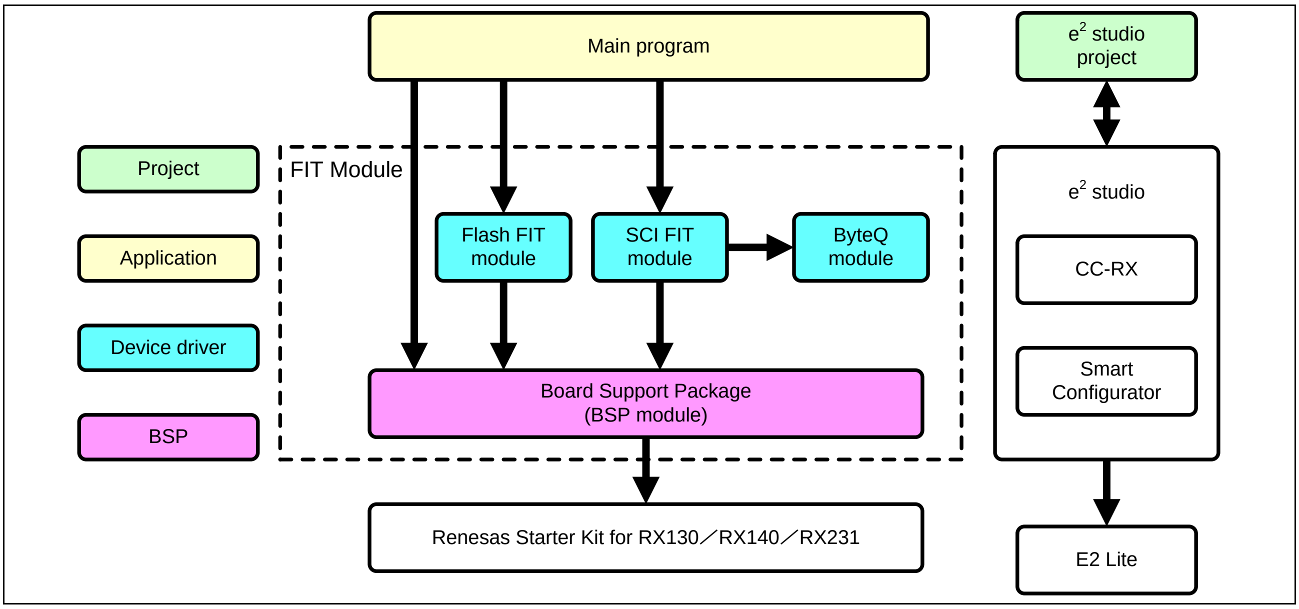


Figure 1.2 Module Configuration

Table 1.3 Module List

Category	Application Note (Document No.)	FIT Module Name
BSP	RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)	r_bsp
Device driver	RX Family Flash Module Using Firmware Integration Technology (R01AN2184)	r_flash_rx
Device driver	RX Family SCI Multi-Mode Module Using Firmware Integration Technology (R01AN1815)	r_sci_rx
Device driver	RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)	r_byteq
Application	Main program	src

1.4 File Structure

Figure 1.3 shows the file structure of this application note.

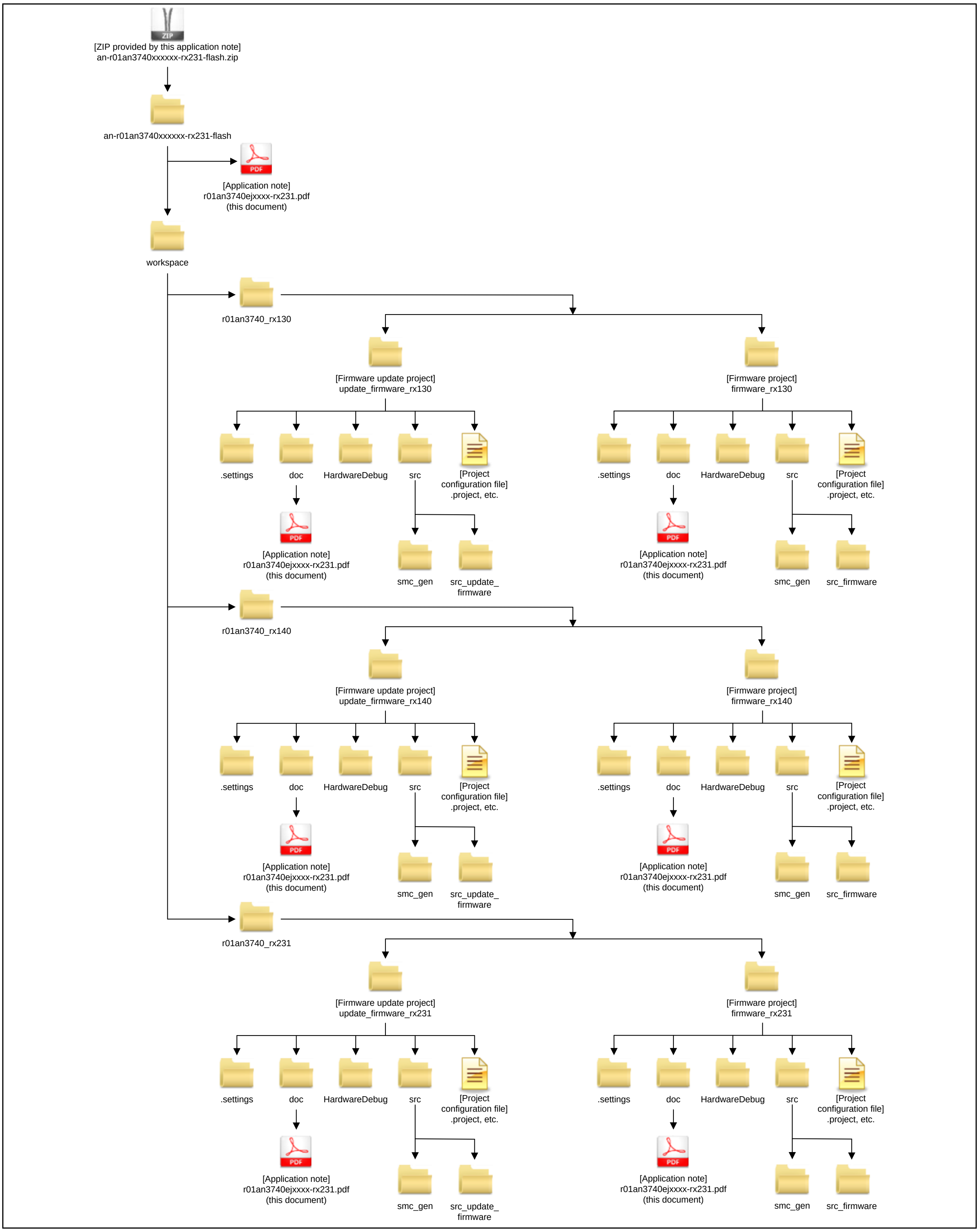


Figure 1.3 File Structure

When the ZIP file provided by this application note is unzipped, the folder is created with the same name as the ZIP, containing associated folders and files.

“Firmware update project (update_firmware_rx130/update_firmware_rx140/update_firmware_rx231)” and “Firmware project (firmware_rx130/firmware_rx140/firmware_rx231)” under the "r01an3740_rx130/r01an3740_rx140/r01an3740_rx231" folder are the projects that set up the sample programs in this application note. The operation of the application note can be confirmed by importing these projects into the workspace in the e^2 studio.

1.5 Project

This application note includes the e^2 studio projects for building and evaluating this application note. The projects have the build configuration and the debug configuration which store the build setting and the debug setting, respectively.

Table 1.4 lists the build configuration and the debug configuration that are registered in the project, and Table 1.5 shows the Target Specific Settings.

Table 1.4 Project Configuration

	Configuration Example	Description
Build configuration	HardwareDebug (Debug on hardware)	Configuration to generate a load module with debug information
Debug configuration	HardwareDebug (E2 Lite)	Performs hardware debugging via E2 Lite emulator using the load module generated with HardwareDebug (Debug on hardware)

Table 1.5 Target Specific Settings

Item	Setting
Toolchain version	V3.04.00
Debug hardware	E2 Lite (RX)
Endianness	Little-endian data
Target selection (RX130 group)	R5F51305ADFN (RX130 LFQFP 80pin)
Target selection (RX140 group)	R5F51406BDFN (RX140 LFQFP 80pin)
Target selection (RX231 group)	R5F52318ADFP (RX231 LQFP 100pin)
Renesas RTOS support	None

2. Obtaining the Development Environment

2.1 e^2 studio

Visit the following URL and download the e^2 studio.

https://www.renesas.com/en-us/products/software-tools/tools/ide/e2studio.html

This document assumes that V2022-04 or later version of e^2 studio is used. If a version earlier than V2022-04 is used, some features of e^2 studio may not be supported. Make sure to download the latest version of e^2 studio on the website.

2.2 CS+

Visit the following URL and download the CS+.

https://www.renesas.com/us/en/software-tool/cs.html

2.3 Compiler Package

Visit the following URL and download the RX Family C/C++ Compiler Package.

https://www.renesas.com/us/en/software-tool/cc-compiler-package-rx-family.html

2.4 Renesas Flash Programmer

Visit the following URL and download the Renesas Flash Programmer.

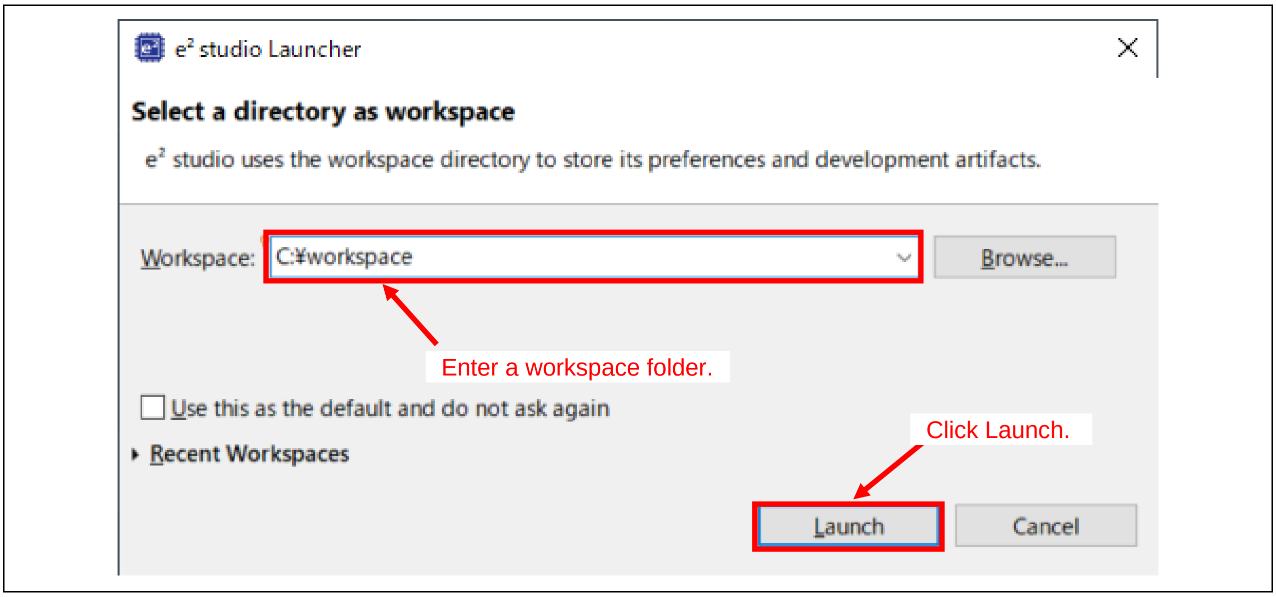
https://www.renesas.com/us/en/software-tool/renesas-flash-programmer-programming-gui.html

3. Setting Up the Project

This application note includes the projects which have been configured the environment. The procedure to import the projects using the smart browser is described here. For importing the projects into CS+, refer to 6.1. Importing a Project into CS+.

3.1 Creating a Workspace

1. Start the e^2 studio.
2. The dialog to select a workspace opens. Enter a workspace and click Launch.



e^2 studio Launcher

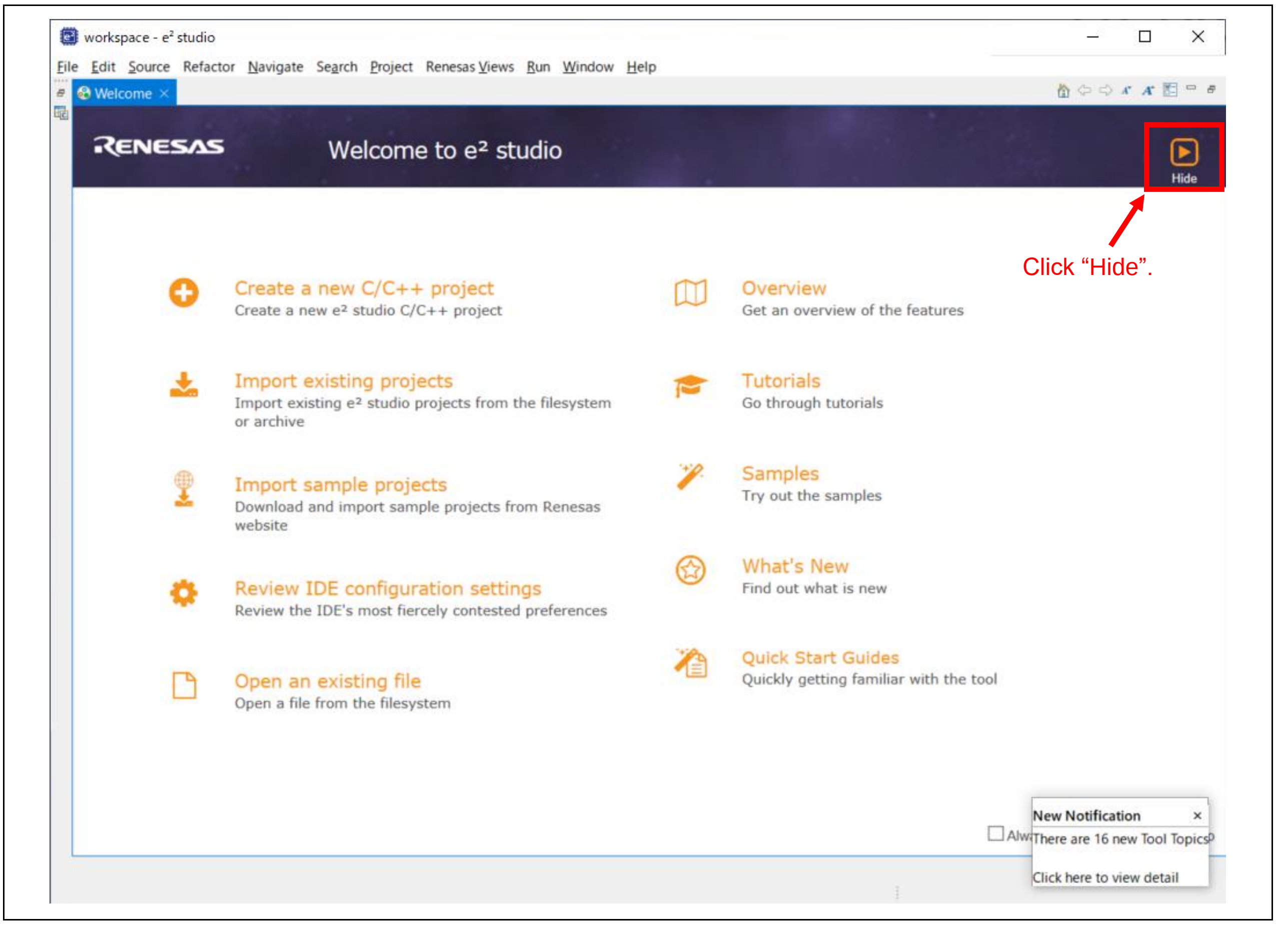
Select a directory as workspace

e^2 studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:¥workspace

Browse...
Enter a workspace folder.
☐ Use this as the default and do not ask again
Recent Workspaces
Click Launch.
Launch
Cancel

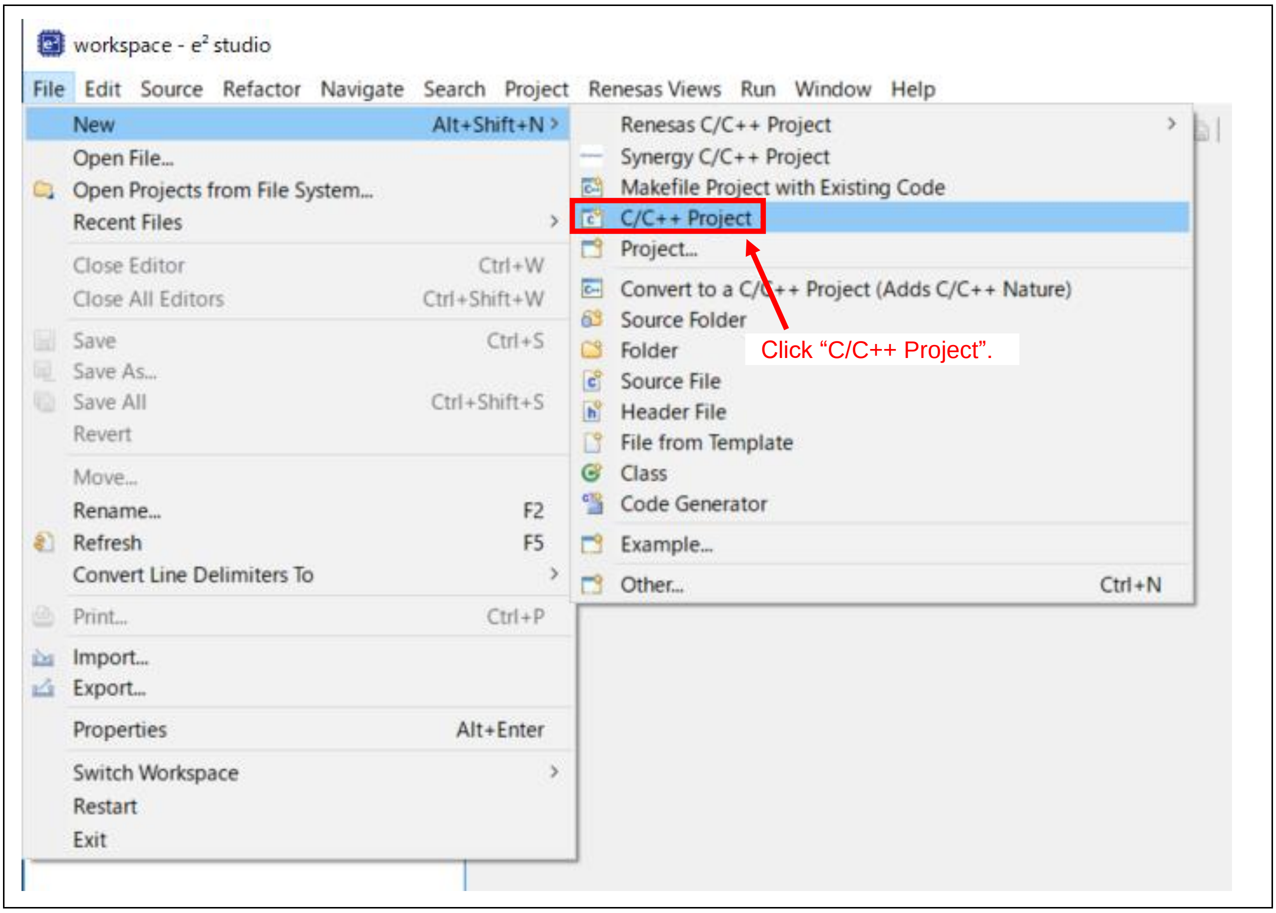
3. When the Welcome dialog opens, click “Hide”.


workspace - e² studio
File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Welcome
RENESAS
Welcome to e² studio
Hide
Click “Hide”.
Create a new C/C++ project
Create a new e² studio C/C++ project
Import existing projects
Import existing e² studio projects from the filesystem or archive
Import sample projects
Download and import sample projects from Renesas website
Review IDE configuration settings
Review the IDE's most fiercely contested preferences
Open an existing file
Open a file from the filesystem
Overview
Get an overview of the features
Tutorials
Go through tutorials
Samples
Try out the samples
What's New
Find out what is new
Quick Start Guides
Quickly getting familiar with the tool
New Notification
There are 16 new Tool Topics
Click here to view detail
Alw

3.2 Creating a Project

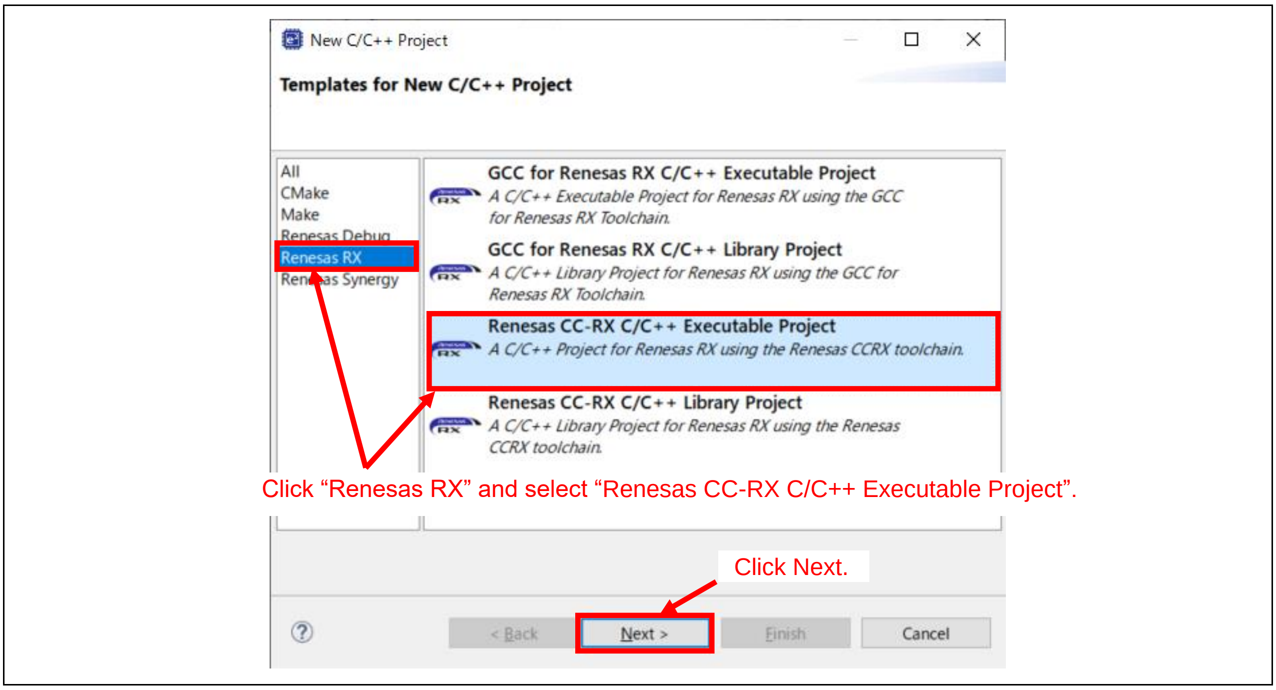
When using the Smart Brower features, a project or a file used has to be selected first. Thus, first create a project with the required MCU selected as the target device. This is a dummy project for using the Smart Browser. For configuring the imported project, refer to 3.4. Changed Setting Information.

1. Select *File* >> *New* >> *C/C++ Project*. The Wizard for creating a new project opens.



Click “C/C++ Project”.

2. Select “Renesas CC-RX C/C++ Executable Project” and click Next.



New C/C++ Project

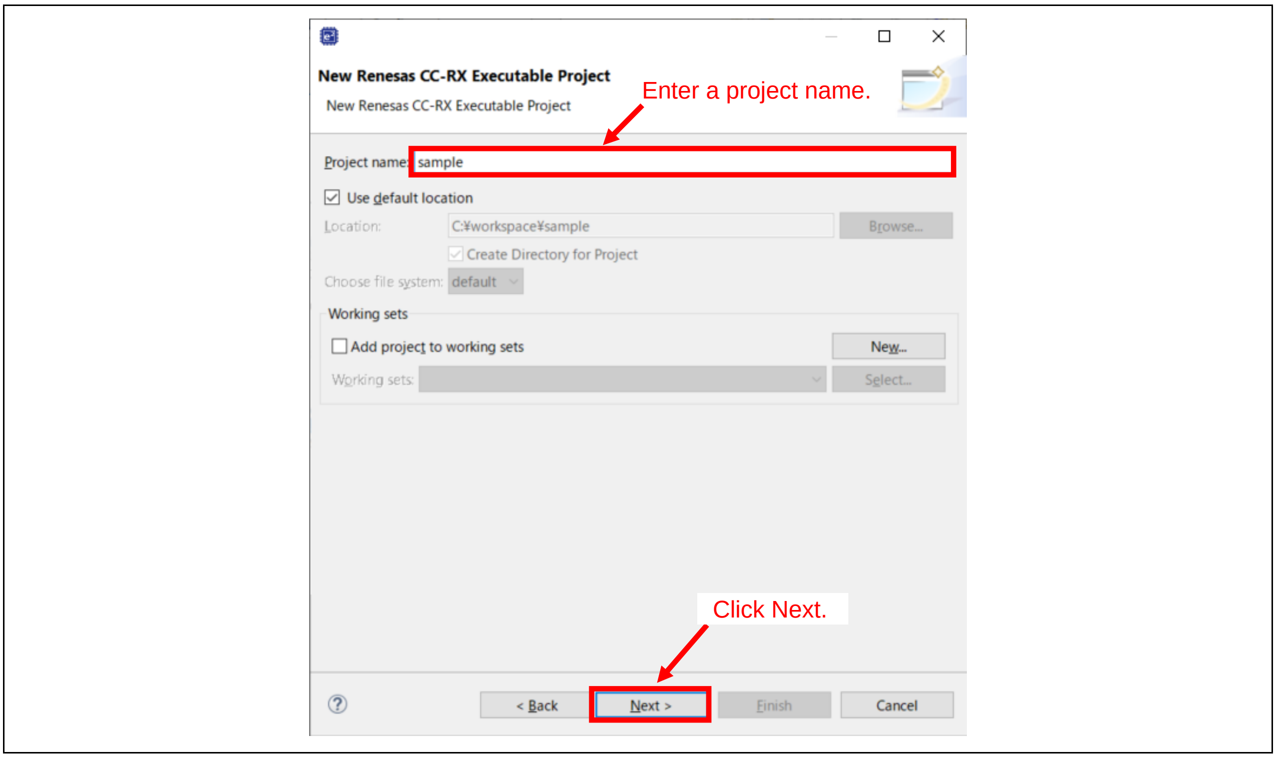
Templates for New C/C++ Project

- GCC for Renesas RX C/C++ Executable Project — *A C/C++ Executable Project for Renesas RX using the GCC for Renesas RX Toolchain.*
- GCC for Renesas RX C/C++ Library Project — *A C/C++ Library Project for Renesas RX using the GCC for Renesas RX Toolchain.*
- Renesas CC-RX C/C++ Executable Project — *A C/C++ Project for Renesas RX using the Renesas CCRX toolchain.*
- Renesas CC-RX C/C++ Library Project — *A C/C++ Library Project for Renesas RX using the Renesas CCRX toolchain.*

Click “Renesas RX” and select “Renesas CC-RX C/C++ Executable Project”.

Click Next.

Enter a project name in the Project name field and click Next.



New Renesas CC-RX Executable Project

New Renesas CC-RX Executable Project

Project name: sample

☑ Use default location

Location: C:¥workspace¥sample

☑ Create Directory for Project

Choose file system: default

Working sets

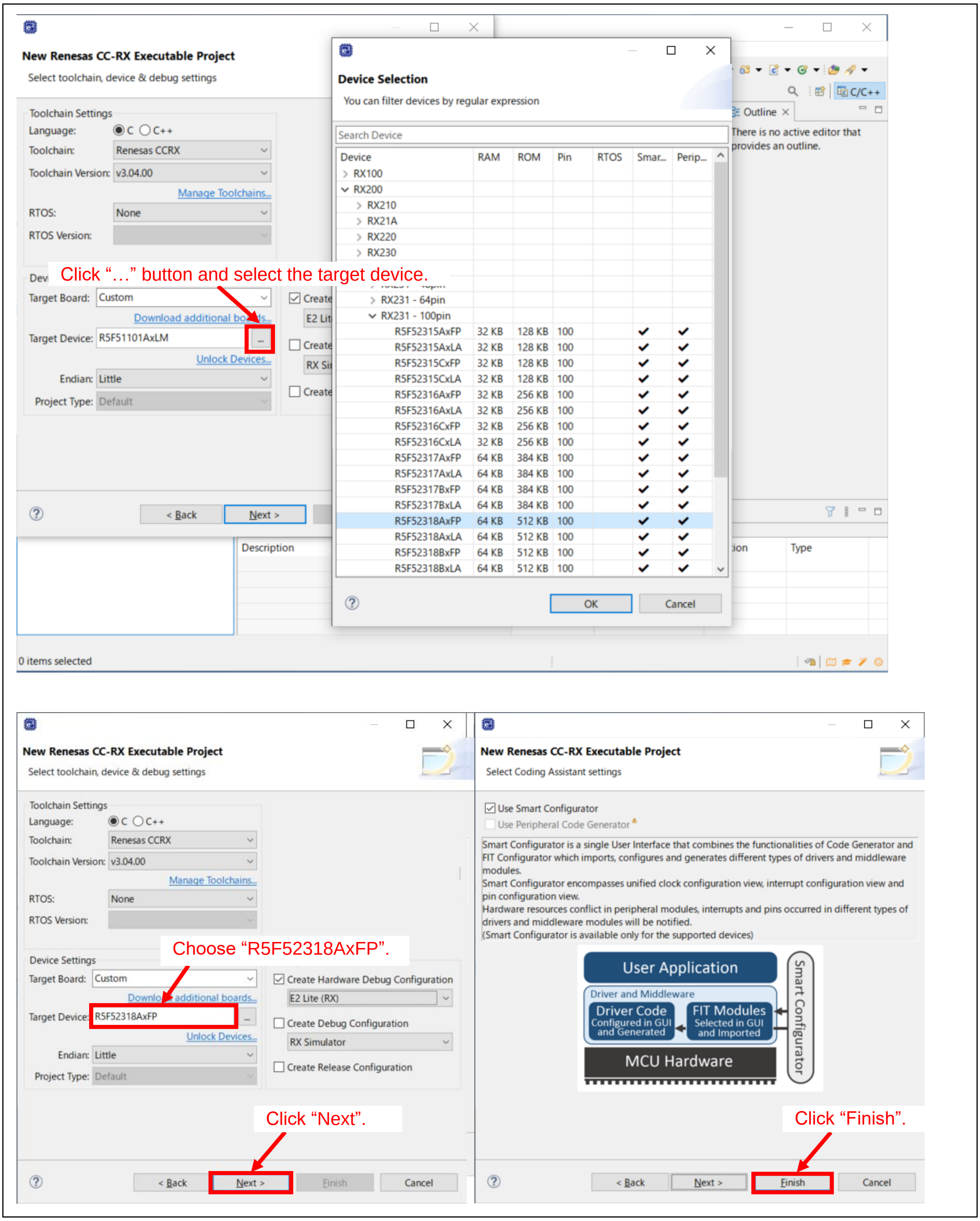
☐ Add project to working sets

Working sets:

Enter a project name.

Click Next.

3. In the Target Device field, choose "R5F51305AxFN" for the RX130 group, "R5F51406BxFN" for the RX140 group, and "R5F52318AxFP" for the RX231 group. Specify other settings as required. Click the Finish button. The following screenshots illustrate an example configuration for the RX231 group.



New Renesas CC-RX Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: ● C ○ C++

Toolchain: Renesas CCRX

Toolchain Version: v3.04.00

RTOS: None

RTOS Version:

Target Board: Custom

Target Device: R5F51101AxLM

Endian: Little

Project Type: Default

Click "…" button and select the target device.

Device Selection

You can filter devices by regular expression

Device	RAM	ROM	Pin	RTOS	Smar...	Perip...
RX100						
RX200						
RX210						
RX21A						
RX220						
RX230						
RX231 - 64pin						
RX231 - 100pin						
R5F52315AxFP	32 KB	128 KB	100		✓	✓
R5F52315AxLA	32 KB	128 KB	100		✓	✓
R5F52315CxFP	32 KB	128 KB	100		✓	✓
R5F52315CxLA	32 KB	128 KB	100		✓	✓
R5F52316AxFP	32 KB	256 KB	100		✓	✓
R5F52316AxLA	32 KB	256 KB	100		✓	✓
R5F52316CxFP	32 KB	256 KB	100		✓	✓
R5F52316CxLA	32 KB	256 KB	100		✓	✓
R5F52317AxFP	64 KB	384 KB	100		✓	✓
R5F52317AxLA	64 KB	384 KB	100		✓	✓
R5F52317BxFP	64 KB	384 KB	100		✓	✓
R5F52317BxLA	64 KB	384 KB	100		✓	✓
R5F52318AxFP	64 KB	512 KB	100		✓	✓
R5F52318AxLA	64 KB	512 KB	100		✓	✓
R5F52318BxFP	64 KB	512 KB	100		✓	✓
R5F52318BxLA	64 KB	512 KB	100		✓	✓

There is no active editor that provides an outline.

0 items selected

New Renesas CC-RX Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: ● C ○ C++

Toolchain: Renesas CCRX

Toolchain Version: v3.04.00

RTOS: None

RTOS Version:

Device Settings

Target Board: Custom

Target Device: R5F52318AxFP

Endian: Little

Project Type: Default

☑ Create Hardware Debug Configuration: E2 Lite (RX)

□ Create Debug Configuration: RX Simulator

□ Create Release Configuration

Choose "R5F52318AxFP".

Click "Next".

New Renesas CC-RX Executable Project

Select Coding Assistant settings

☑ Use Smart Configurator

□ Use Peripheral Code Generator

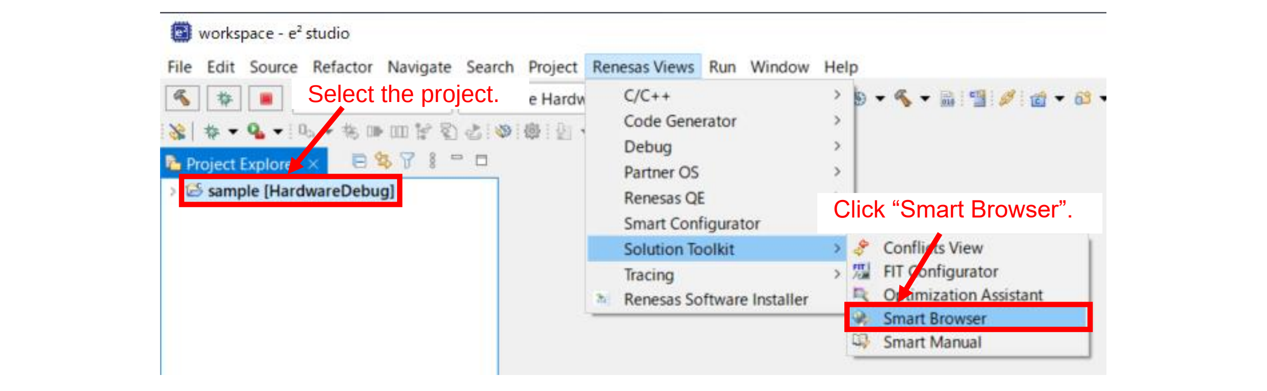
Smart Configurator is a single User Interface that combines the functionalities of Code Generator and FIT Configurator which imports, configures and generates different types of drivers and middleware modules.
Smart Configurator encompasses unified clock configuration view, interrupt configuration view and pin configuration view.
Hardware resources conflict in peripheral modules, interrupts and pins occurred in different types of drivers and middleware modules will be notified.
(Smart Configurator is available only for the supported devices)

Click "Finish".

3.3 Importing the Project

This section describes the procedure to import the sample program project into the created workspace.

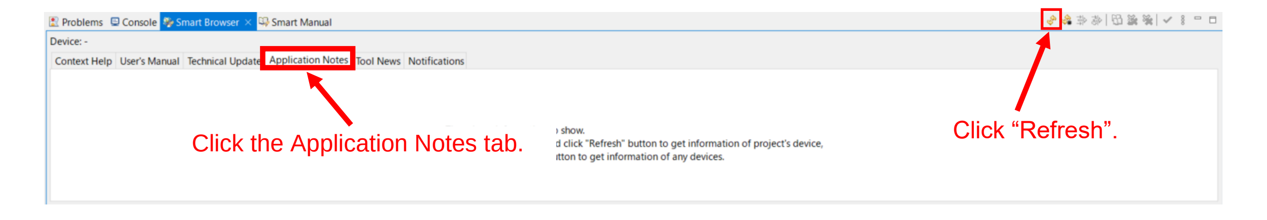
1. In the Project Explorer, select the project created in 3.2 “Creating a Project”.
2. Open the Smart Browser.



Select the project.

Click “Smart Browser”.

3. Click the Application Notes tab in the Smart Browser tab.
4. Click the Refresh icon.

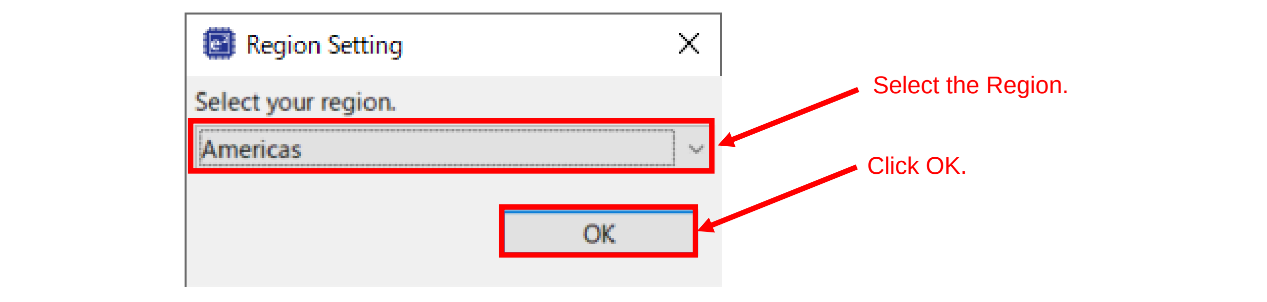

Problems Console Smart Browser Smart Manual
Device: -
Context Help User's Manual Technical Update Application Notes Tool News Notifications

› show.
d click "Refresh" button to get information of project's device,
ıtton to get information of any devices.

Click the Application Notes tab.

Click “Refresh”.

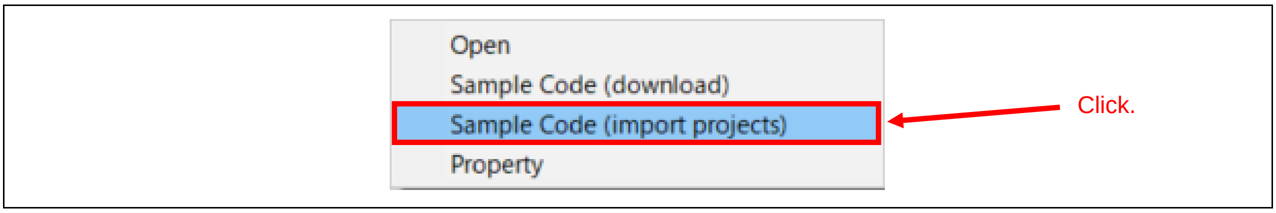
5. When the Region Setting dialog appears, select the region where you are working and click OK.



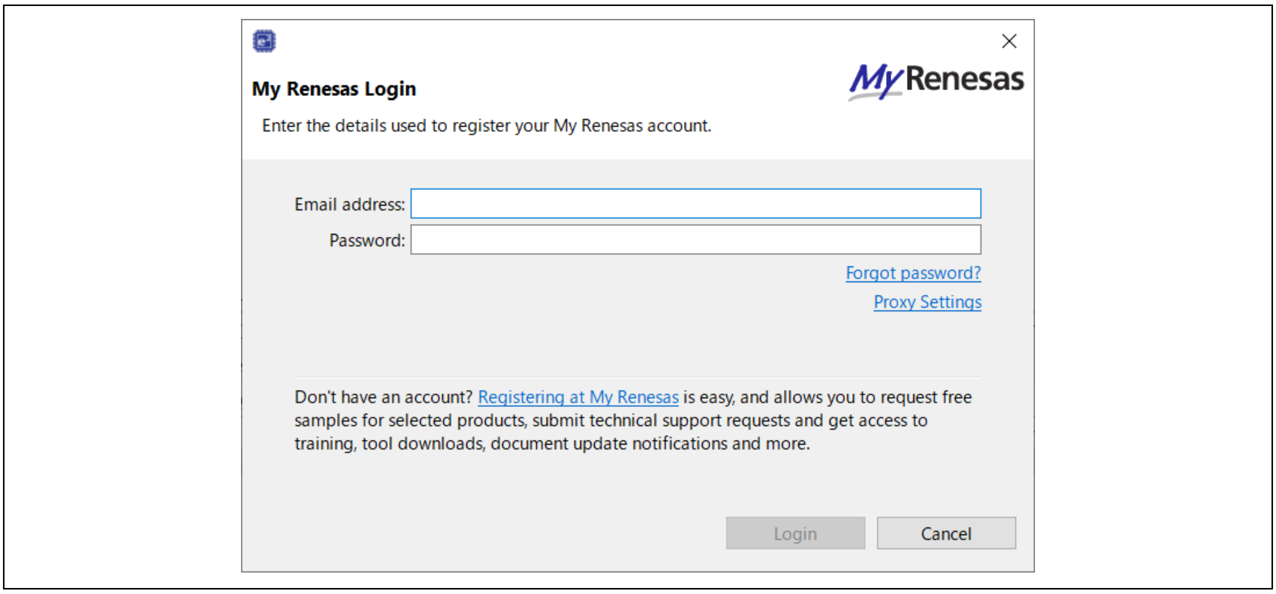
Select the Region.

Click OK.

6. Select this application note from the list and right click on it. Then select “Sample Code (import projects)” from the context menu.[*1]



Note: 1. If verification by My Renesas has never been done, “My Renesas” dialog opens before downloading the file. Enter the mail address and the password you registered on the Renesas website.



My Renesas Login

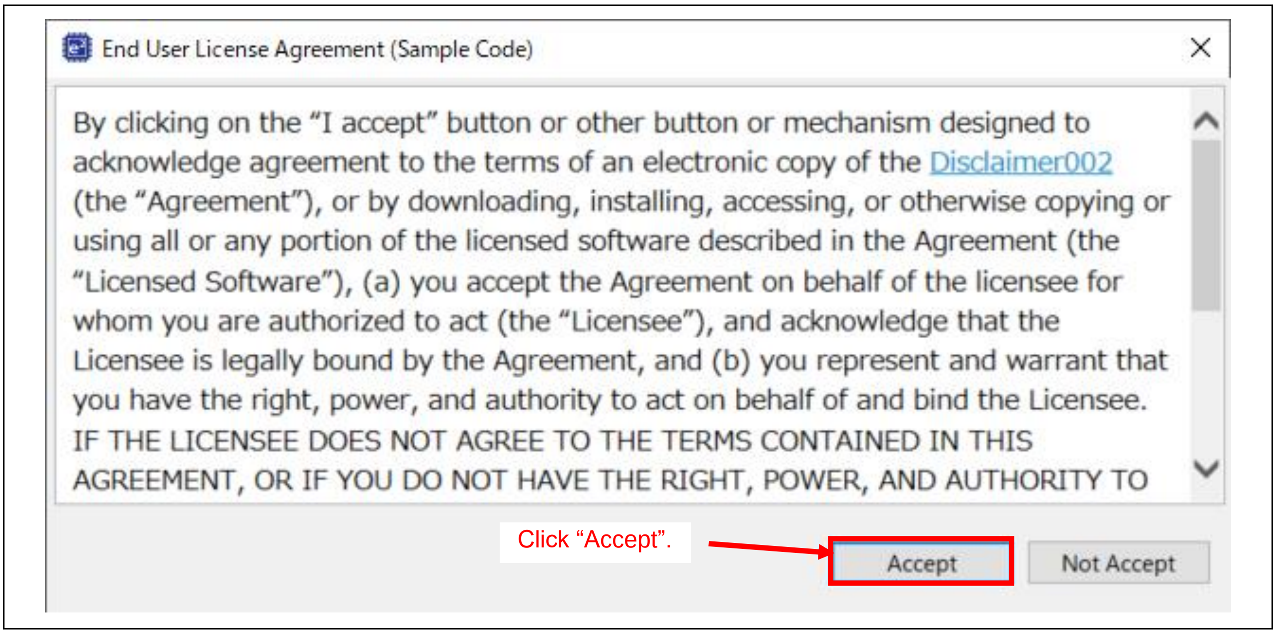
Enter the details used to register your My Renesas account.

Email address:

Password:

Don't have an account? Registering at My Renesas is easy, and allows you to request free samples for selected products, submit technical support requests and get access to training, tool downloads, document update notifications and more.

7. Click the Accept button.



End User License Agreement (Sample Code)

By clicking on the “I accept” button or other button or mechanism designed to acknowledge agreement to the terms of an electronic copy of the Disclaimer002 (the “Agreement”), or by downloading, installing, accessing, or otherwise copying or using all or any portion of the licensed software described in the Agreement (the “Licensed Software”), (a) you accept the Agreement on behalf of the licensee for whom you are authorized to act (the “Licensee”), and acknowledge that the Licensee is legally bound by the Agreement, and (b) you represent and warrant that you have the right, power, and authority to act on behalf of and bind the Licensee. IF THE LICENSEE DOES NOT AGREE TO THE TERMS CONTAINED IN THIS AGREEMENT, OR IF YOU DO NOT HAVE THE RIGHT, POWER, AND AUTHORITY TO

8. Save the application note.
9. RX130 group: Select “update_firmware_rx130” and “firmware_rx130” from the Project section and click the Finish button
   RX140 group: Select “update_firmware_rx140” and “firmware_rx140” from the Project section and click the Finish button
   RX231 group: Select “update_firmware_rx231” and “firmware_rx231” from the Project section and click the Finish button
10. After the project is imported, delete the project created for using the smart browser (“sample” in this document) as it is not necessary anymore.

3.4 Changed Setting Information

In this application note, settings in the configuration file and the project have been changed for each FIT module to set up the sample programs. The details are described in the sub-sections.

3.4.1 Configuration Option

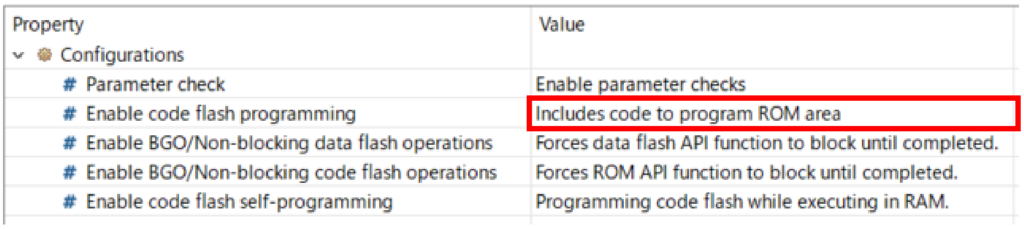
The configuration options to set up the sample programs have been changed for each FIT module.

For items of the configuration options and their settings, refer to the document in the doc folder of each FIT module.

The following describes changes in the configuration options of Smart Configurator.

(1) Modifying the Flash FIT module

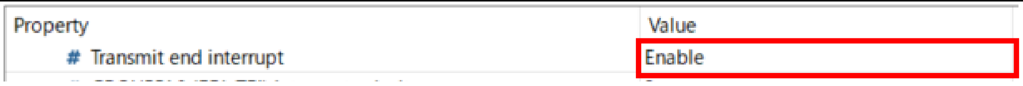
In the software component configuration screen, the following setting is changed to allow the Flash FIT module to reprogram the code flash memory.



Property	Value
⌄ Configurations	
# Parameter check	Enable parameter checks
# Enable code flash programming	Includes code to program ROM area
# Enable BGO/Non-blocking data flash operations	Forces data flash API function to block until completed.
# Enable BGO/Non-blocking code flash operations	Forces ROM API function to block until completed.
# Enable code flash self-programming	Programming code flash while executing in RAM.

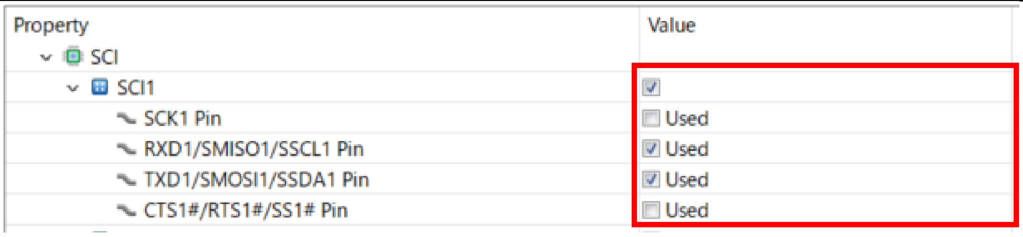
(2) Modifying the SCI FIT module

In the software component configuration screen, the transmit data empty interrupt is set to be enabled.



Property	Value
# Transmit end interrupt	Enable

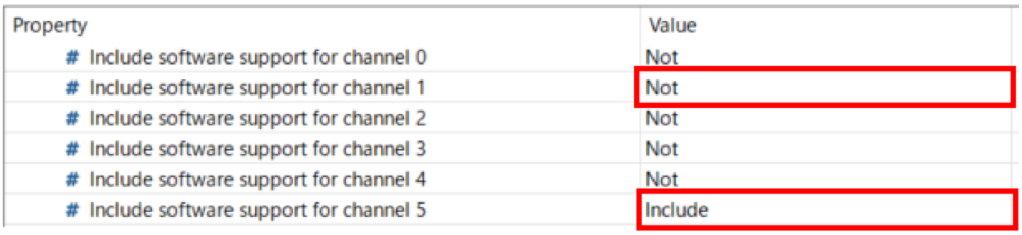
In the software component configuration screen, the pins of RXD1 and TXD1 are set to be enabled.



Property	Value
⌄ SCI	
⌄ SCI1	☑
SCK1 Pin	□ Used
RXD1/SMISO1/SSCL1 Pin	☑ Used
TXD1/SMOSI1/SSDA1 Pin	☑ Used
CTS1#/RTS1#/SS1# Pin	□ Used

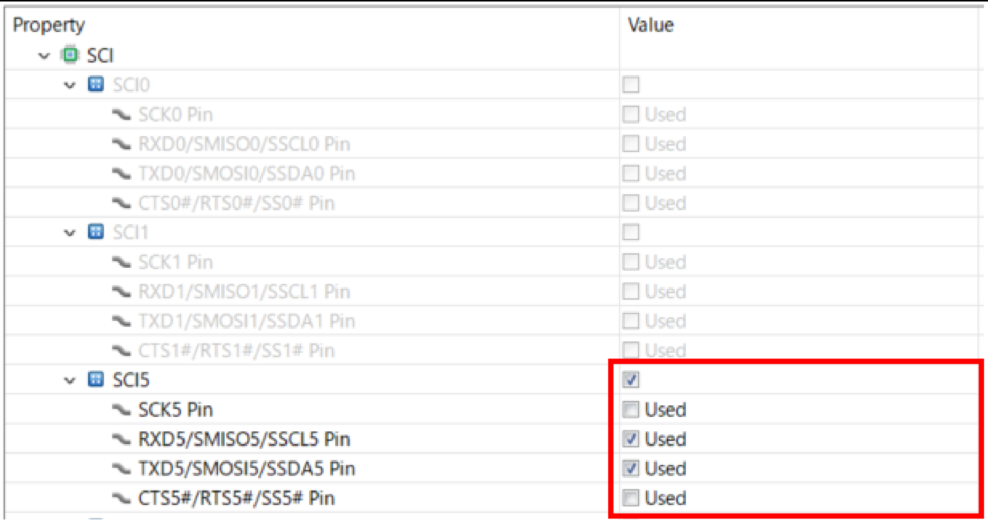
(3) Modifying the SCI FIT module (RX231 group only)

In the software component configuration screen, the SCI channel used in the SCI FIT module is changed from CH1 to CH5.



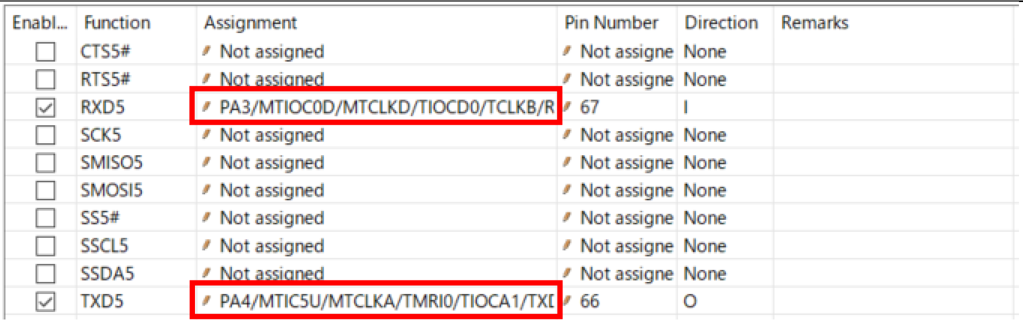
Property	Value
# Include software support for channel 0	Not
# Include software support for channel 1	Not
# Include software support for channel 2	Not
# Include software support for channel 3	Not
# Include software support for channel 4	Not
# Include software support for channel 5	Include

In the software component configuration screen, the pins of RXD5 and TXD5 are set to be enabled.



Property	Value
SCI	
SCI0	□
SCK0 Pin	□ Used
RXD0/SMISO0/SSCL0 Pin	□ Used
TXD0/SMOSI0/SSDA0 Pin	□ Used
CTS0#/RTS0#/SS0# Pin	□ Used
SCI1	□
SCK1 Pin	□ Used
RXD1/SMISO1/SSCL1 Pin	□ Used
TXD1/SMOSI1/SSDA1 Pin	□ Used
CTS1#/RTS1#/SS1# Pin	□ Used
SCI5	☑
SCK5 Pin	□ Used
RXD5/SMISO5/SSCL5 Pin	☑ Used
TXD5/SMOSI5/SSDA5 Pin	☑ Used
CTS5#/RTS5#/SS5# Pin	□ Used

In the pin configuration screen, the assignment is changed to PA3 for RXD5 and PA4 for TXD5.



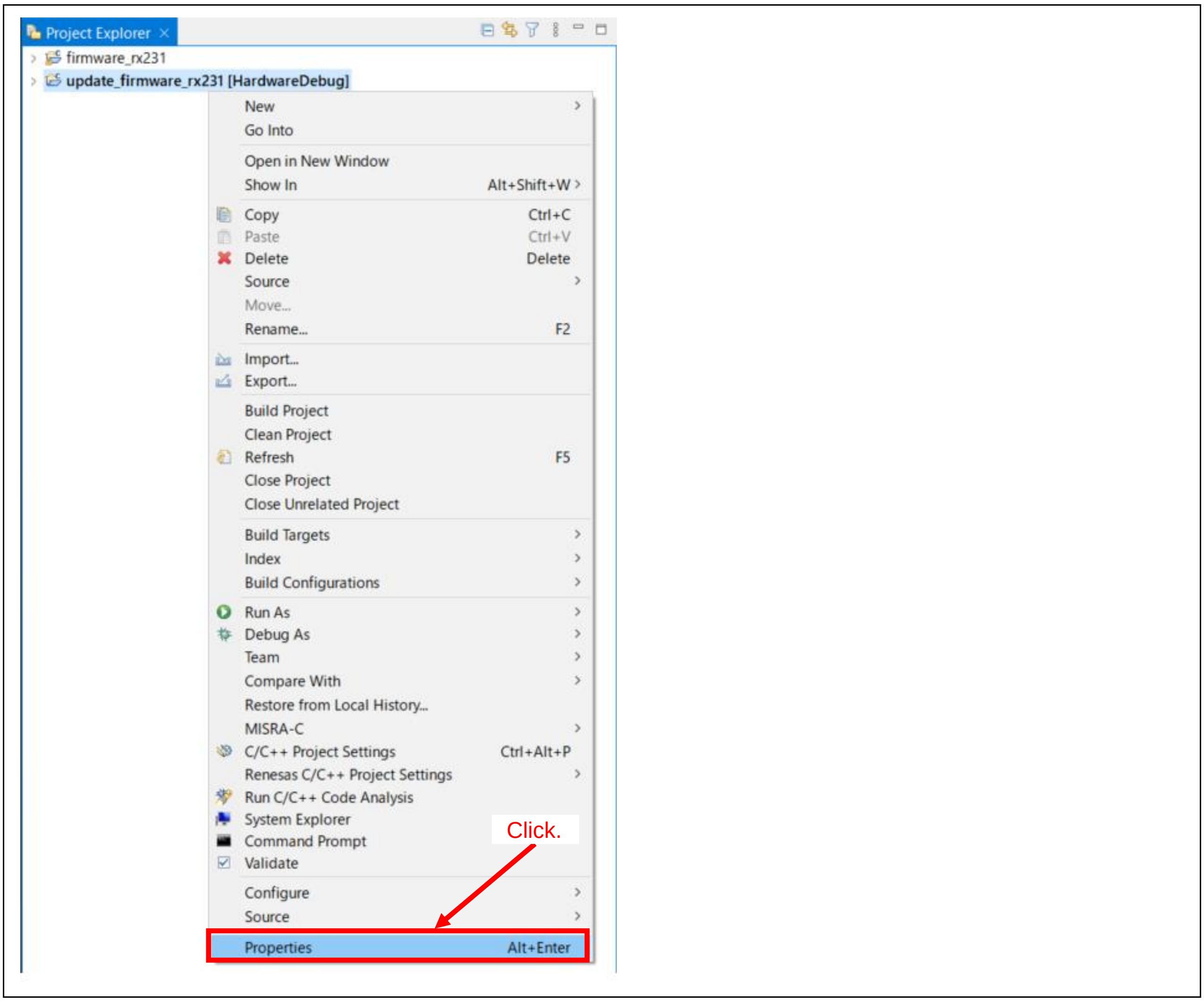
Enabl...	Function	Assignment	Pin Number	Direction	Remarks
□	CTS5#	Not assigned	Not assigne	None	
□	RTS5#	Not assigned	Not assigne	None	
☑	RXD5	PA3/MTIOC0D/MTCLKD/TIOCD0/TCLKB/R	67	I	
□	SCK5	Not assigned	Not assigne	None	
□	SMISO5	Not assigned	Not assigne	None	
□	SMOSI5	Not assigned	Not assigne	None	
□	SS5#	Not assigned	Not assigne	None	
□	SSCL5	Not assigned	Not assigne	None	
□	SSDA5	Not assigned	Not assigne	None	
☑	TXD5	PA4/MTIC5U/MTCLKA/TMRI0/TIOCA1/TXI	66	O	

3.4.2 Modifying the Project Setting

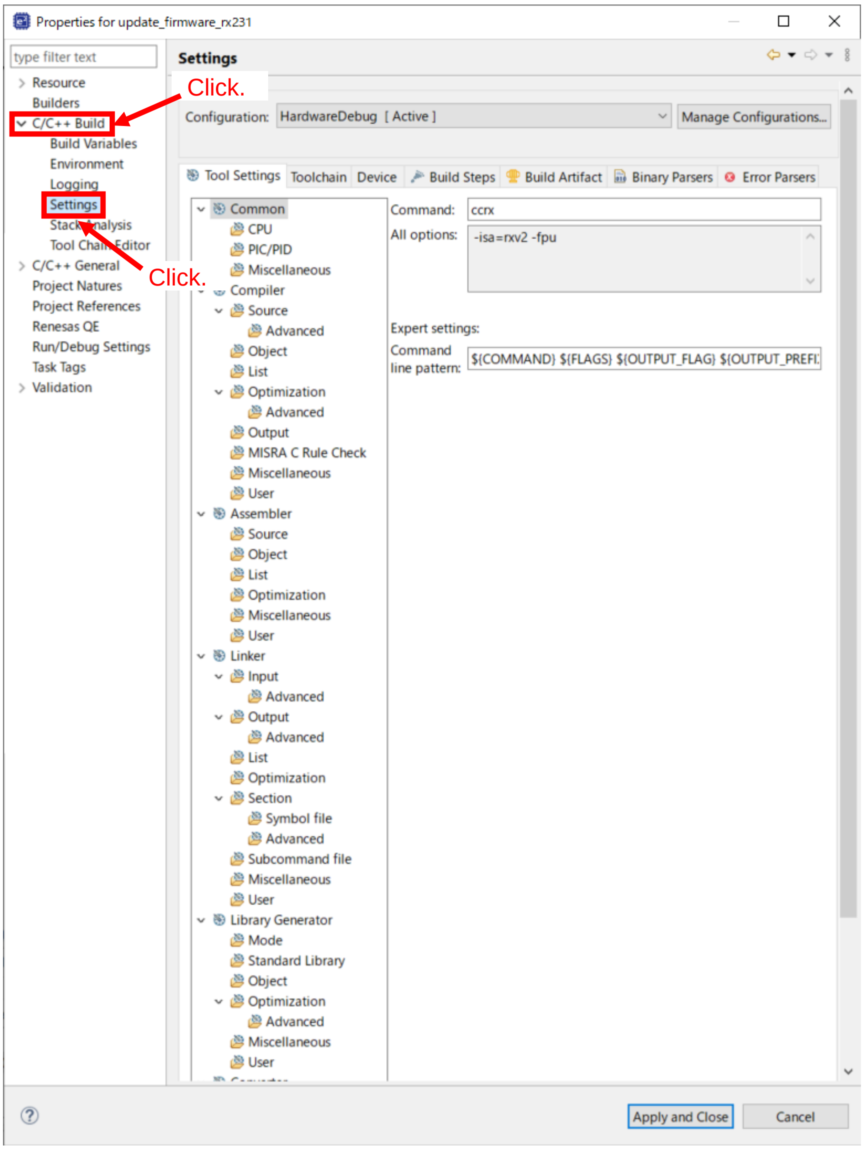
The default build-time settings in the firmware update project have been changed to values described in Table 3.1 and Table 3.2, respectively. The default build-time settings in the firmware project have been changed to values described in Table 3.3, respectively.

You can confirm the settings changed with the following procedure:

1. Right click on the target project (update_firmware_rx130 or firmware_rx130 for the RX130 group, update_firmware_rx140 or firmware_rx140 for the RX140 group, and update_firmware_rx231 or firmware_rx231 for the RX231 group) in the Project Explorer and select Properties from the context menu. * The following screenshots illustrate an example configuration for the RX231 group.



2. Select “C/C++ Build” and then “Settings”. * The following screenshots illustrate an example configuration for the RX231 group.



Click.

Click.

Configuration: HardwareDebug [Active]

Command: ccrx

All options: -isa=rxv2 -fpu

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFI

3. In the Tool Settings tab, confirm that settings have been changed to values described in Table 3.1, Table 3.2 and Table 3.3 for the firmware update project and the firmware project, respectively.

Table 3.1 Changed Build Settings of the Project (Firmware Update Project) (1/2)

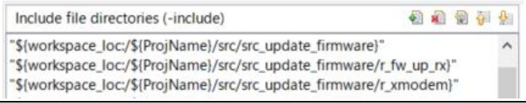
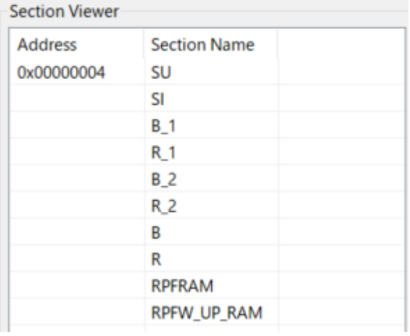
Item	Changed Item	Description
Compiler - Source	Include path is added to the “Include file directories” section.	Add include paths each FIT module needs to specify. When using the Smart Configurator to incorporate the FIT module, the include path is specified automatically.
		Add include paths for the sample program. In this project, “src/src_update_firmware”, “src/src_update_firmware/r_fw_up_rx”, and “src/src_update_firmware/r_xmodem” are added. Setting example:  Include file directories (-include) "${workspace_loc:/${ProjName}/src/src_update_firmware}" "${workspace_loc:/${ProjName}/src/src_update_firmware/r_fw_up_rx}" "${workspace_loc:/${ProjName}/src/src_update_firmware/r_xmodem}"
Linker - Section	Sections RPFRAM and RPFW_UP_RAM are added to the RAM area.	Specify the RAM area to be used by the sample program. Setting example:  Section Viewer Address / Section Name 0x00000004 / SU SI B_1 R_1 B_2 R_2 B R RPFRAM RPFW_UP_RAM

Table 3.2 Changed Build Settings of the Project (Firmware Update Project) (2/2)

<table>
<tr><th>Item</th><th>Changed Item</th><th>Description</th></tr>
<tr><td>Linker
- Section</td><td>Sections FW_UP_VER and FW_UP_COMPLETE are added to the ROM area.

The start address of the FW_UP_VER section is set as follows:
0xFFFF6800 for RX130 group
0xFFFF6000 for RX140 and RX231 groups

The start address of the C_1 section is set as follows:
0xFFFF6808 for RX130 group
0xFFFF6008 for RX140 and RX231 groups

The start address of the FW_UP_COMPLETE section is set as follows:
0xFFFF73F0 for RX130 group
0xFFFF6FF0 for RX140 and RX231 groups

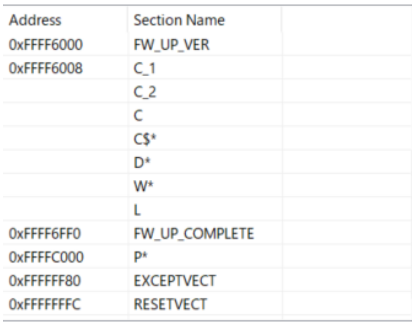
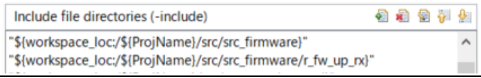
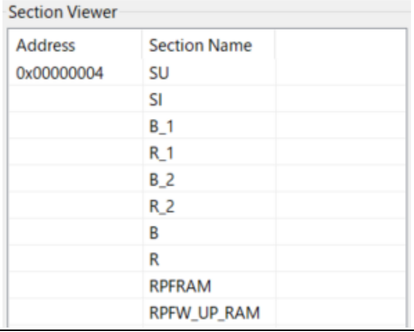
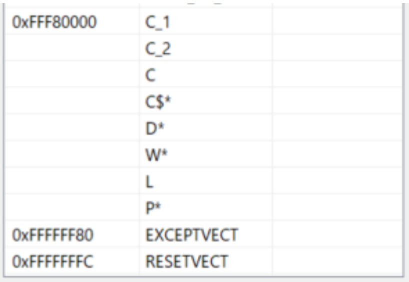
The start address of the P* section is set to 0xFFFFC000.</td><td>Specify the ROM area to place the sample program. Place the constants data above the alternate area of the start-up program protection and place the P* section in the default area of the start-up program protection.
Setting example:

<table>
<tr><th>Address</th><th>Section Name</th></tr>
<tr><td>0xFFFF6000</td><td>FW_UP_VER</td></tr>
<tr><td>0xFFFF6008</td><td>C_1</td></tr>
<tr><td></td><td>C_2</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td></td><td>C$*</td></tr>
<tr><td></td><td>D*</td></tr>
<tr><td></td><td>W*</td></tr>
<tr><td></td><td>L</td></tr>
<tr><td>0xFFFF6FF0</td><td>FW_UP_COMPLETE</td></tr>
<tr><td>0xFFFFC000</td><td>P*</td></tr>
<tr><td>0xFFFFFF80</td><td>EXCEPTVECT</td></tr>
<tr><td>0xFFFFFFFC</td><td>RESETVECT</td></tr>
</table></td></tr>
<tr><td>Linker
- Section
- Symbol file</td><td>‘PFRAM=RPFRAM’ and ‘PFW_UP_RAM=RPFW_UP_RAM’ are added to the “ROM to RAM mapped section” section.</td><td>Add the ROM to RAM mapping since the sample program executes the program for rewriting the code flash memory on the RAM.</td></tr>
</table>

Table 3.3 Changed Build Settings of the Project (Firmware Project)

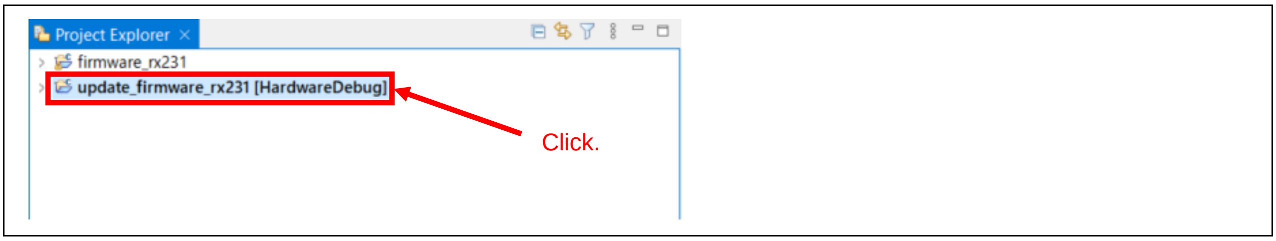
Item	Changed Item	Description
Compiler - Source	Include path is added to the "Include file directories" section.	Add include paths each FIT module needs to specify. When using the Smart Configurator to incorporate the FIT module, the include path is specified automatically.
		Add include paths for the sample program. In this project, "src/src_firmware" and "src/src_firmware/r_fw_up_rx" are added. Setting example:  Include file directories (-include) "${workspace_loc:/${ProjName}/src/src_firmware}" "${workspace_loc:/${ProjName}/src/src_firmware/r_fw_up_rx}"
Linker - Section	Sections RPFRAM and RPFW_UP_RAM are added to the RAM area.	Specify the RAM area to be used by the sample program. Setting example:  Section Viewer Address / Section Name 0x00000004 SU SI B_1 R_1 B_2 R_2 B R RPFRAM RPFW_UP_RAM
Linker - Section	The start address of the section placed in the ROM is set to: 0xFFFE0000 for RX130 group 0xFFFC0000 for RX140 group 0xFFF80000 for RX231 group	Specify the ROM area to place the sample program. Place the sample program at the start address of the code flash memory. Setting example:  0xFFF80000 C_1 C_2 C C$* D* W* L P* 0xFFFFFF80 EXCEPTVECT 0xFFFFFFFC RESETVECT Note: Do not place data in the following areas since the firmware program cannot use these areas: For RX130 group: Addresses FFFF_6800h to FFFF_BFFFh For RX140 and RX231 groups: Addresses FFFF_6000h to FFFF_BFFFh
Linker - Section - Symbol file	'PFRAM=RPFRAM' and 'PFW_UP_RAM=RPFW_UP_RAM' are added to the "ROM to RAM mapped section" section.	Add the ROM to RAM mapping since the sample program executes the program for switching the start-up program protection area on the RAM.

4. Operation Confirmation

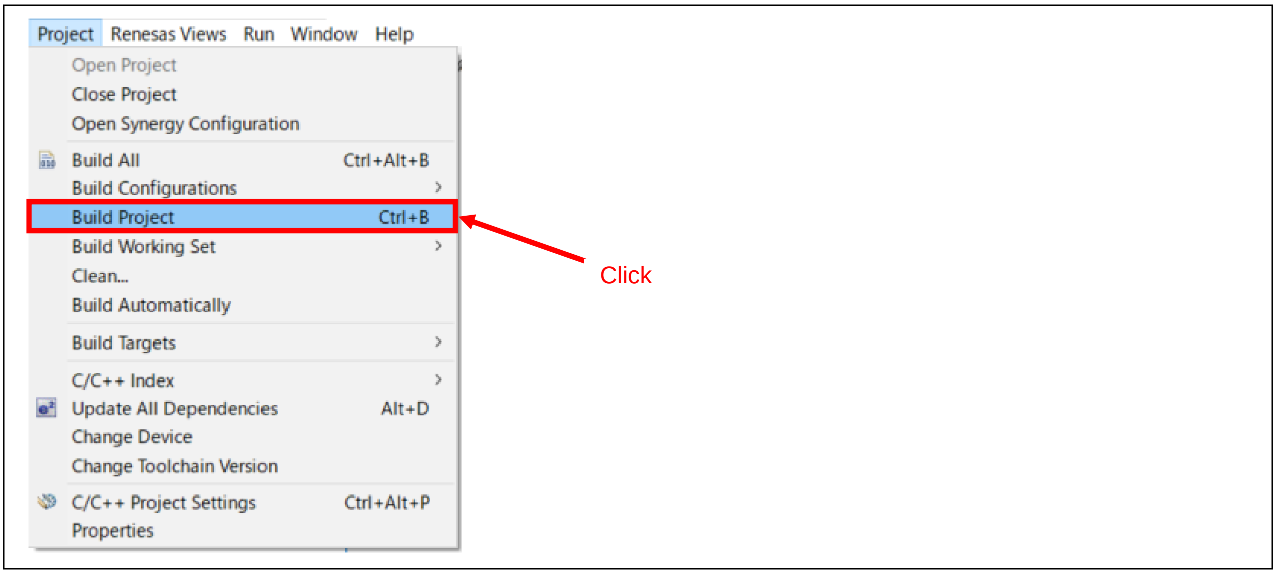
4.1 Building the Project

Follow the procedure below to build the project and create the load module.

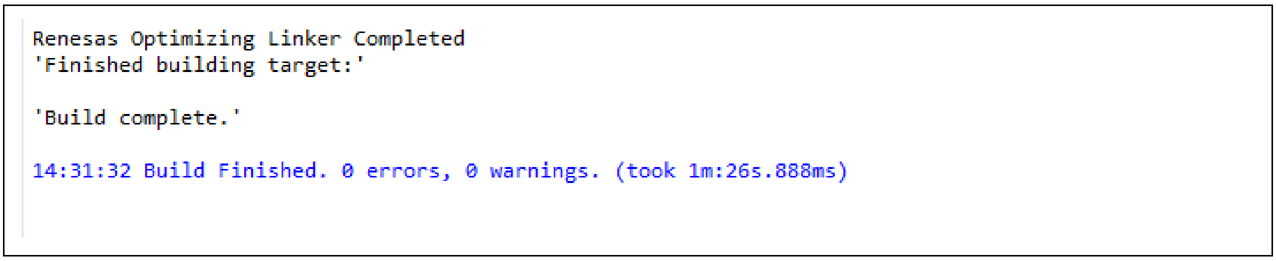
1. Click the project to be built (update_firmware_rx130 or firmware_rx130 for RX130 group, update_firmware_rx140 or firmware_rx140 for RX140 group, and update_firmware_rx231 or firmware_rx231 for RX231 group). * The following screenshots illustrate an example configuration for the RX231 group.



2. Select *Project >> Build Project*.



3. The build is completed when the message "Build complete." is displayed in the Console panel.



```
Renesas Optimizing Linker Completed
'Finished building target:'

'Build complete.'

14:31:32 Build Finished. 0 errors, 0 warnings. (took 1m:26s.888ms)
```

4.2 Preparing Debugging

4.2.1 Preparing Devices

The evaluation board needs to be prepared before debugging.

Table 4.1 lists Devices and Configurations and Figure 4.1 shows the Debug Configuration.

Table 4.1 Devices and Configurations

No.	Device	Remarks
1	Development PC	PC used for development
2	Evaluation board (Renesas Starter Kit for RX130/RX140/RX231)	—
3	Host PC • Serial communication software which is capable of XMODEM/SUM transfer	Development PC can be used as the host PC.
4	USB cable (Mini Type-B)	Renesas Starter Kit for RX130/RX140/RX231 converts serial I/O signals from RX231 to USB serial data. When it is connected to the host PC via USB, it can work as a virtual com port.

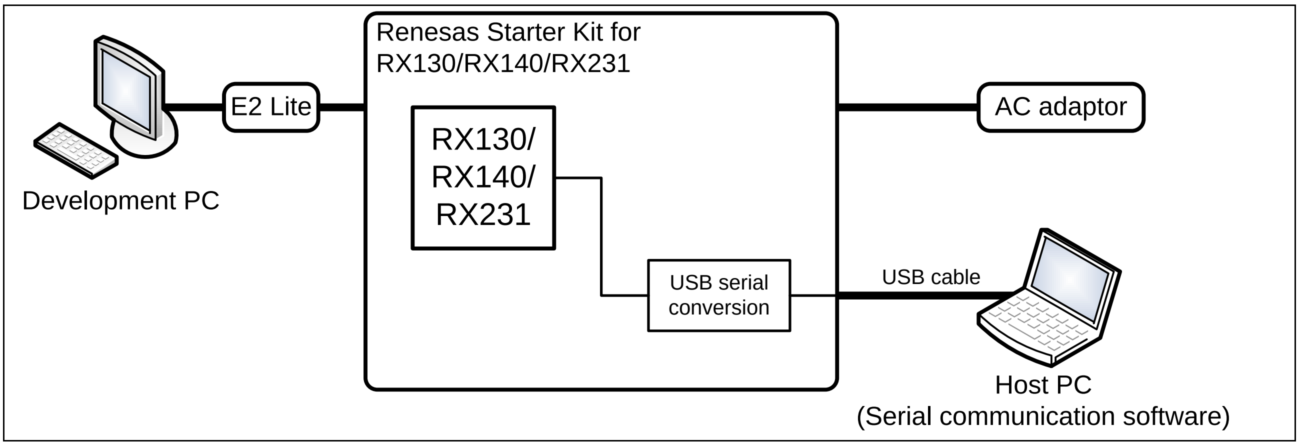


Figure 4.1 Debug Configuration

4.2.2 Setting the Host PC

Table 4.2 lists the serial communication specification for the device and the host PC. For configuration of the terminal software, refer to the document for the terminal software.

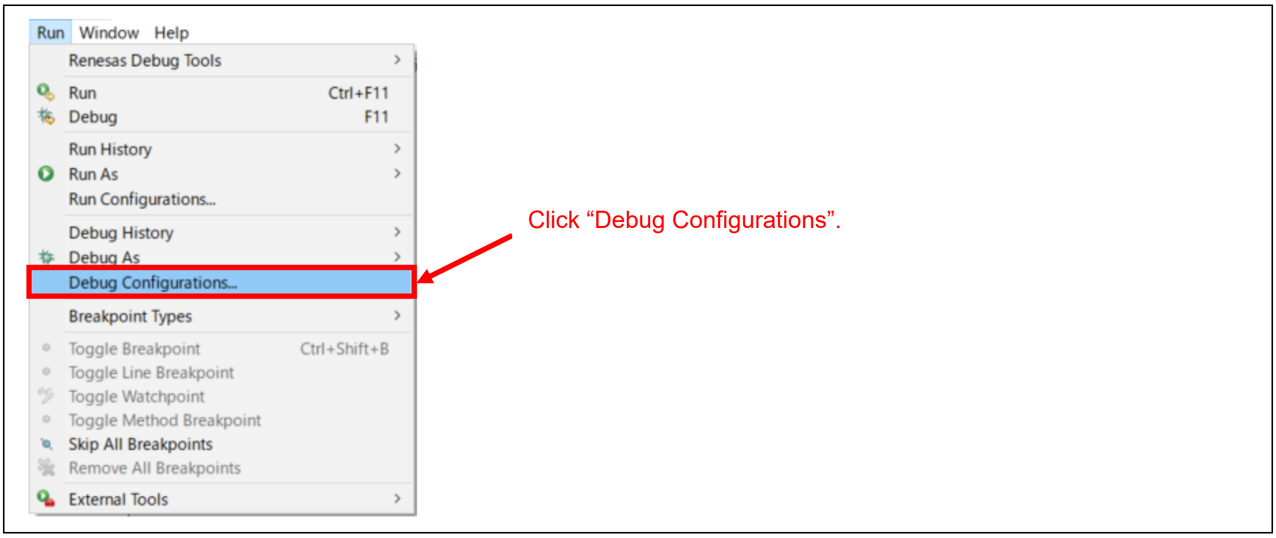
Table 4.2 Communication Specification

Item	Description
Communication method	Asynchronous communication
Bit rate	115200 bps
Data length	8 bits
Parity	None
Stop bit	1 bit
Flow control	None

4.3 Debugging the Project

Follow the procedure below to start debugging the firmware update project. The procedure can be used for the firmware project in the same manner.

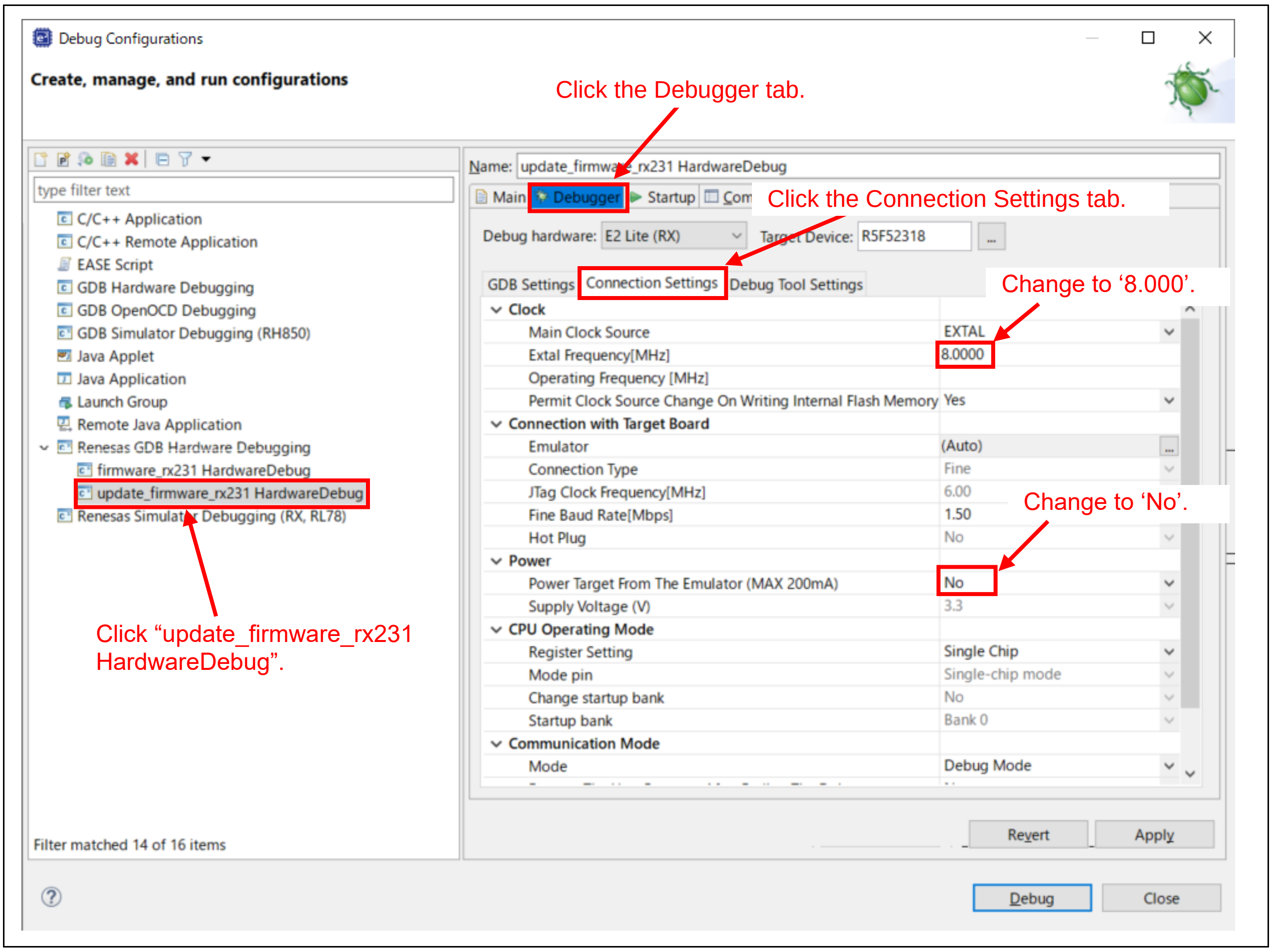
1. Select *Run* >> *Debug Configurations* in the e^2 studio.



Click “Debug Configurations”.

2. RX130 group:
   Select “update_firmware_rx130 HardwareDebug” under “Renesas GDB Hardware Debugging”.
   RX140 group:
   Select “update_firmware_rx140 HardwareDebug” under “Renesas GDB Hardware Debugging”.
   RX231 group:
   Select “update_firmware_rx231 HardwareDebug” under “Renesas GDB Hardware Debugging”.

   Click the Debugger tab and then the Connection Settings tab. Change “EXTAL frequency value” to ‘8.0000’ and “Power Target From The Emulator” to ‘No’. * The following screenshots illustrate an example configuration for the RX231 group.



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- EASE Script
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB Simulator Debugging (RH850)
- Java Applet
- Java Application
- Launch Group
- Remote Java Application
- Renesas GDB Hardware Debugging
  - firmware_rx231 HardwareDebug
  - update_firmware_rx231 HardwareDebug
- Renesas Simulator Debugging (RX, RL78)

Filter matched 14 of 16 items

Name: update_firmware_rx231 HardwareDebug

Debug hardware: E2 Lite (RX) Target Device: R5F52318

Clock	
Main Clock Source	EXTAL
Extal Frequency[MHz]	8.0000
Operating Frequency [MHz]	
Permit Clock Source Change On Writing Internal Flash Memory	Yes
Connection with Target Board	
Emulator	(Auto)
Connection Type	Fine
JTag Clock Frequency[MHz]	6.00
Fine Baud Rate[Mbps]	1.50
Hot Plug	No
Power	
Power Target From The Emulator (MAX 200mA)	No
Supply Voltage (V)	3.3
CPU Operating Mode	
Register Setting	Single Chip
Mode pin	Single-chip mode
Change startup bank	No
Startup bank	Bank 0
Communication Mode	
Mode	Debug Mode

Click the Debugger tab.

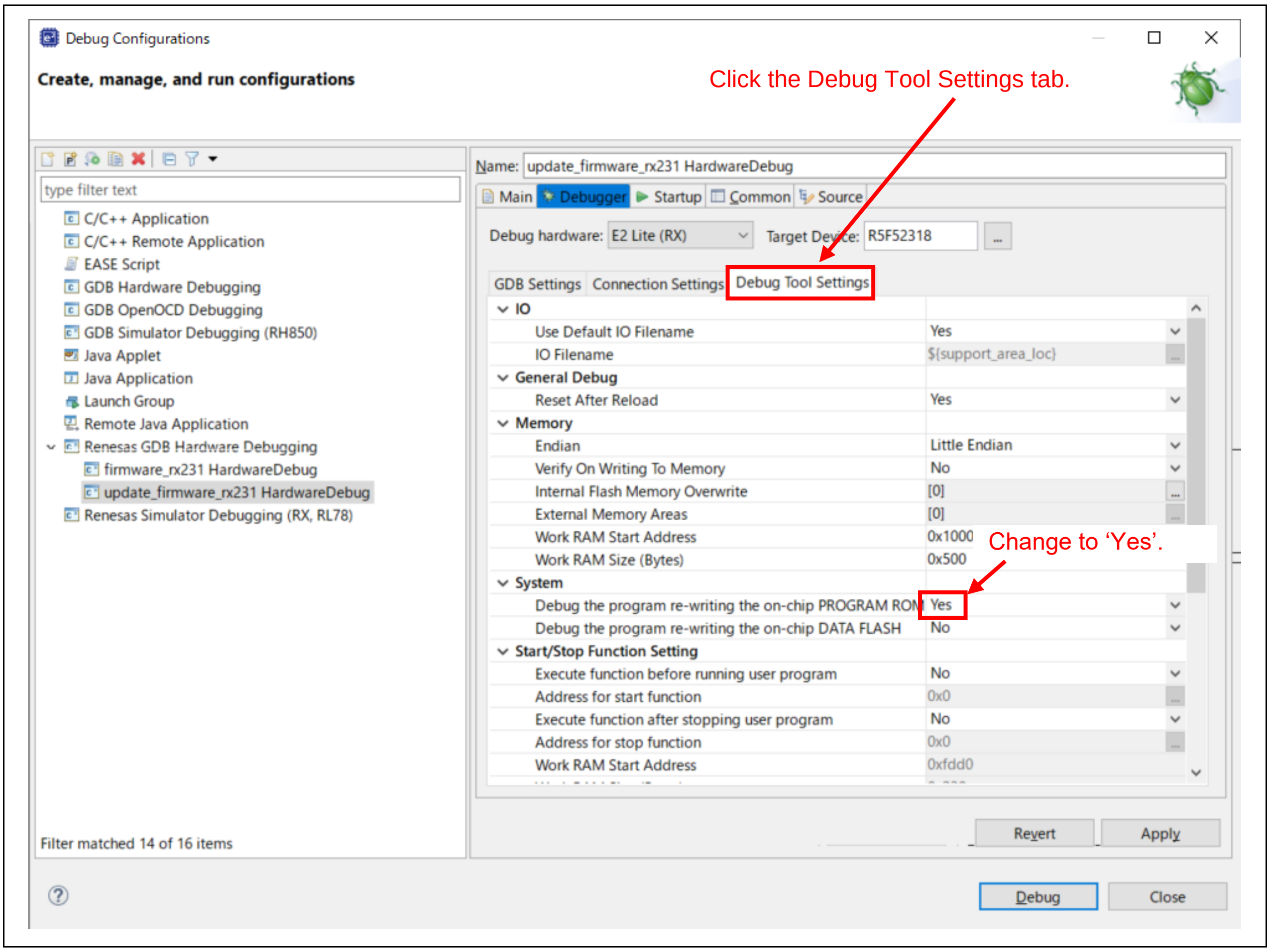
Click the Connection Settings tab.

Change to ‘8.000’.

Change to ‘No’.

Click “update_firmware_rx231 HardwareDebug”.

3. Click the Debug Tool Settings tab. Change “Debug the program re-writing the on-chip PROGRAM ROM” to ‘Yes’. * The following screenshots illustrate an example configuration for the RX231 group.



Debug Configurations

Create, manage, and run configurations

Click the Debug Tool Settings tab.

type filter text

- C/C++ Application
- C/C++ Remote Application
- EASE Script
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB Simulator Debugging (RH850)
- Java Applet
- Java Application
- Launch Group
- Remote Java Application
- Renesas GDB Hardware Debugging
  - firmware_rx231 HardwareDebug
  - update_firmware_rx231 HardwareDebug
- Renesas Simulator Debugging (RX, RL78)

Filter matched 14 of 16 items

Name: update_firmware_rx231 HardwareDebug

Main | Debugger | Startup | Common | Source

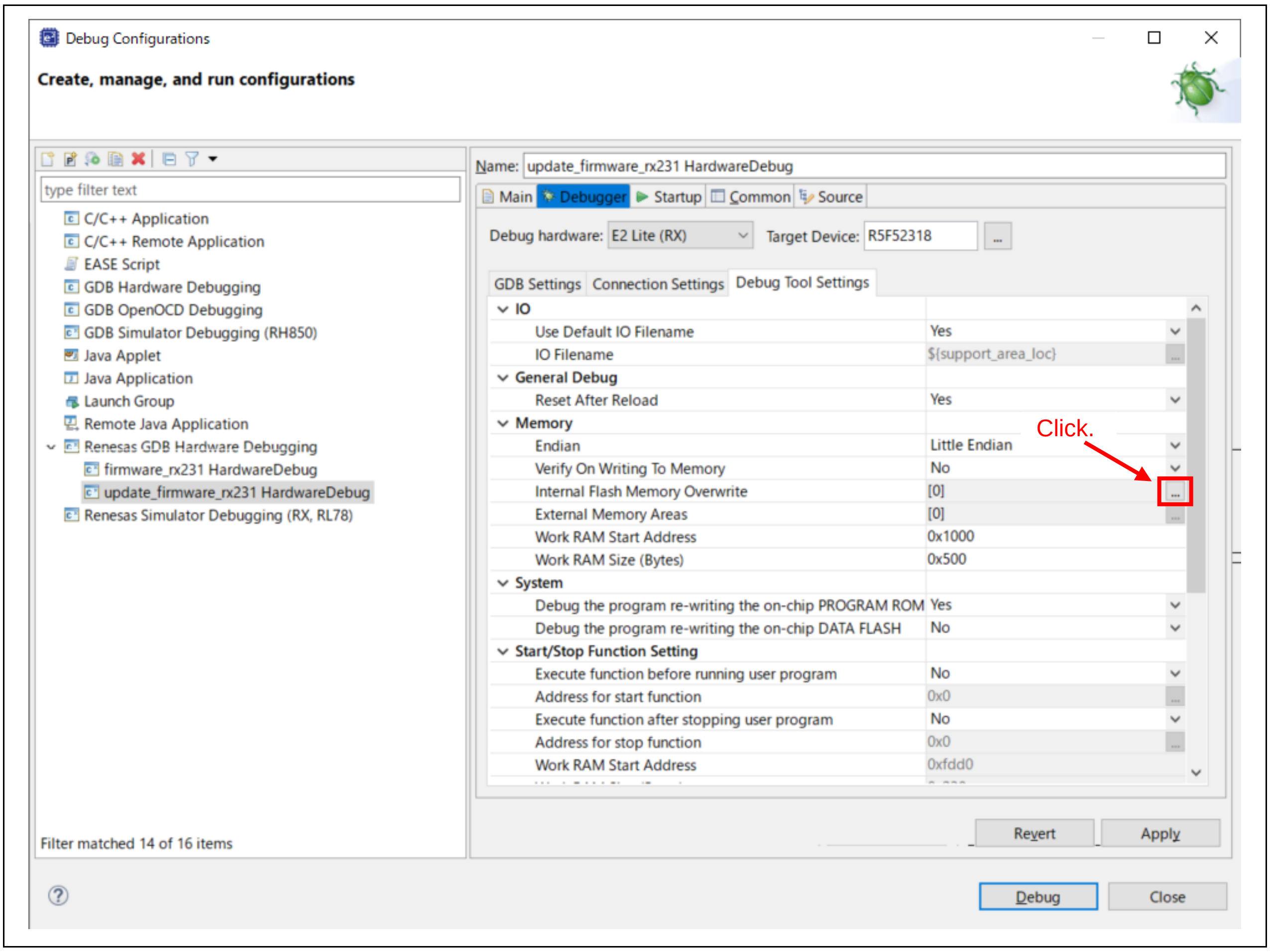
Debug hardware: E2 Lite (RX) Target Device: R5F52318

GDB Settings | Connection Settings | Debug Tool Settings

IO	
Use Default IO Filename	Yes
IO Filename	${support_area_loc}
General Debug	
Reset After Reload	Yes
Memory	
Endian	Little Endian
Verify On Writing To Memory	No
Internal Flash Memory Overwrite	[0]
External Memory Areas	[0]
Work RAM Start Address	0x1000
Work RAM Size (Bytes)	0x500
System	
Debug the program re-writing the on-chip PROGRAM ROM	Yes
Debug the program re-writing the on-chip DATA FLASH	No
Start/Stop Function Setting	
Execute function before running user program	No
Address for start function	0x0
Execute function after stopping user program	No
Address for stop function	0x0
Work RAM Start Address	0xfdd0

Change to ‘Yes’.

4. Click "…" button on the right side of "Internal Flash Memory Overwrite". * The following screenshots illustrate an example configuration for the RX231 group.



Debug Configurations

Create, manage, and run configurations

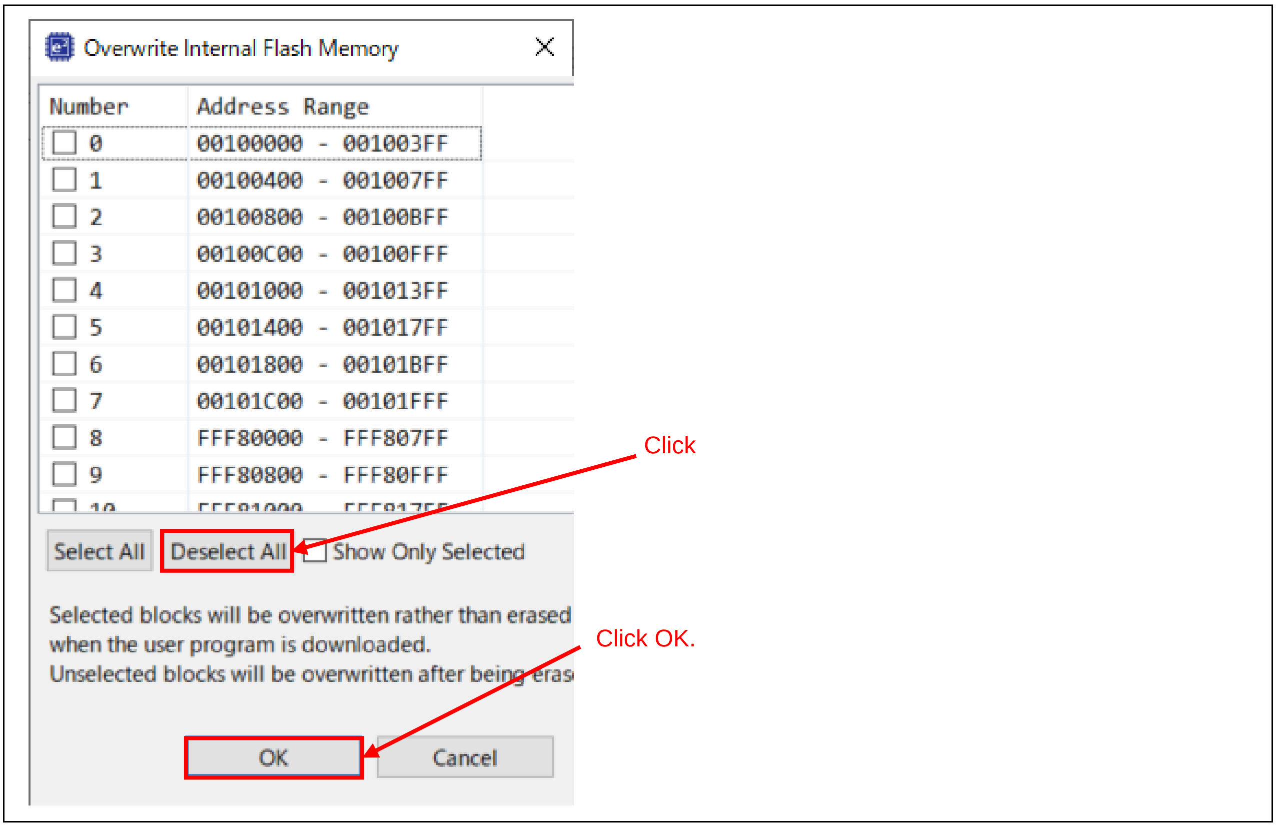
Name: update_firmware_rx231 HardwareDebug

Debug hardware: E2 Lite (RX) Target Device: R5F52318

Setting	Value
IO	
Use Default IO Filename	Yes
IO Filename	${support_area_loc}
General Debug	
Reset After Reload	Yes
Memory	
Endian	Little Endian
Verify On Writing To Memory	No
Internal Flash Memory Overwrite	[0]
External Memory Areas	[0]
Work RAM Start Address	0x1000
Work RAM Size (Bytes)	0x500
System	
Debug the program re-writing the on-chip PROGRAM ROM	Yes
Debug the program re-writing the on-chip DATA FLASH	No
Start/Stop Function Setting	
Execute function before running user program	No
Address for start function	0x0
Execute function after stopping user program	No
Address for stop function	0x0
Work RAM Start Address	0xfdd0

Filter matched 14 of 16 items

5. Specify to execute overwrite operation after erasing all blocks in the code flash memory. Click the Deselect All button and click OK.



Overwrite Internal Flash Memory

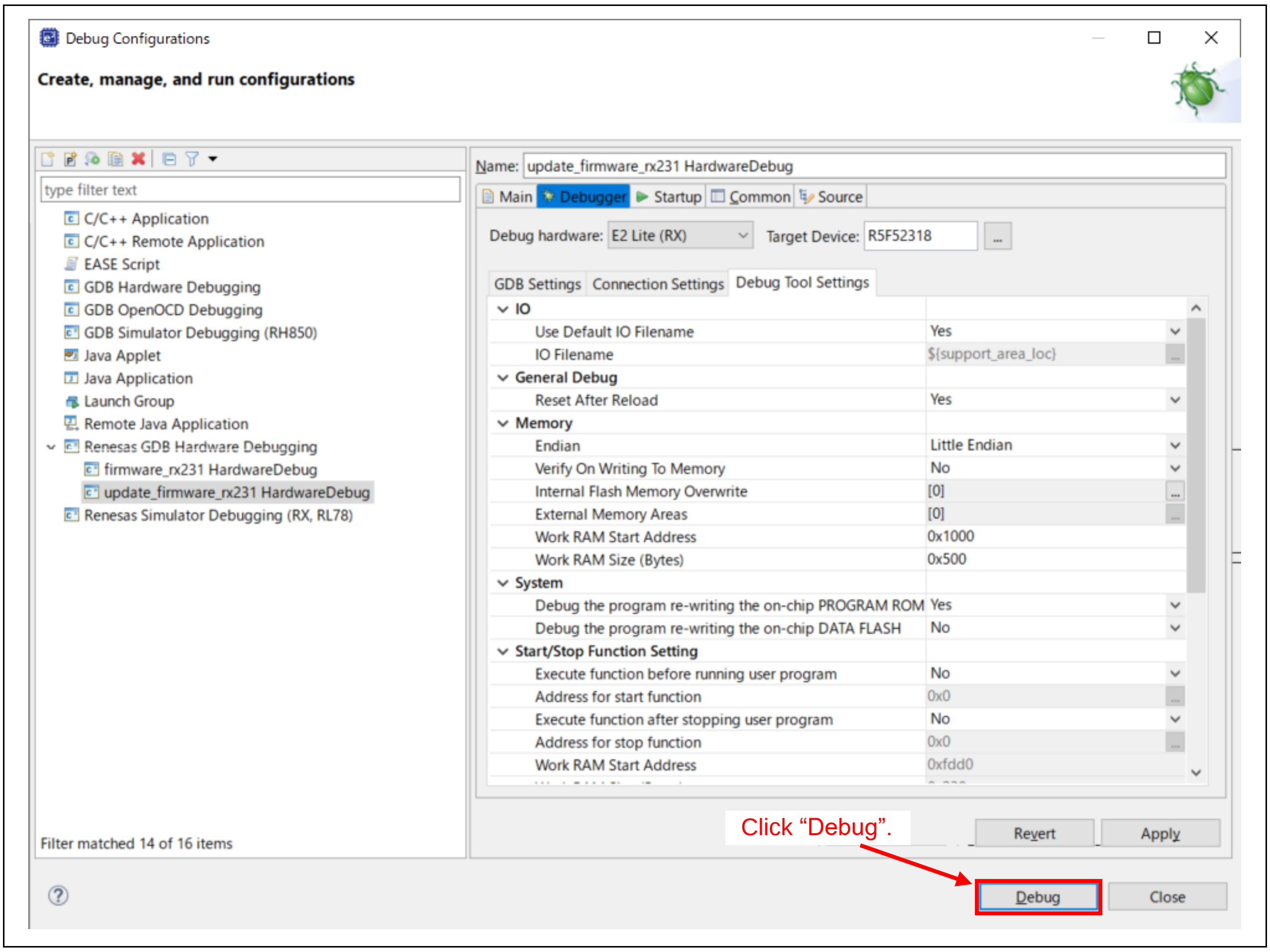
Number	Address Range
☐ 0	00100000 - 001003FF
☐ 1	00100400 - 001007FF
☐ 2	00100800 - 00100BFF
☐ 3	00100C00 - 00100FFF
☐ 4	00101000 - 001013FF
☐ 5	00101400 - 001017FF
☐ 6	00101800 - 00101BFF
☐ 7	00101C00 - 00101FFF
☐ 8	FFF80000 - FFF807FF
☐ 9	FFF80800 - FFF80FFF
☐ 10	FFF81000 - FFF817FF

Select All Deselect All ☐ Show Only Selected

Selected blocks will be overwritten rather than erased when the user program is downloaded.
Unselected blocks will be overwritten after being eras

OK Cancel

6. Click the Debug button. * The following screenshots illustrate an example configuration for the RX231 group.



Debug Configurations

Create, manage, and run configurations

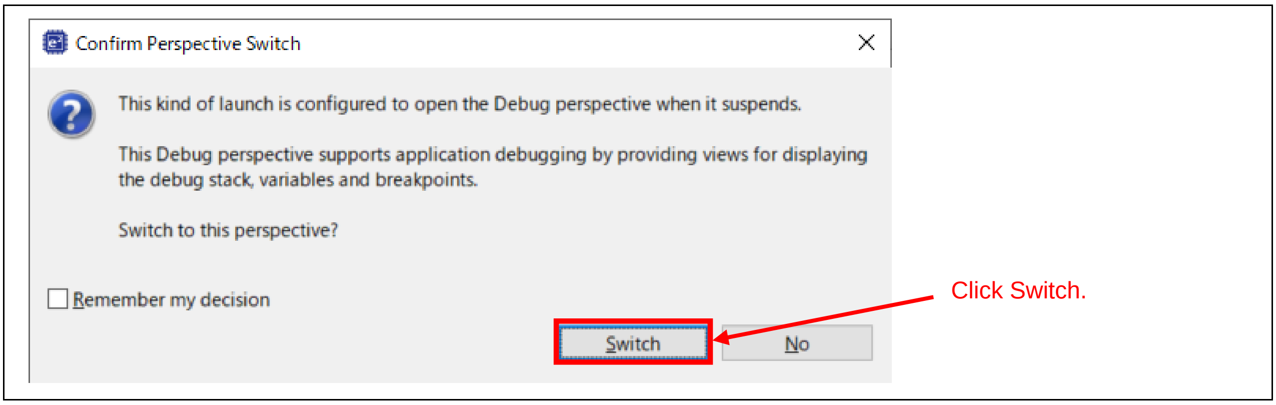
Name: update_firmware_rx231 HardwareDebug

Debug hardware: E2 Lite (RX)   Target Device: R5F52318

IO	
Use Default IO Filename	Yes
IO Filename	${support_area_loc}
General Debug	
Reset After Reload	Yes
Memory	
Endian	Little Endian
Verify On Writing To Memory	No
Internal Flash Memory Overwrite	[0]
External Memory Areas	[0]
Work RAM Start Address	0x1000
Work RAM Size (Bytes)	0x500
System	
Debug the program re-writing the on-chip PROGRAM ROM	Yes
Debug the program re-writing the on-chip DATA FLASH	No
Start/Stop Function Setting	
Execute function before running user program	No
Address for start function	0x0
Execute function after stopping user program	No
Address for stop function	0x0
Work RAM Start Address	0xfdd0

Click “Debug”.

When the following message appears, click Switch.



Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

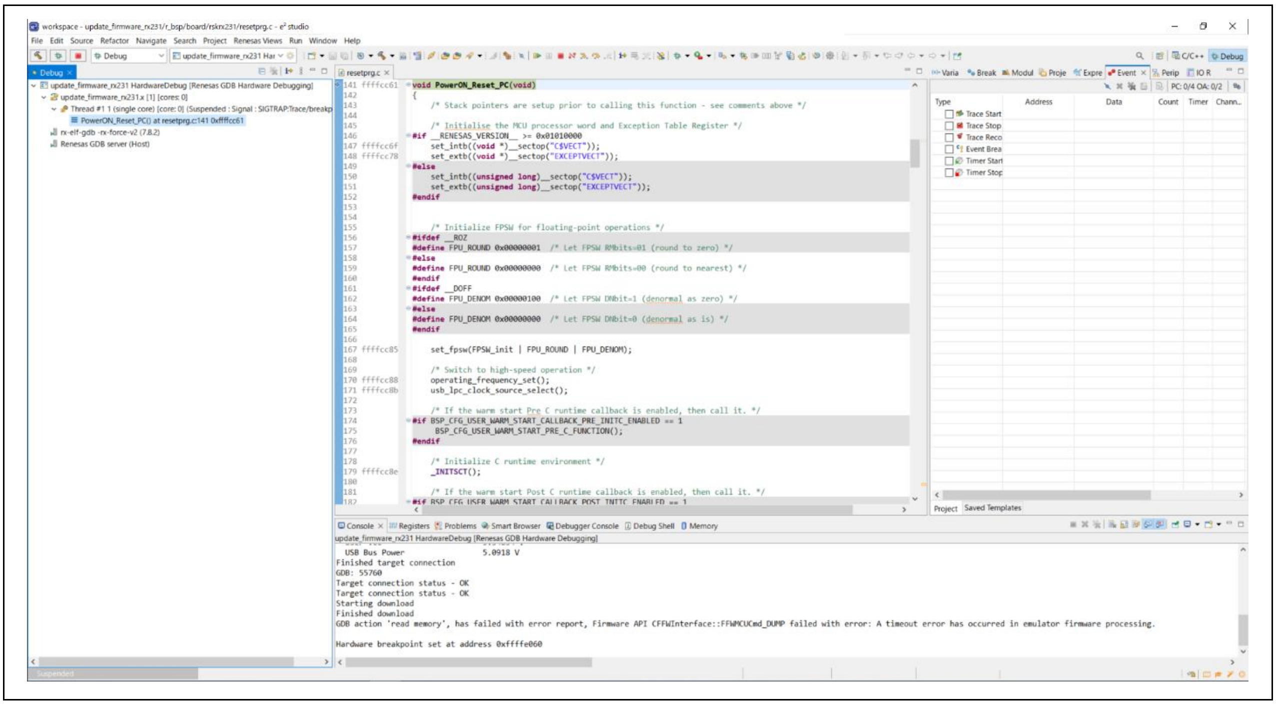
This Debug perspective supports application debugging by providing views for displaying the debug stack, variables and breakpoints.

Switch to this perspective?

☐ Remember my decision

Click Switch.

When the load module has been downloaded, the Debug perspective opens. * The following screenshots illustrate an example configuration for the RX231 group.


workspace - update_firmware_rx231/r_bsp/board/rskrx231/resetprg.c - e² studio

- update_firmware_rx231 HardwareDebug [Renesas GDB Hardware Debugging]
  - update_firmware_rx231.x [1] [cores: 0]
    - Thread #1 1 (single core) [core: 0] (Suspended : Signal : SIGTRAP:Trace/breakp
      - PowerON_Reset_PC() at resetprg.c:141 0xffffcc61
  - rx-elf-gdb -rx-force-v2 (7.8.2)
  - Renesas GDB server (Host)

```
ffffcc61  void PowerON_Reset_PC(void)
          {
              /* Stack pointers are setup prior to calling this function - see comments above */

              /* Initialise the MCU processor word and Exception Table Register */
          #if __RENESAS_VERSION__ >= 0x01010000
ffffcc6f      set_intb((void *)__sectop("C$VECT"));
ffffcc78      set_extb((void *)__sectop("EXCEPTVECT"));
          #else
              set_intb((unsigned long)__sectop("C$VECT"));
              set_extb((unsigned long)__sectop("EXCEPTVECT"));
          #endif


              /* Initialize FPSW for floating-point operations */
          #ifdef __ROZ
          #define FPU_ROUND 0x00000001  /* Let FPSW RMbits=01 (round to zero) */
          #else
          #define FPU_ROUND 0x00000000  /* Let FPSW RMbits=00 (round to nearest) */
          #endif
          #ifdef __DOFF
          #define FPU_DENOM 0x00000100  /* Let FPSW DNbit=1 (denormal as zero) */
          #else
          #define FPU_DENOM 0x00000000  /* Let FPSW DNbit=0 (denormal as is) */
          #endif

ffffcc85      set_fpsw(FPSW_init | FPU_ROUND | FPU_DENOM);

              /* Switch to high-speed operation */
ffffcc88      operating_frequency_set();
ffffcc8b      usb_lpc_clock_source_select();

              /* If the warm start Pre C runtime callback is enabled, then call it. */
          #if BSP_CFG_USER_WARM_START_CALLBACK_PRE_INITC_ENABLED == 1
               BSP_CFG_USER_WARM_START_PRE_C_FUNCTION();
          #endif

              /* Initialize C runtime environment */
ffffcc8e      _INITSCT();

              /* If the warm start Post C runtime callback is enabled, then call it. */
          #if BSP_CFG_USER_WARM_START_CALLBACK_POST_INITC_ENABLED == 1
```

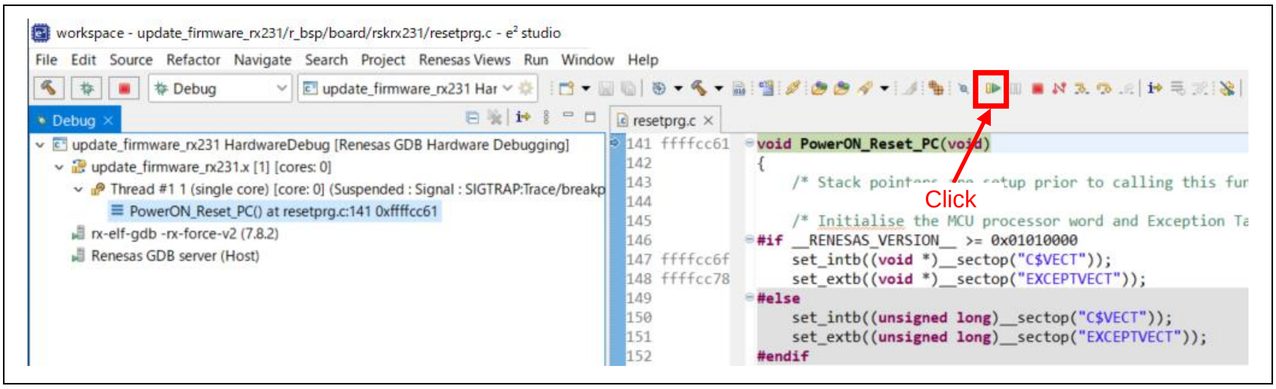
update_firmware_rx231 HardwareDebug [Renesas GDB Hardware Debugging]

```
  USB Bus Power                    5.0918 V
Finished target connection
GDB: 55760
Target connection status - OK
Target connection status - OK
Starting download
Finished download
GDB action 'read memory', has failed with error report, Firmware API CFFWInterface::FFWMCUCmd_DUMP failed with error: A timeout error has occurred in emulator firmware processing.

Hardware breakpoint set at address 0xffffe060
```

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
Console Registers Problems Smart Browser Debugger Console Debug Shell Memory
Varia Break Modul Proje Expre Event Perip IO R
Type Address Data Count Timer Chann...
Trace Start
Trace Stop
Trace Reco
Event Brea
Timer Start
Timer Stop
Project Saved Templates
Suspended

7. Click the Resume icon on the toolbar to execute the program. The program breaks at the beginning of the main function. * The following screenshots illustrate an example configuration for the RX231 group.


workspace - update_firmware_rx231/r_bsp/board/rskrx231/resetprg.c - e² studio

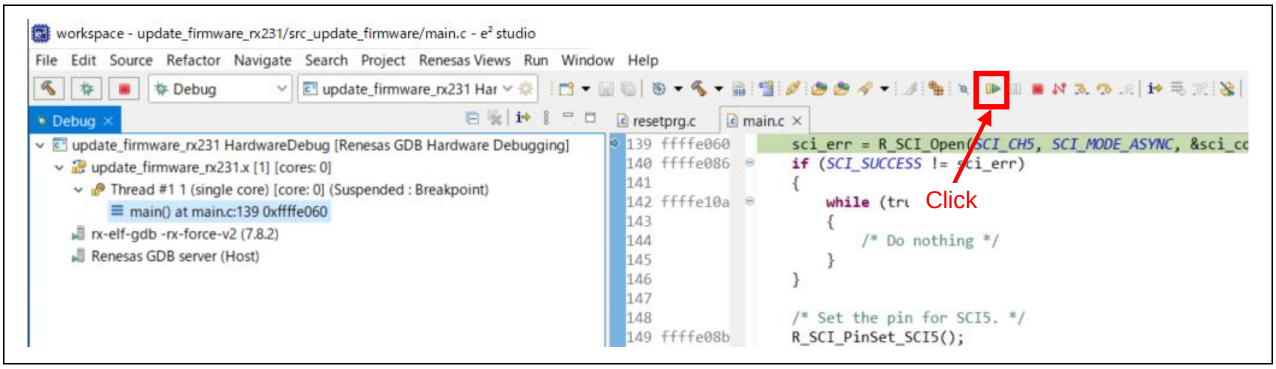
- update_firmware_rx231 HardwareDebug [Renesas GDB Hardware Debugging]
  - update_firmware_rx231.x [1] [cores: 0]
    - Thread #1 1 (single core) [core: 0] (Suspended : Signal : SIGTRAP:Trace/breakp
      - PowerON_Reset_PC() at resetprg.c:141 0xffffcc61
  - rx-elf-gdb -rx-force-v2 (7.8.2)
  - Renesas GDB server (Host)

```
ffffcc61  void PowerON_Reset_PC(void)
          {
              /* Stack pointers are setup prior to calling this fur

              /* Initialise the MCU processor word and Exception Ta
          #if __RENESAS_VERSION__ >= 0x01010000
ffffcc6f      set_intb((void *)__sectop("C$VECT"));
ffffcc78      set_extb((void *)__sectop("EXCEPTVECT"));
          #else
              set_intb((unsigned long)__sectop("C$VECT"));
              set_extb((unsigned long)__sectop("EXCEPTVECT"));
          #endif
```

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
update_firmware_rx231 Har
Debug
resetprg.c
Click

8. Click the Resume icon on the toolbar again after the break in step 7. * The following screenshots illustrate an example configuration for the RX231 group.


workspace - update_firmware_rx231/src_update_firmware/main.c - e² studio

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
update_firmware_rx231 Har
Debug
resetprg.c
main.c
Click

update_firmware_rx231 HardwareDebug [Renesas GDB Hardware Debugging]
update_firmware_rx231.x [1] [cores: 0]
Thread #1 1 (single core) [core: 0] (Suspended : Breakpoint)
main() at main.c:139 0xffffe060
rx-elf-gdb -rx-force-v2 (7.8.2)
Renesas GDB server (Host)

```
ffffe060        sci_err = R_SCI_Open(SCI_CH5, SCI_MODE_ASYNC, &sci_cc
ffffe086        if (SCI_SUCCESS != sci_err)
                {
ffffe10a            while (tru
                    {
                        /* Do nothing */
                    }
                }

                /* Set the pin for SCI5. */
ffffe08b        R_SCI_PinSet_SCI5();
```

9. Confirm the following message is output in the terminal software.

RX130 group:

```
RX130 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute program
>
```

RX140 group:

```
RX140 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
2...Execute program
>
```

RX231 group:

```
RX231 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute program
>
```

5. Application Overview

5.1 Configuration of the Firmware Update Program

This section explains the configuration of the firmware update program, which is a sample program for this application note. This program is stored in the default area of the start-up program protection. Constants data (such as initial values of variables and string literals) are stored in the constants data area in ROM. There are two constants data areas: constants data area 1 and constants data area 2, and constants data are stored in either one of them.

A constants data area consists of three areas: version information storage area, constants data storage area, and write complete information storage area.

Figure 5.1 shows Memory Map of the Firmware Update Program, Table 5.1 shows Configuration of the Constants Data Area.

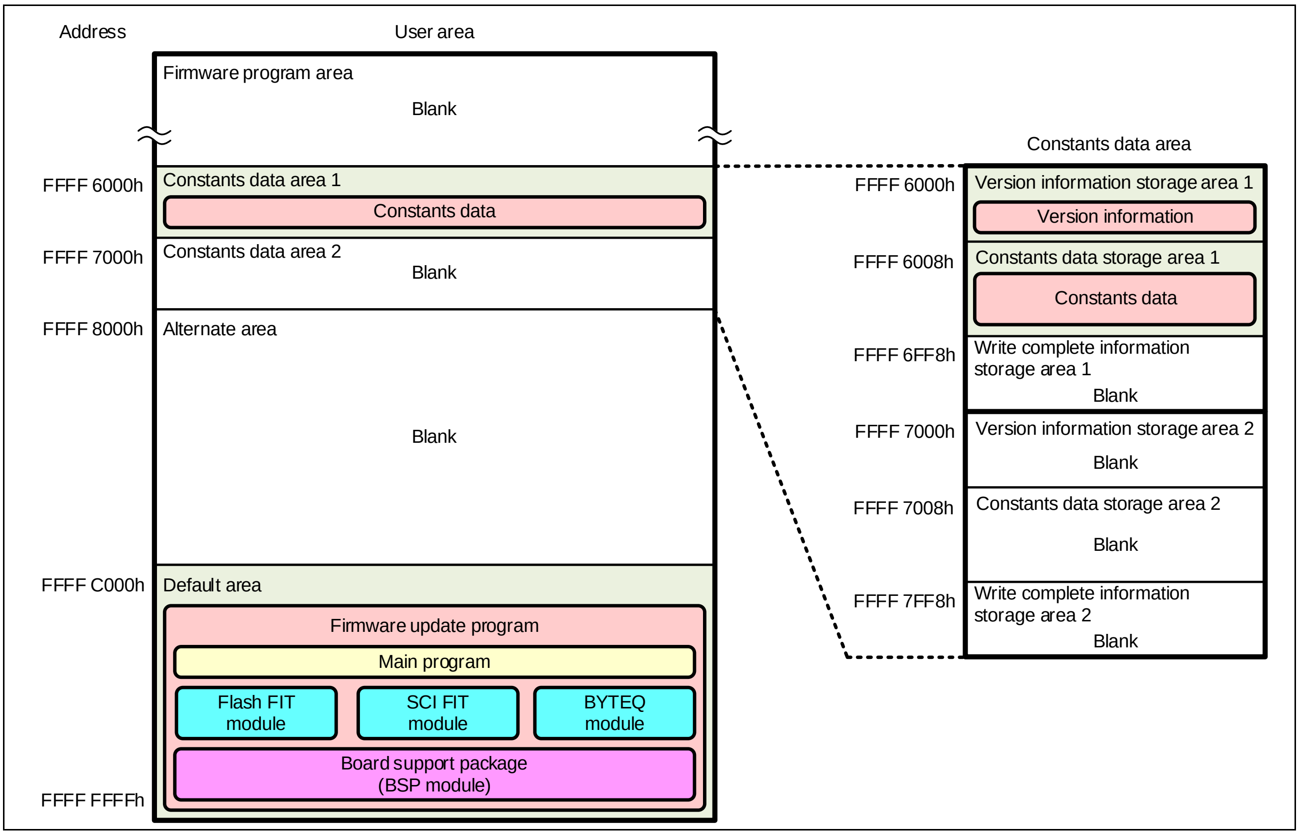


Figure 5.1 Memory Map of the Firmware Update Program

Table 5.1 Configuration of the Constants Data Area

Area Name	Description
Version information storage area	Area for storing the version information of the firmware update program.
Constants data storage area	Area for storing constants data
Write complete information storage area	Area into which the version information is programmed when the firmware update program is updated.

5.2 Operation Overview

This section explains the sample programs; firmware update program and firmware program in this application note.

The firmware update program is stored in the default area of the start-up program protection and the constants data area 1. The firmware update program receives the firmware program (.mot file) through the serial communication using the XMODEM/SUM protocol and program the firmware into the code flash memory. Then the default area and the alternate area of the star-up program protection are switched temporarily to program the code flash memory except the default area and the constants data area. Since this can protect the firmware update program, if programming the firmware failed, for example, due to temporary blackout, the firmware firmware still can be updated by restarting the firmware update program.

The firmware program is programmed in the area other than the default area of the start-up program protection area and constants data area using the firmware update program. The firmware program outputs the message to the host PC using the serial communication. When the firmware receives the command from the host PC, it switches between the default area and the alternate area, and then executes a software reset. This causes the firmware update program to restart.

Table 5.2 lists the Functional Comparison of Sample Programs.

Table 5.2 Functional Comparison of Sample Programs

Function	Firmware Update Program	Firmware Program
Erasing/programming the code flash memory	Supported	Not supported
Switching the start-up program protection area and software reset	Supported	Supported

5.2.1 Programming the Firmware Update Program

Follow the procedure in 4.3 Debugging the Project to program the firmware update program in the default area of the start-up program protection area and the constants data area. Alternatively, start in boot mode and use the Renesas Flash Programmer to program the firmware update program in the default area of the start-up program protection area and the constants data area.

For details on using the Renesas Flash Programmer, refer to the user’s manual for the Renesas Flash Programmer.

5.2.2 Programming the Firmware Program

The following procedure describes the flow of operation to program the firmware program using the firmware update program.

1. The firmware update program is launched in single-chip mode. The firmware update program starts up the SCI and outputs the menu displayed in the terminal software on the host PC.

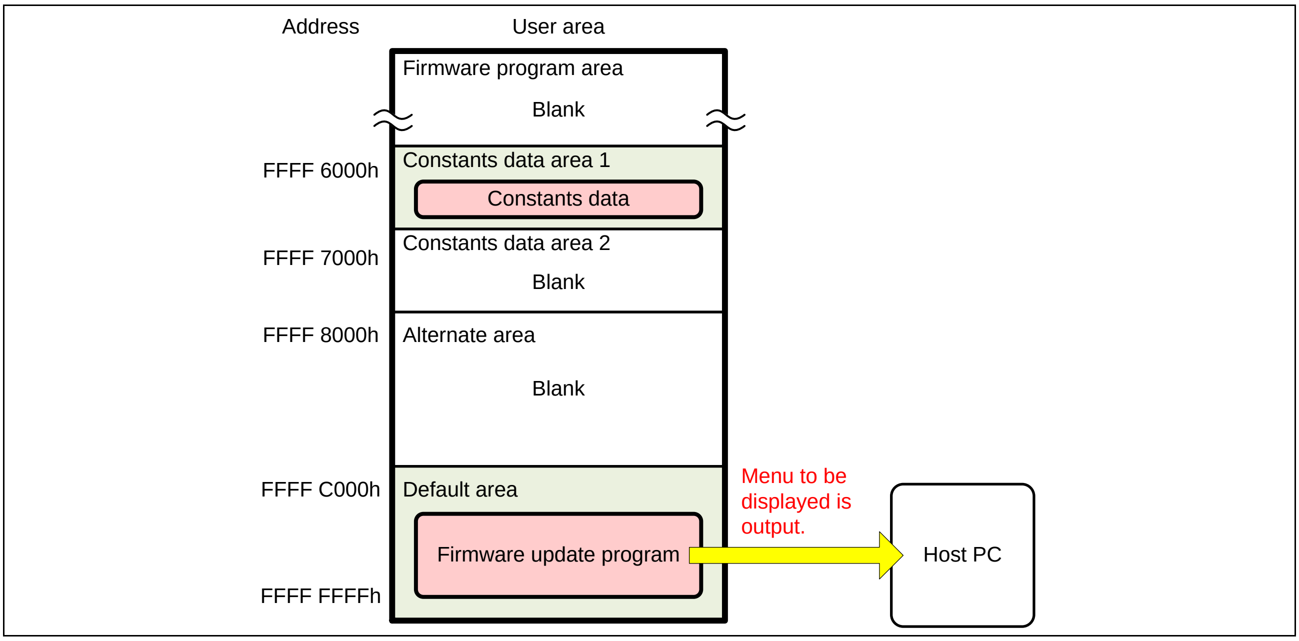


Figure 5.2 Launching the Firmware Update Program (for RX231 Group)

2. The firmware program update command is sent from the terminal software to program the firmware program. The firmware update program places the flash memory rewrite processing into the on-chip RAM so that programming the code flash memory is performed from the on-chip RAM.

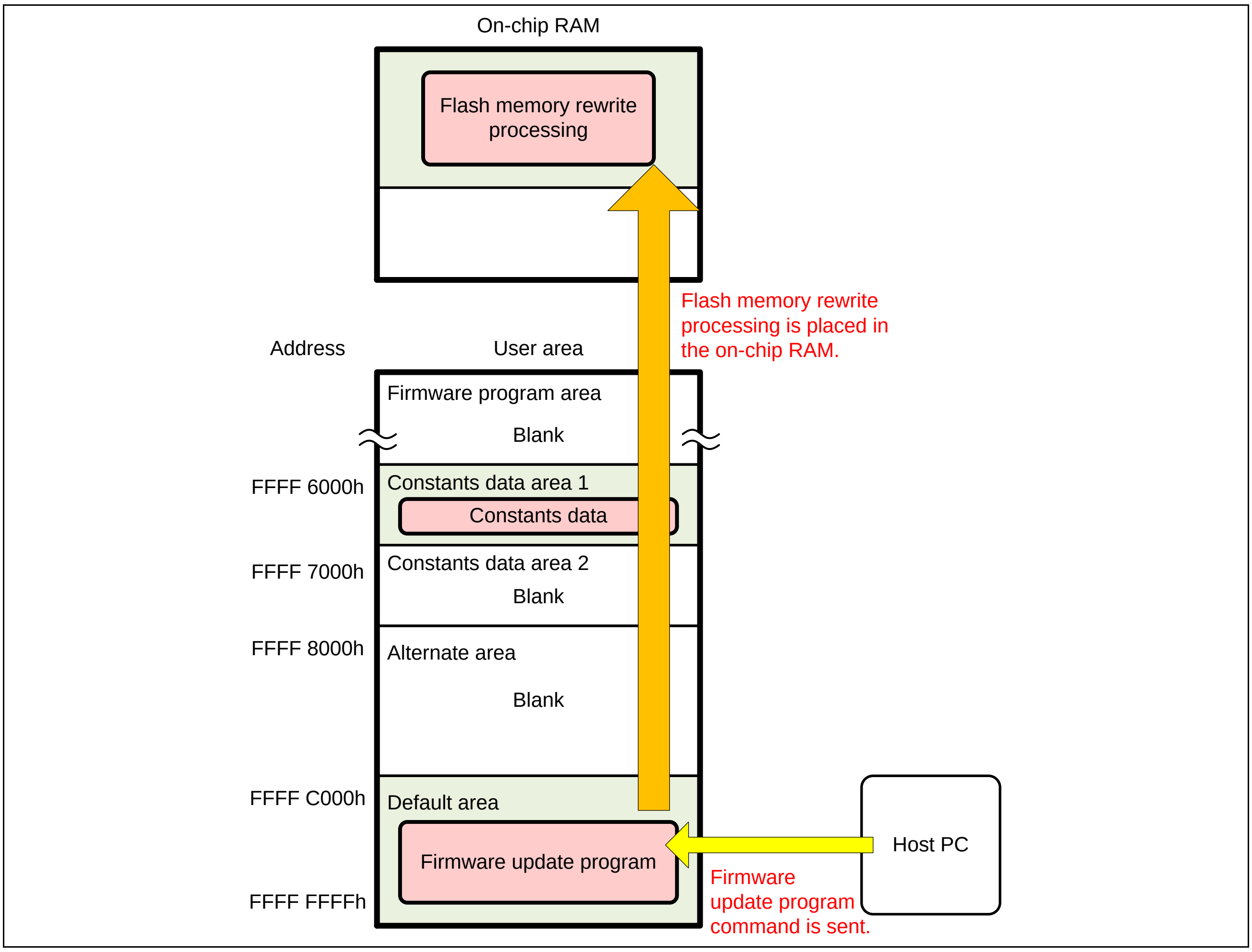


Figure 5.3 Sending the Firmware Program Update Command (for RX231 Group)

3. The firmware update program branches to flash memory rewrite processing in the on-chip RAM and erases the alternate area and the firmware program area. After erasing the code flash memory, it returns to the firmware update program in the default area.

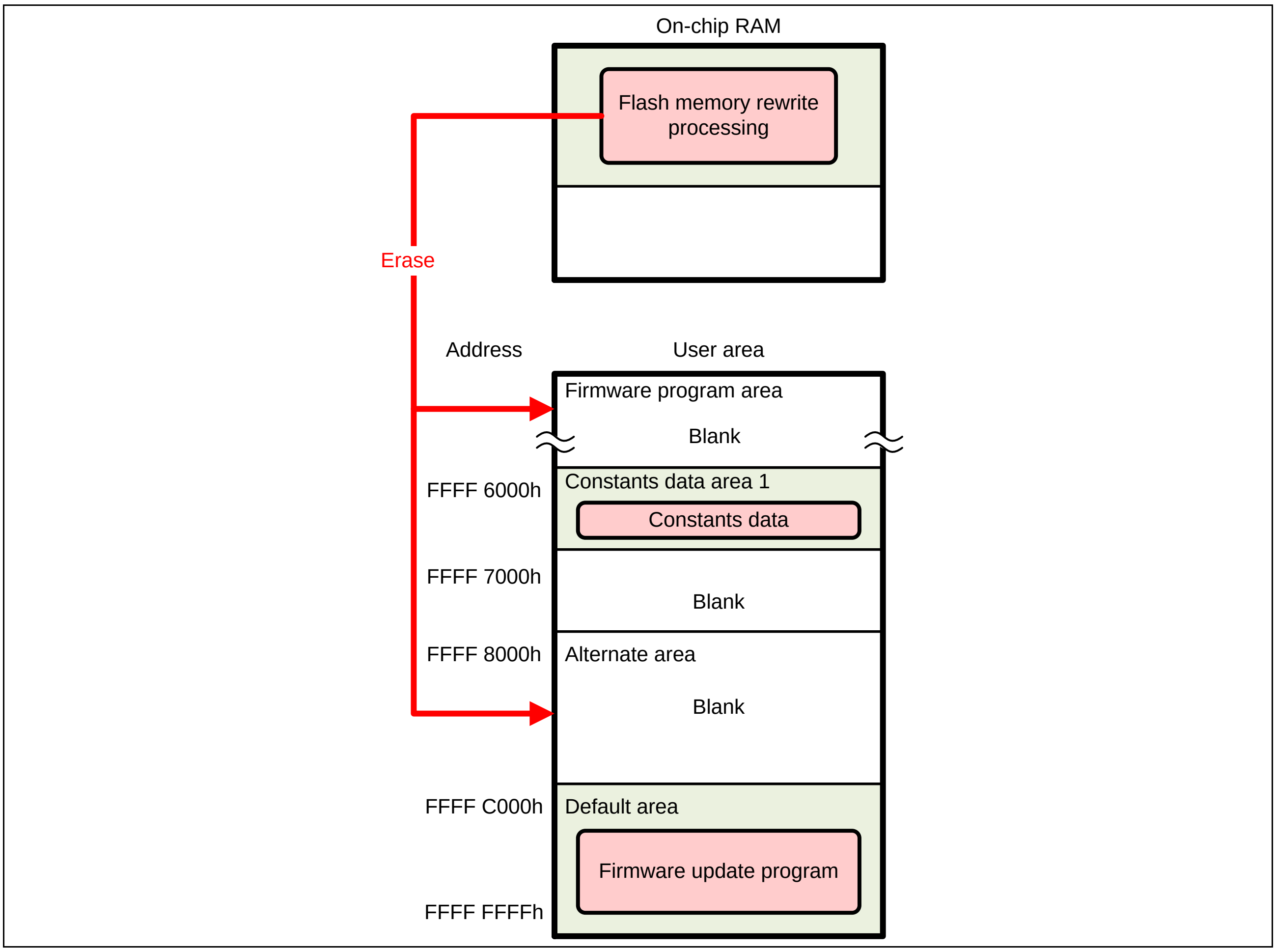


Figure 5.4 Erasing the Code Flash Memory (for RX231 Group)

4. The firmware program is transmitted using the terminal software. The firmware update program analyzes the received data and stores the data for programming the code flash memory into the write buffer of the on-chip RAM.

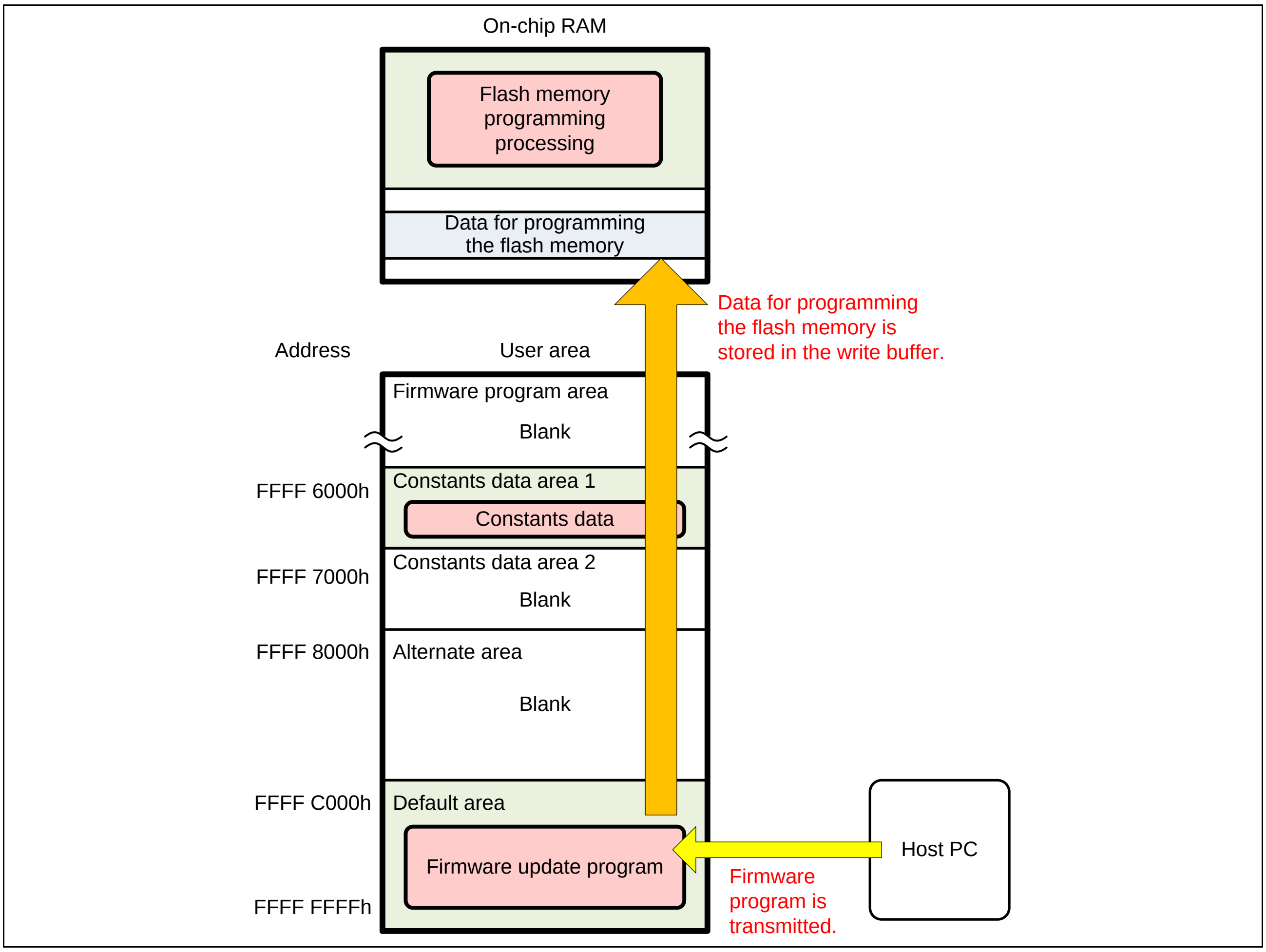


Figure 5.5 Transmitting the Firmware Program (for RX231 Group)

5. When the write buffer in the on-chip RAM becomes full with the data, the firmware update program branches to flash memory rewrite processing in the on-chip RAM. Flash memory rewrite processing temporarily switches between the default area and the alternate area of the start-up program protection according to the setting of the flash initial setting register (FISR). Then it programs the data in the write buffer to the code flash memory. After the data has been programmed, the start-up program protection area are switched back to the original.

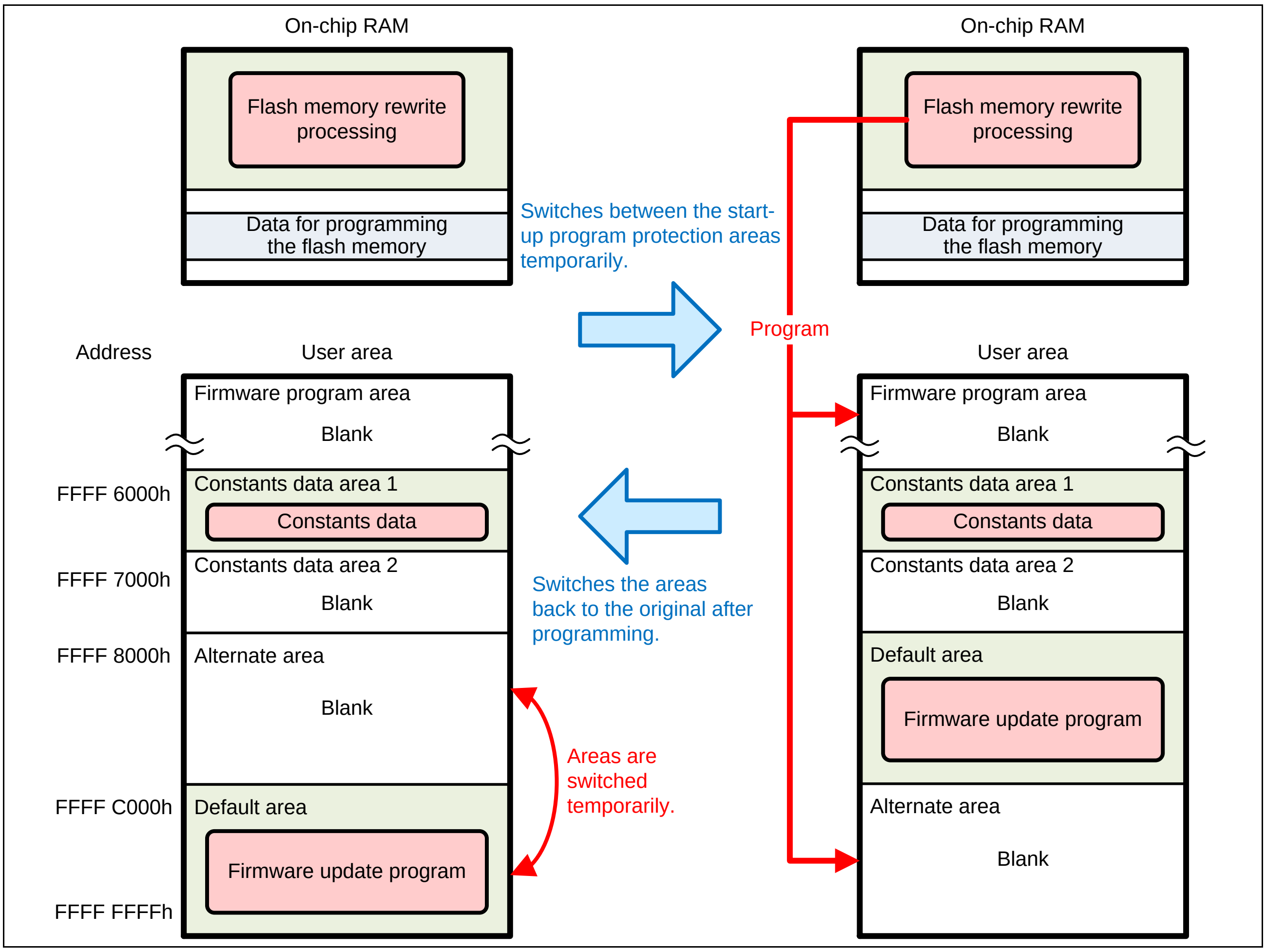


Figure 5.6 Programming the Firmware Program (for RX231 Group)

6. Steps 4 and 5 are repeated until all firmware program data are programmed.
7. After the firmware program is programmed, the firmware update program branches to firmware start-up processing. The firmware update program switches between the default area and the alternate area of the start-up program protection permanently according to the extra area setting, and then executes a software reset. The firmware program is launched.

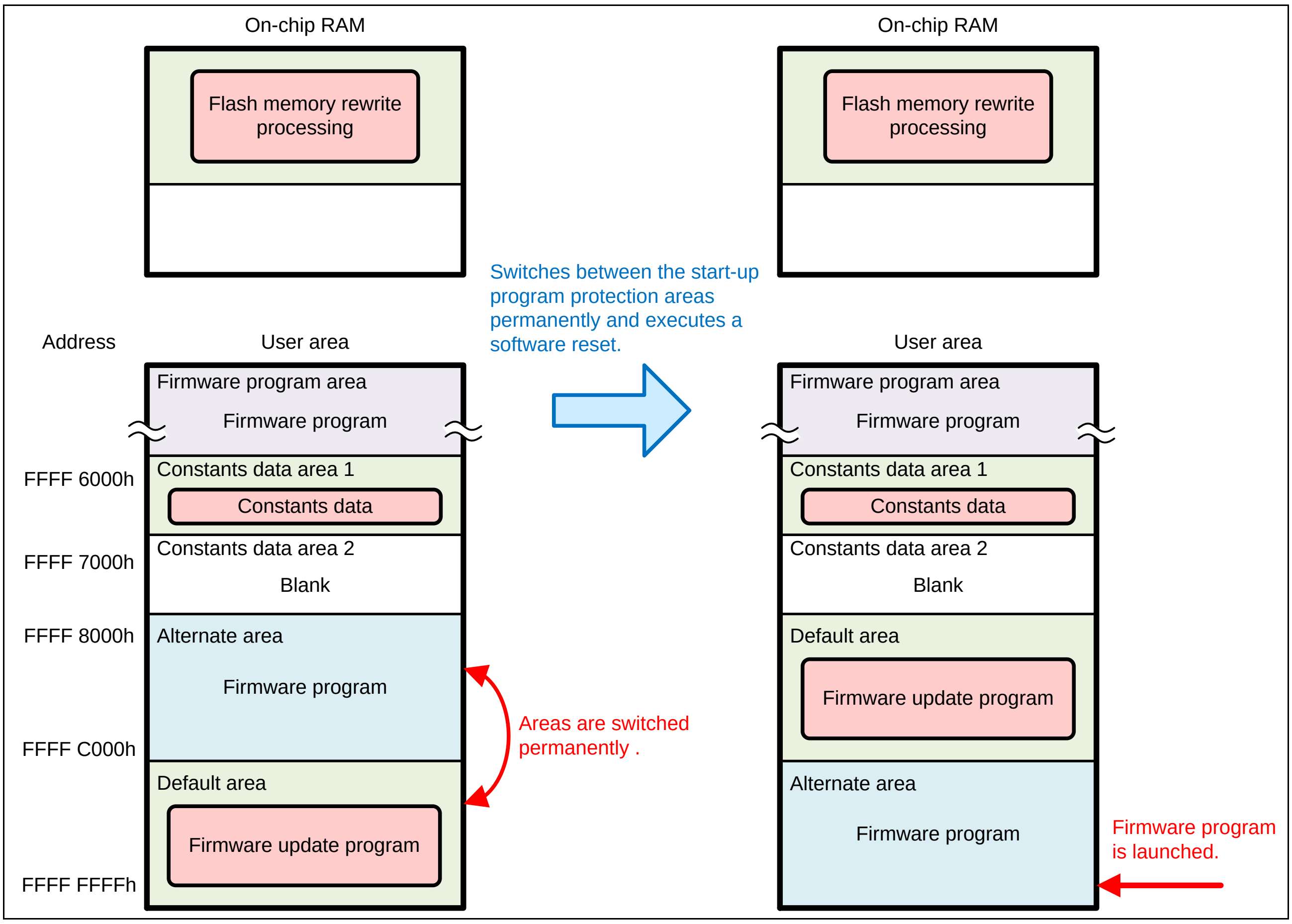


Figure 5.7 Software Reset and Launching the Firmware Program (for RX231 Group)

5.2.3 Updating the Firmware Program

This section describes the flow of operation to update the firmware program using the firmware update program.

1. The default area and the alternate area of the start-up program protection are switched permanently according to the extra area setting by the firmware program which has not yet been updated. Then the device is reset.

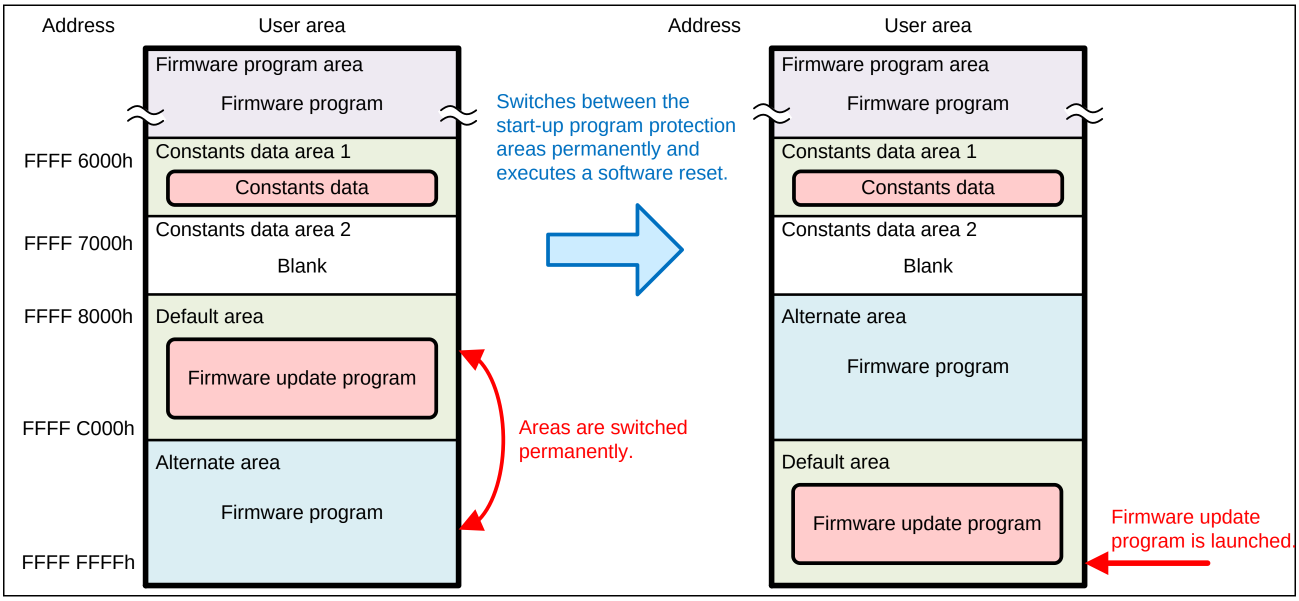


Figure 5.8 Operation of the Firmware Program Before Being Updated (for RX231 Group)

2. The firmware update program is launched. The firmware update program starts up the SCI and outputs the menu displayed in the terminal software on the host PC.

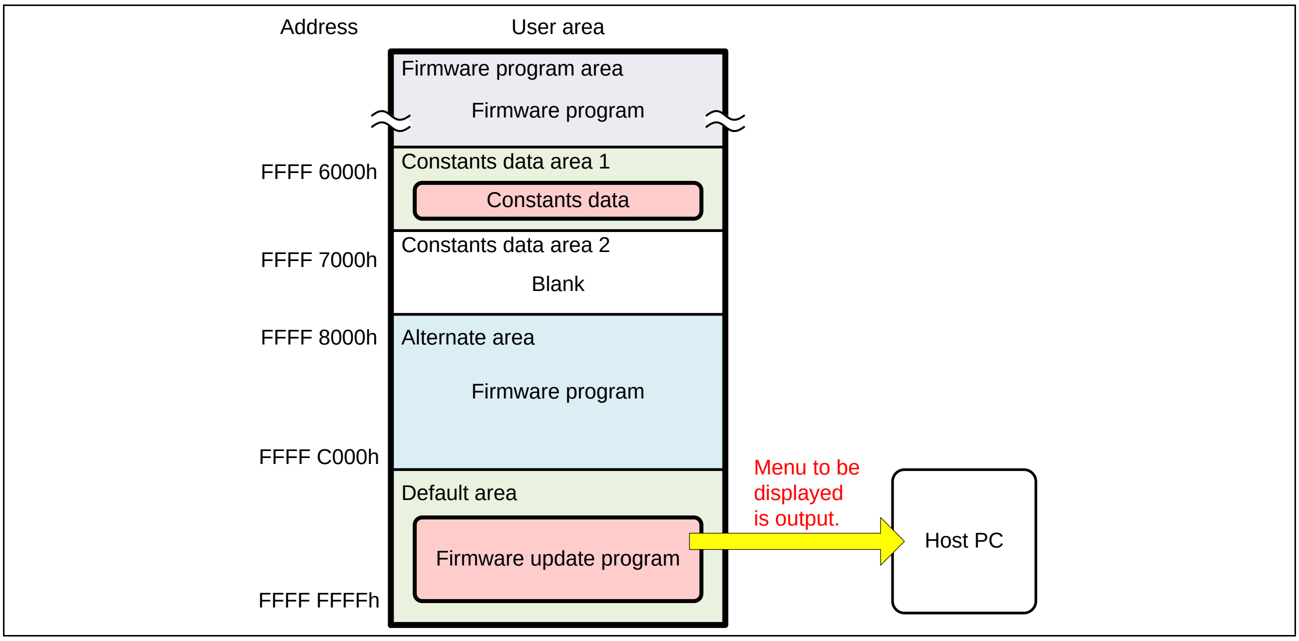


Figure 5.9 Launching the Firmware Update Program (for RX231 Group)

3. The firmware program update command is sent from the terminal software to update the firmware program. The firmware program update program places the flash memory rewrite processing into the on-chip RAM so that programming the code flash memory is performed from the on-chip RAM.

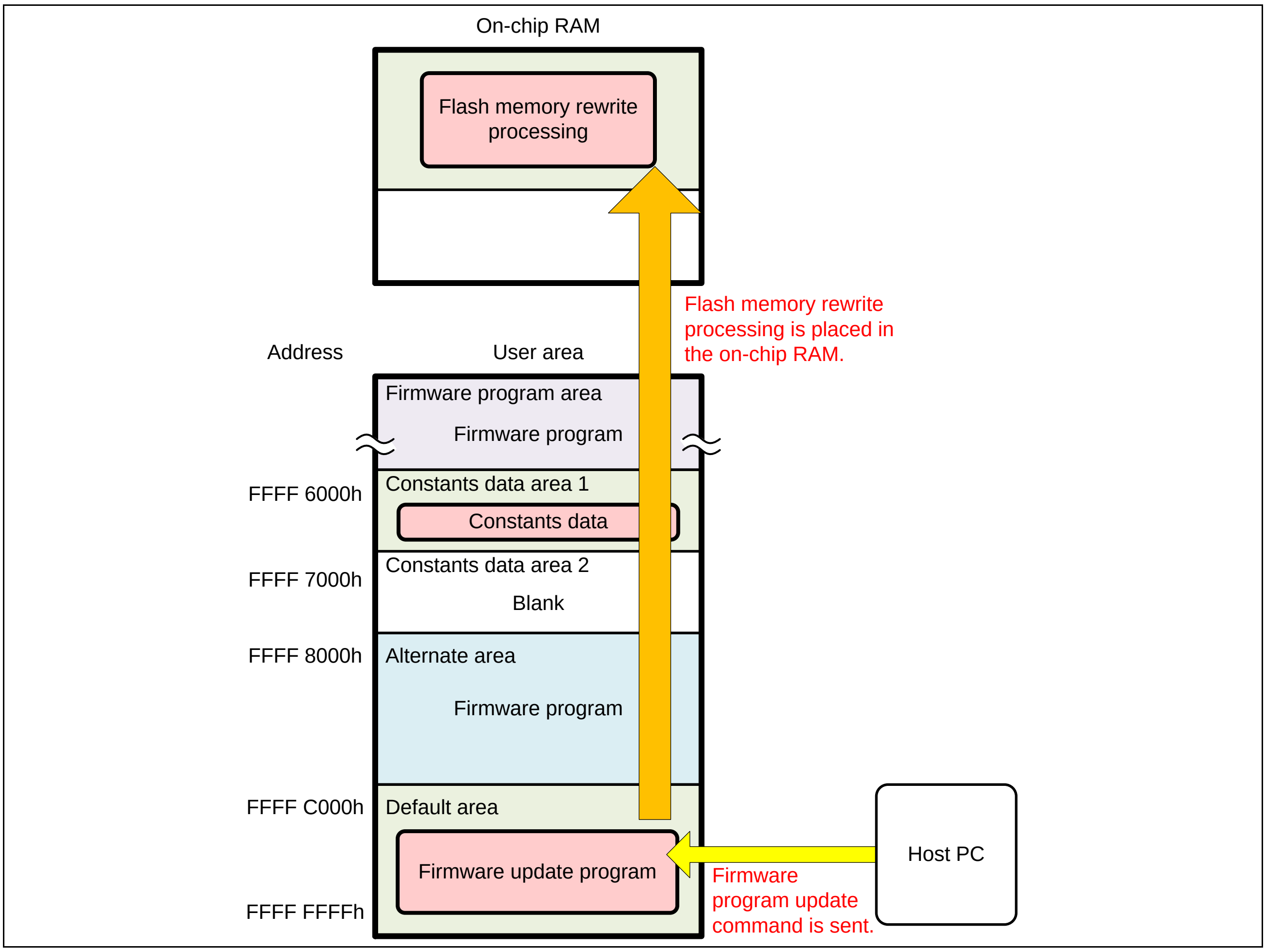


Figure 5.10 Sending the Firmware Program Update Command (for RX231 Group)

4. The firmware update program branches to flash memory rewrite processing in the on-chip RAM and erases the alternate area and the firmware program area. After erasing the code flash memory, it returns to the firmware update program in the default area.

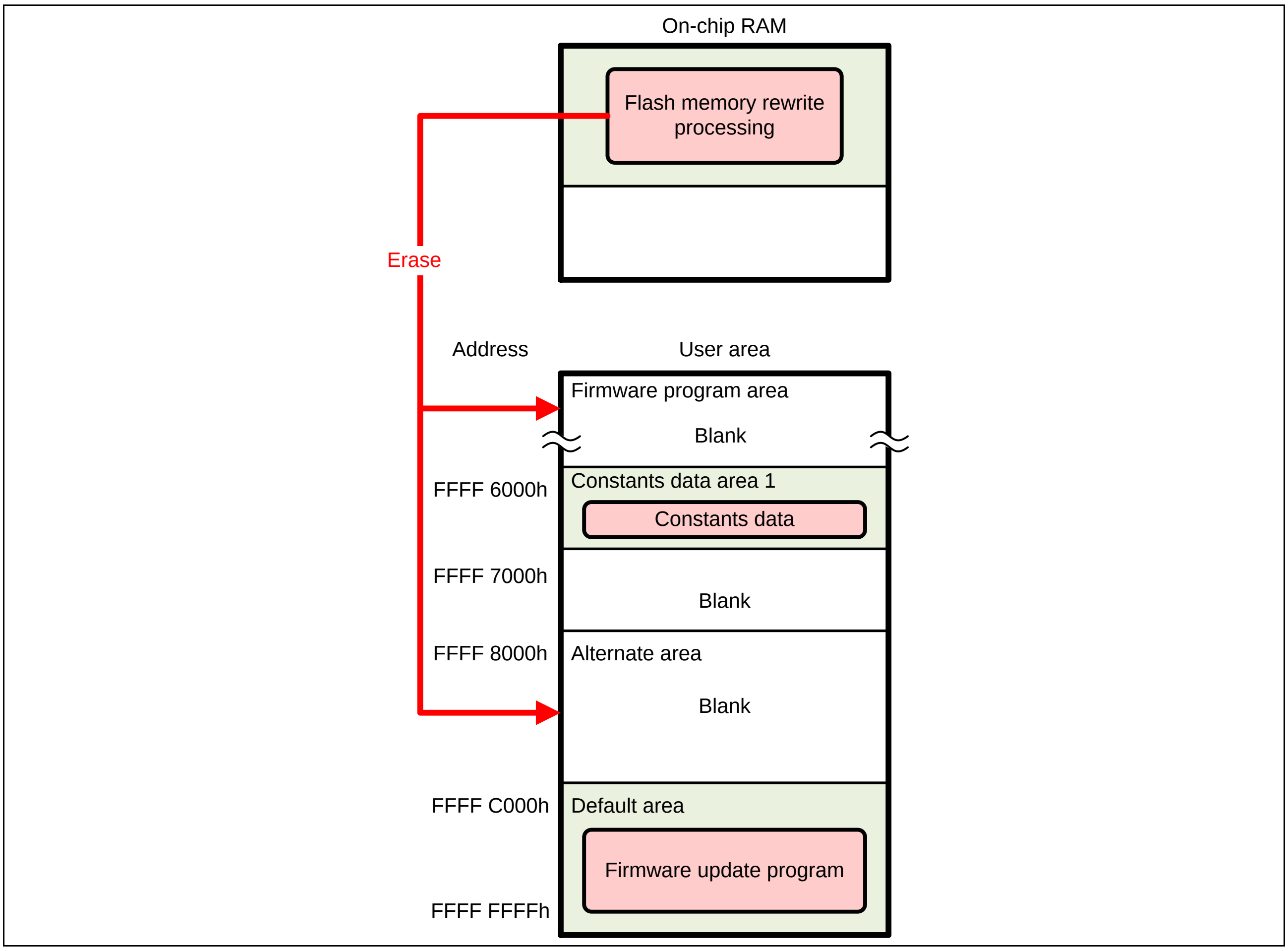


Figure 5.11 Erasing the Code Flash Memory (for RX231 Group)

5. The new firmware program is transmitted using the terminal software. The firmware update program analyzes the received data and stores the data for programming the code flash memory into the write buffer of the on-chip RAM.

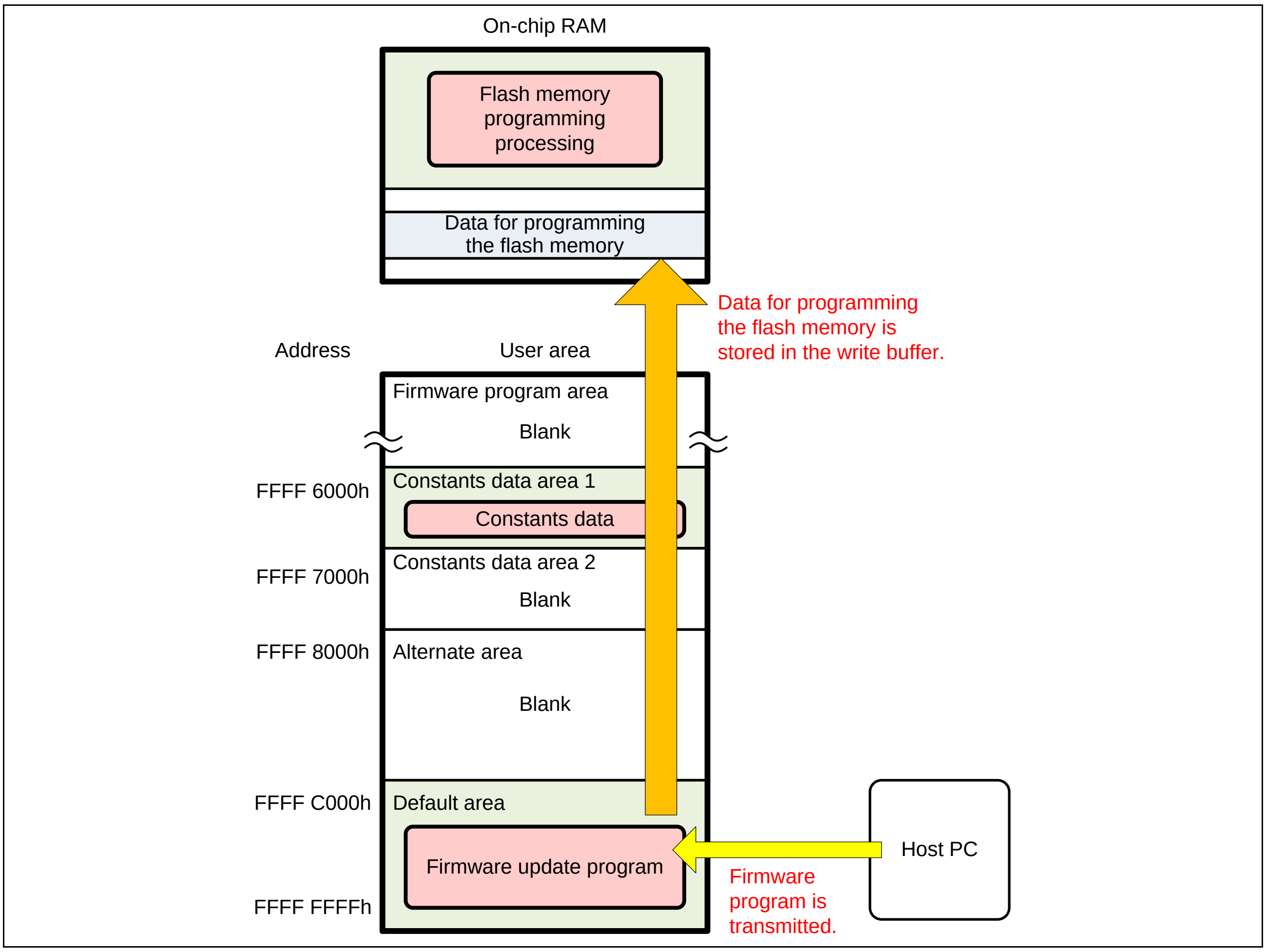


Figure 5.12 Transmitting the Firmware Program (for RX231 Group)

6. When the write buffer in the on-chip RAM becomes full with the data, the firmware update program branches to flash memory rewrite processing in the on-chip RAM. Flash memory rewrite processing temporarily switches between the default area and the alternate area of the start-up program protection according to the setting of the flash initial setting register (FISR). Then the data stored in the write buffer is programmed into the code flash memory. After the programming has been completed, the start-up program protection areas are switched back to the original.

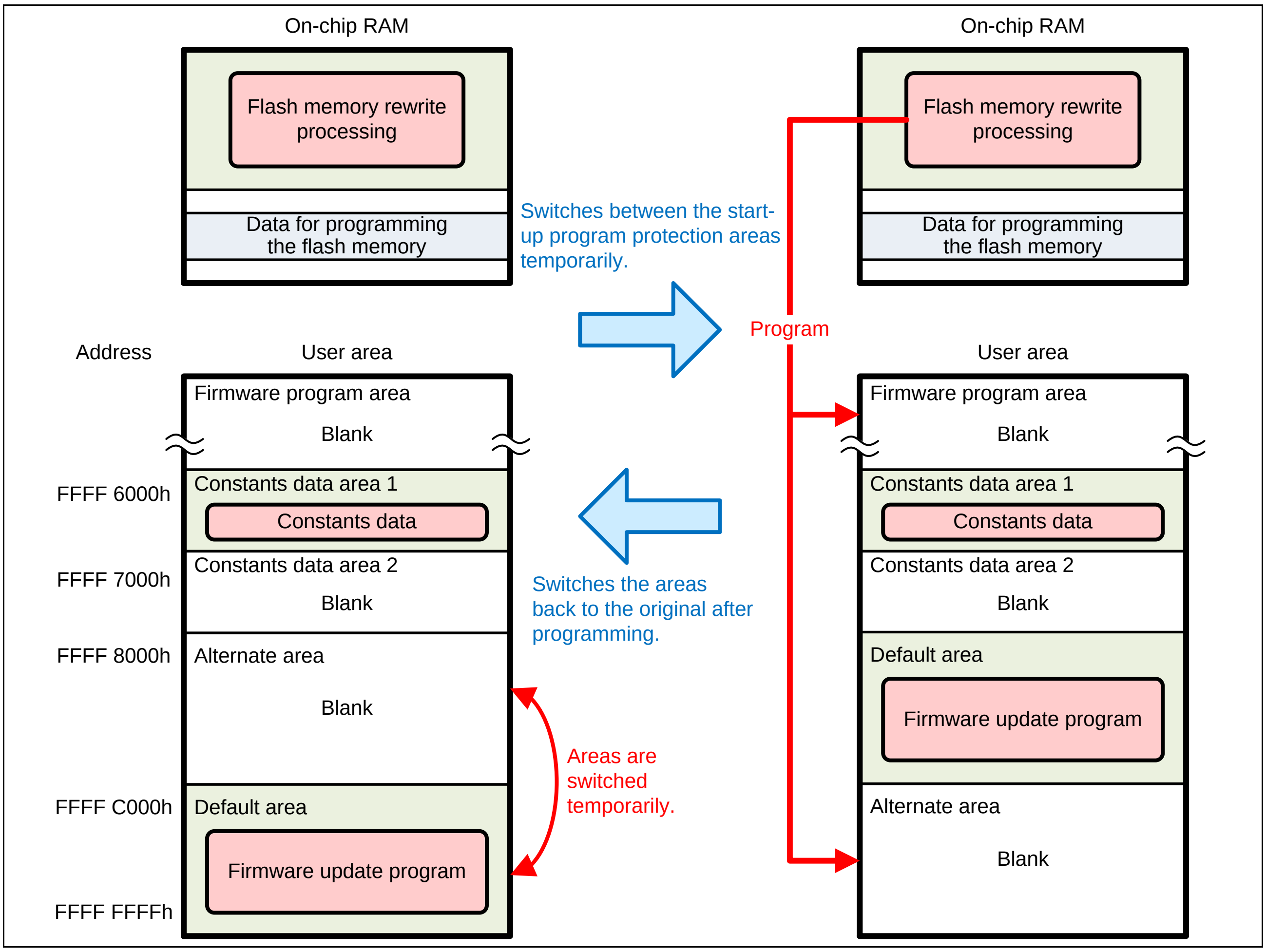


Figure 5.13 Programming the Firmware Program (for RX231 Group)

7. Steps 5 and 6 are repeated until all new firmware program data are programmed.
8. After the new firmware program is programmed, the firmware update program branches to firmware start-up processing. The firmware update program switches between the default area and the alternate area of the start-up program protection permanently according to the extra area setting, and then executes a software reset. The updated firmware program is launched.

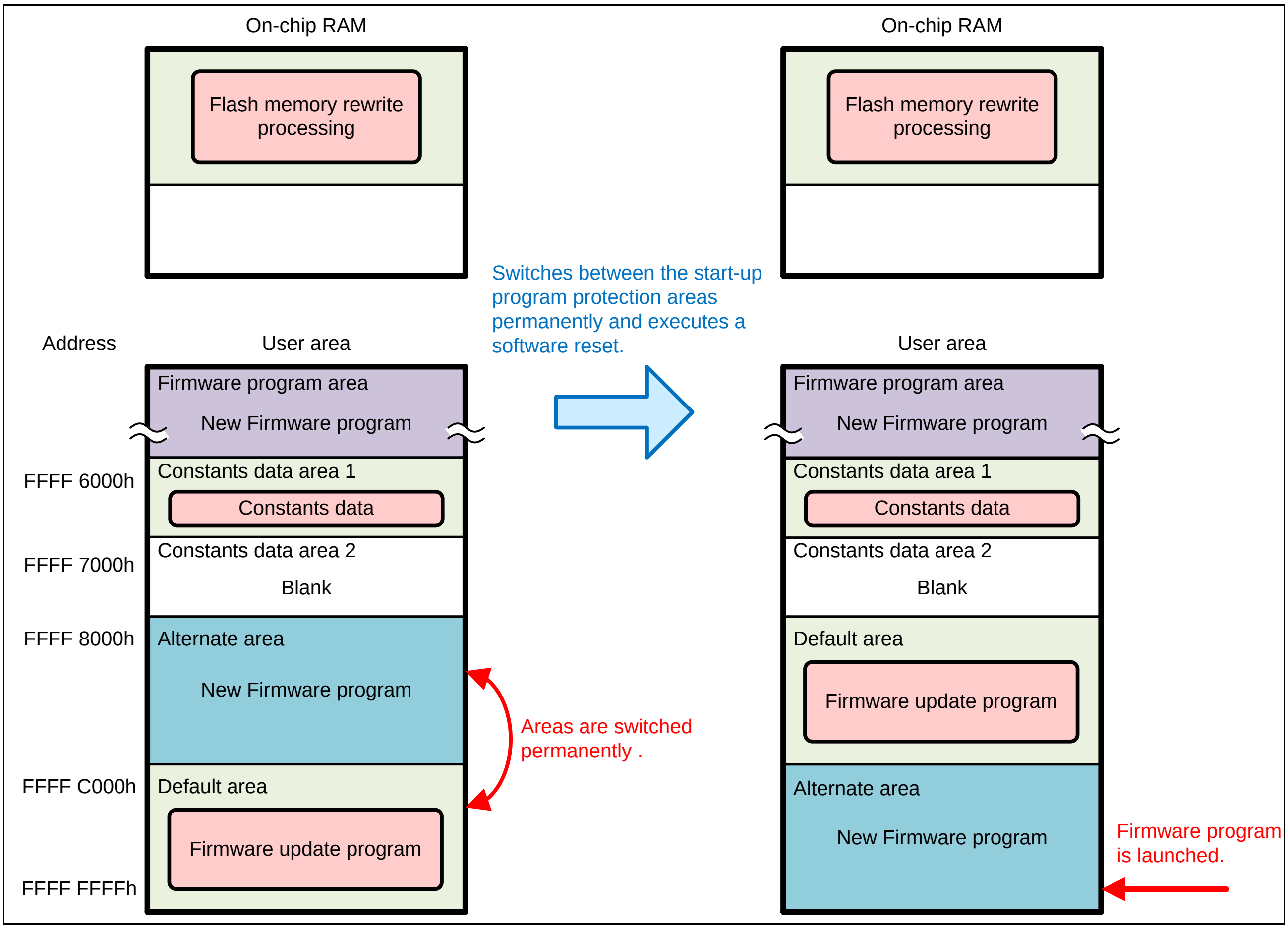


Figure 5.14 Software Reset and Launching the Updated Firmware Program (for RX231 Group)

5.2.4 Updating the Firmware Update Program

This section describes the flow of operation to update the firmware update program using the firmware update program. Note that the firmware program on the code flash memory is erased when updating the firmware update program. Thus the firmware program must be programmed after updating the firmware update program.

1. After the default area and the alternate area of the startup program protection have been switched permanently according to the extra area setting by the firmware program, the device is reset.

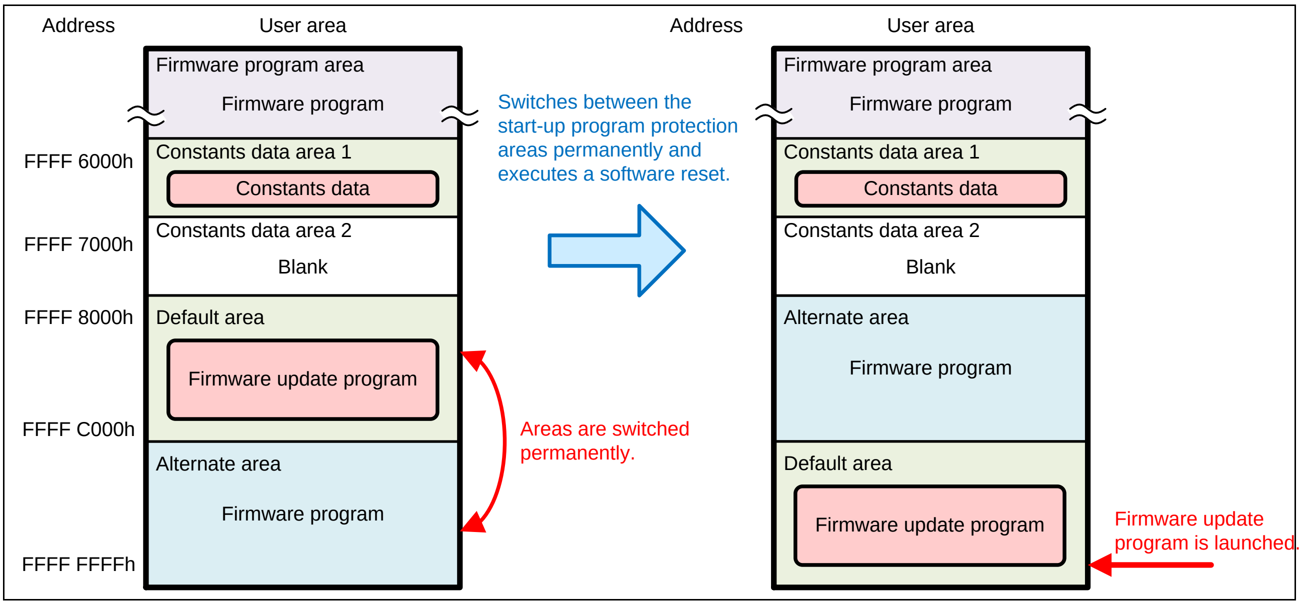


Figure 5.15 Firmware Program Operation (for RX231 Group)

2. The firmware update program is launched. The firmware update program starts up the SCI and outputs the menu displayed in the terminal software on the host PC.

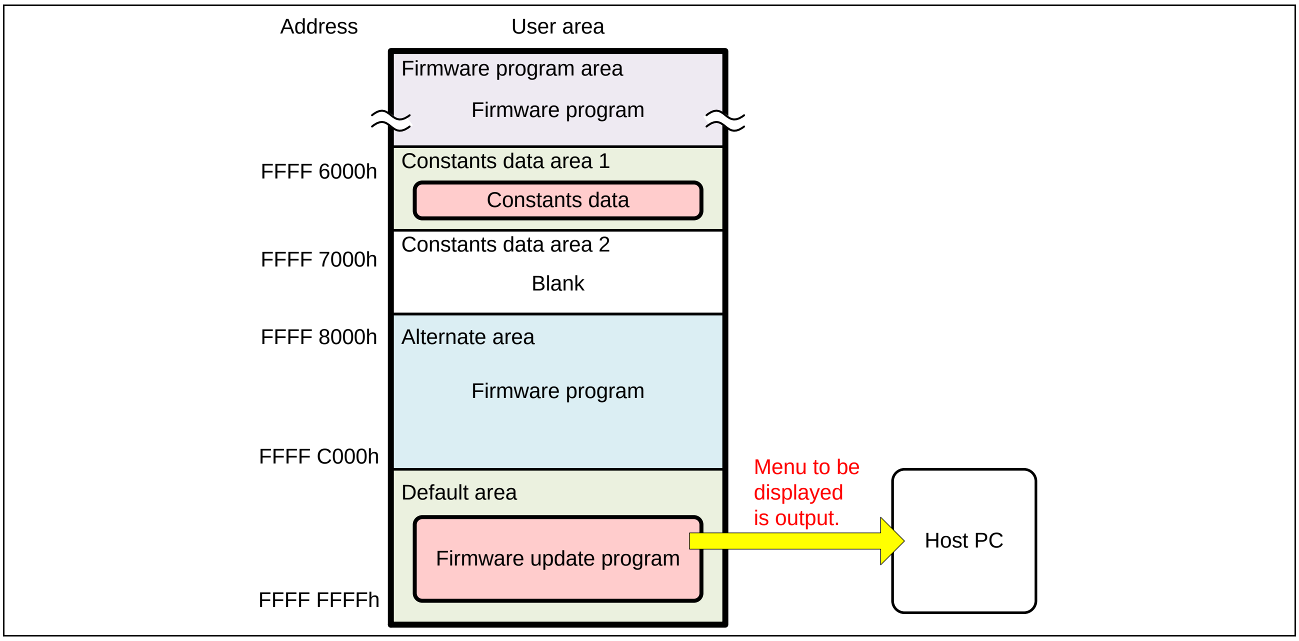


Figure 5.16 Launching the Firmware Update Program (for RX231 Group)

3. The firmware update program update command is sent from the terminal software to update the firmware update program. The firmware update program places the flash memory rewrite processing in the on-chip RAM so that programming the code flash memory is performed from the on-chip RAM.

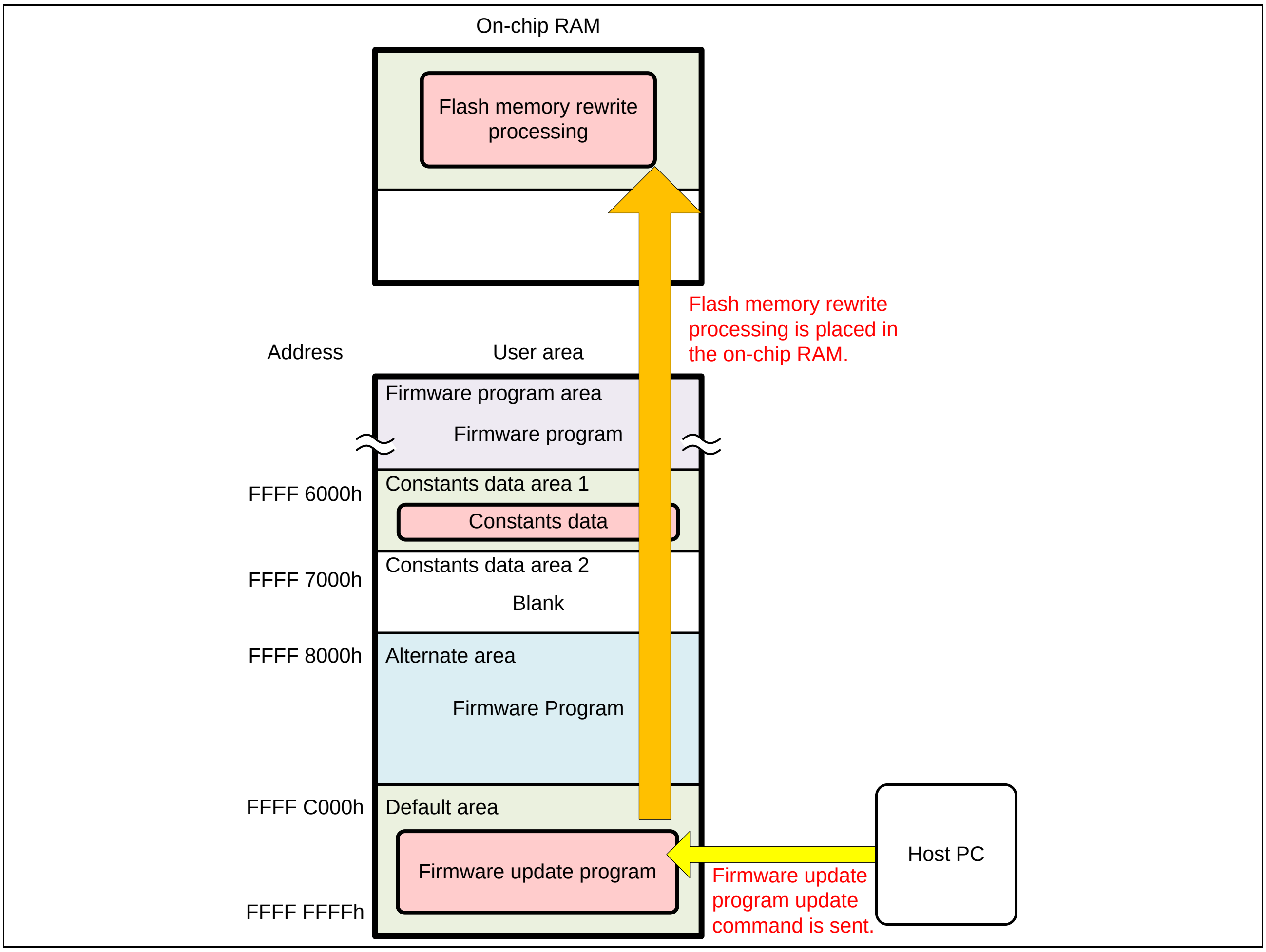


Figure 5.17 Sending the Firmware update program Update Command (for RX231 Group)

4. The firmware update program branches to flash memory rewrite processing in the on-chip RAM and erases the alternate area and the constants data area 2. After erasing the code flash memory, it returns to the firmware update program in the default area.

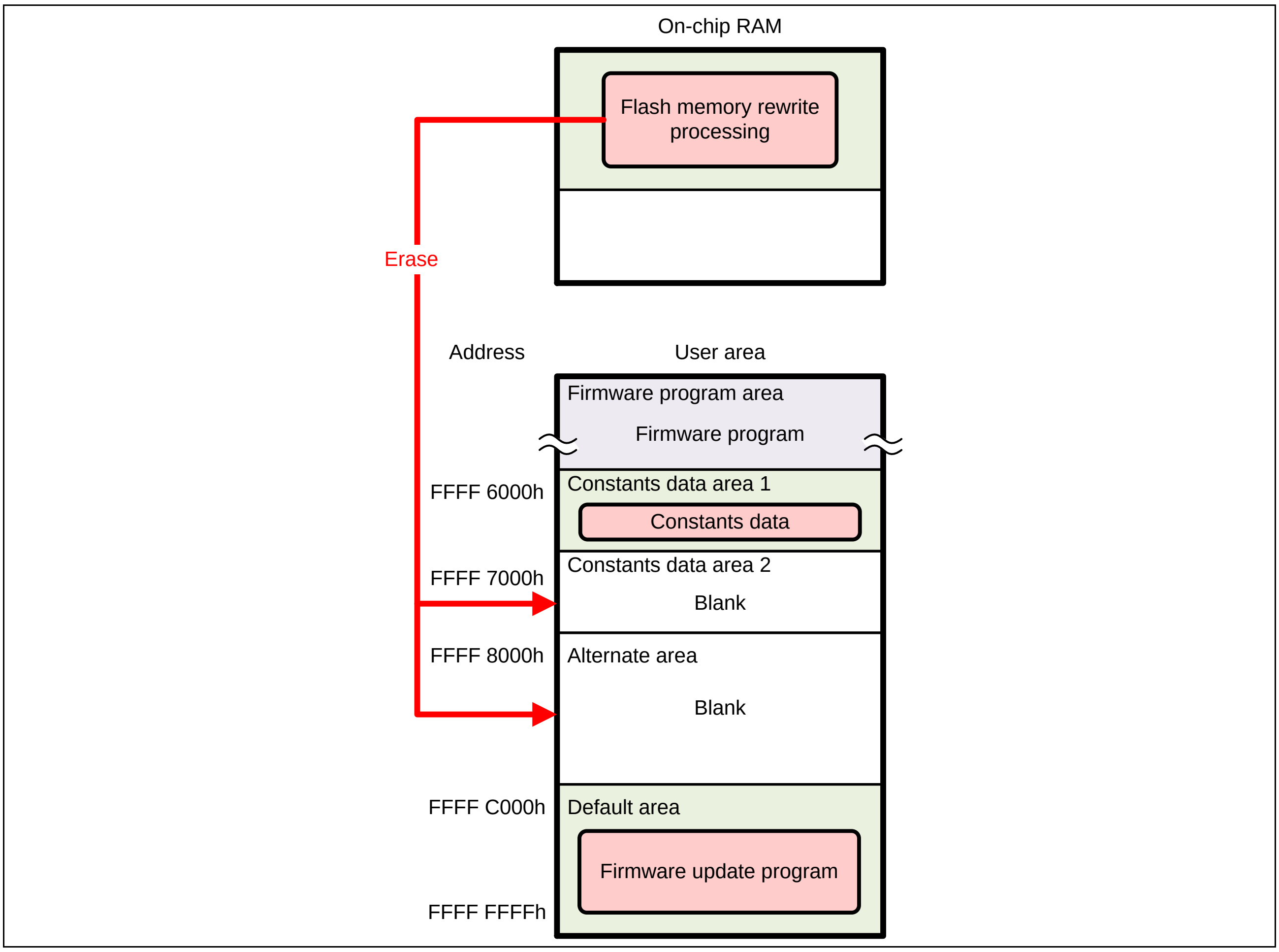


Figure 5.18 Erasing the Code Flash Memory (for RX231 Group)

5. The new firmware update program is transmitted by the terminal software. The firmware update program before being updated analyzes the received data and stores the data for programming the code flash memory into the write buffer of the on-chip RAM.

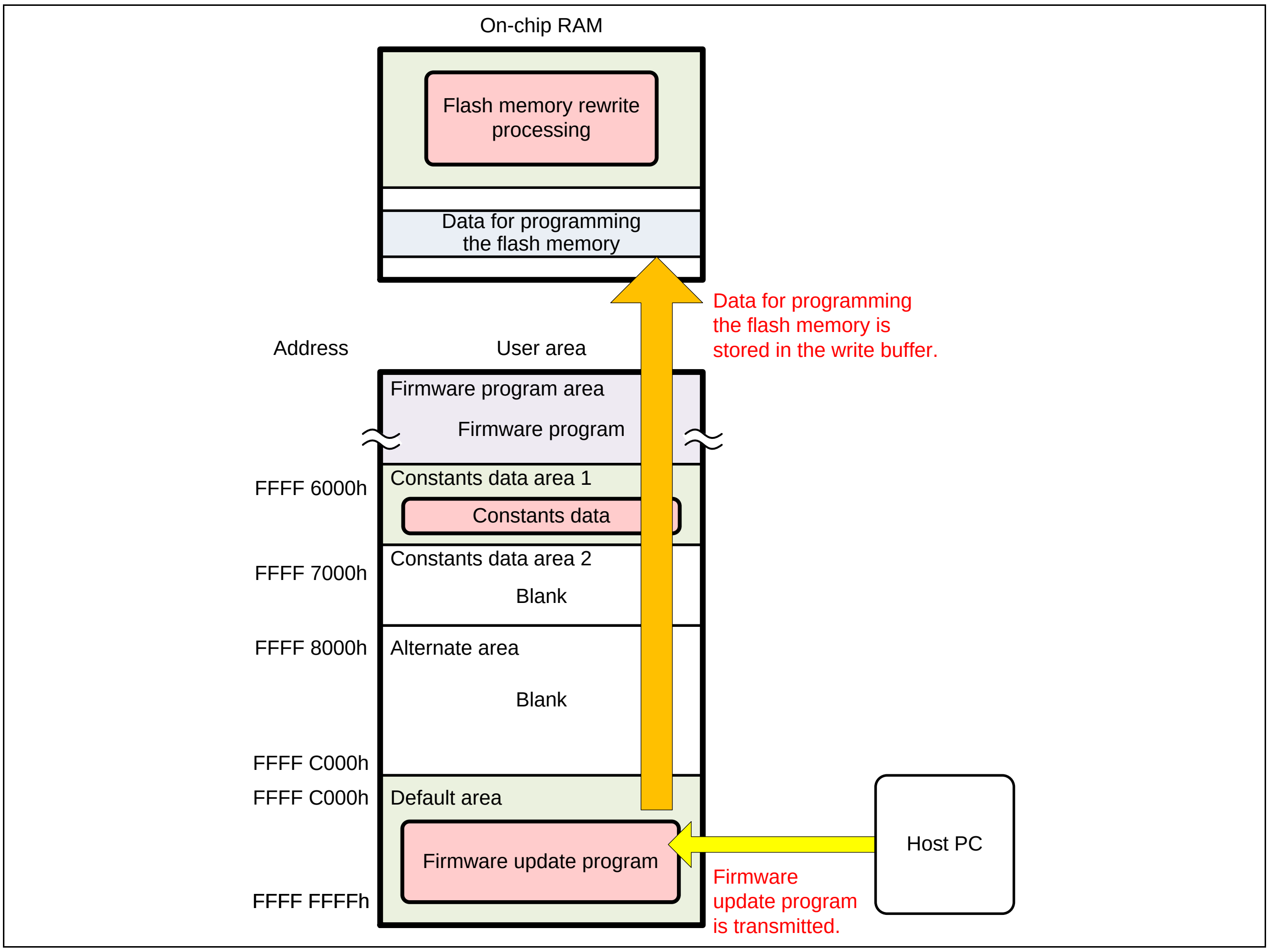


Figure 5.19 Transmitting the Firmware Update Program (for RX231 Group)

6. When the write buffer becomes full with the data in the on-chip RAM, the firmware update program branches to flash memory rewrite processing in the on-chip RAM. Flash memory rewrite processing temporarily switches between the default area and the alternate area of the start-up program protection according to the setting of the flash initial setting register (FISR). Then the data stored in the write buffer is programmed into the code flash memory. After the programming has been completed, the start-up program protection areas are switched back to the original.

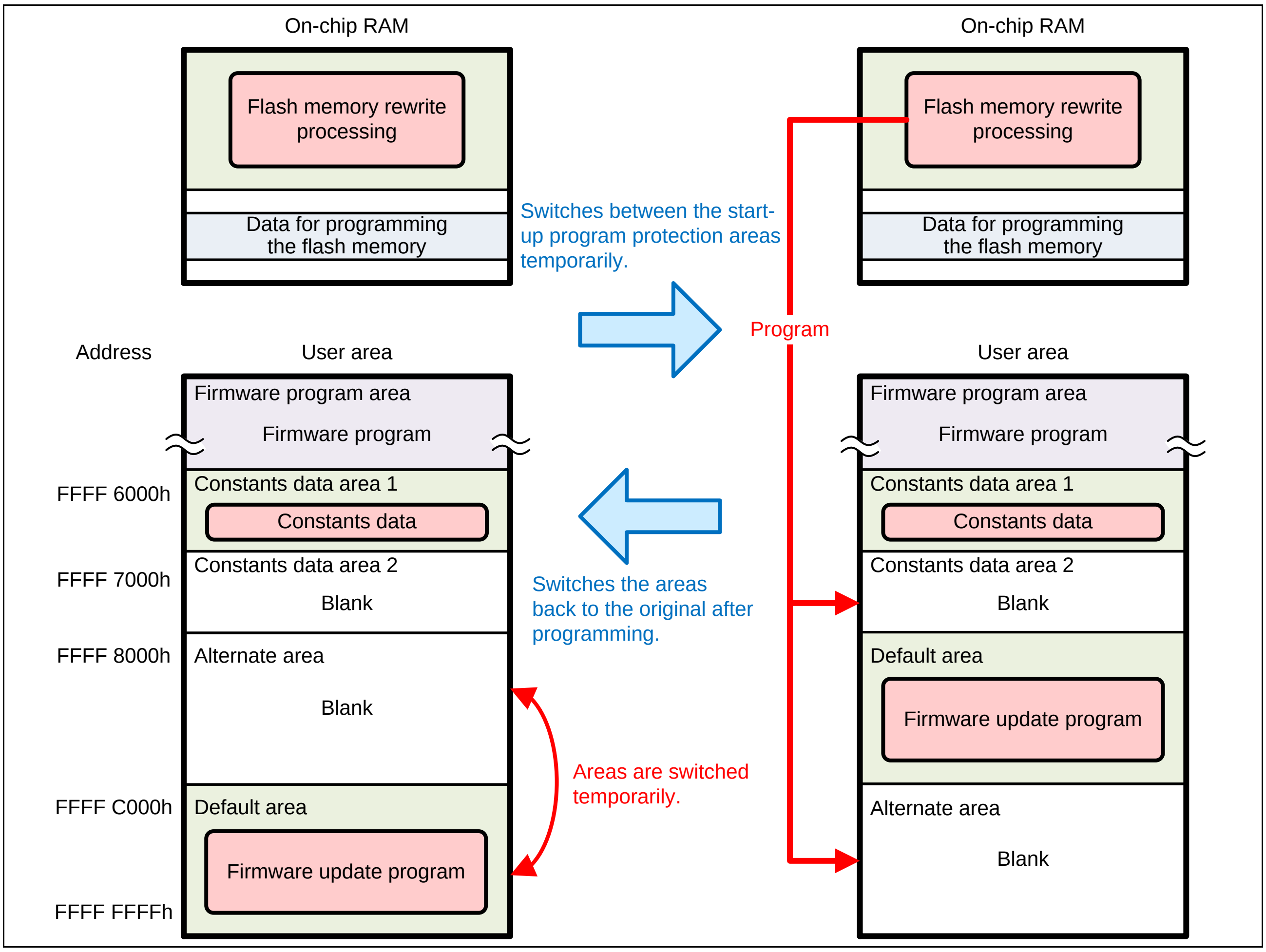


Figure 5.20 Programming the Firmware Update Program (for RX231 Group)

7. Steps 5 and 6 are repeated until all data of the new firmware update program are programmed.
8. After the new firmware update program is programmed, the version information of the new firmware update program is retrieved from the version information storage area 2 in the constants data area 2, and then programmed into the write complete information storage area 2.

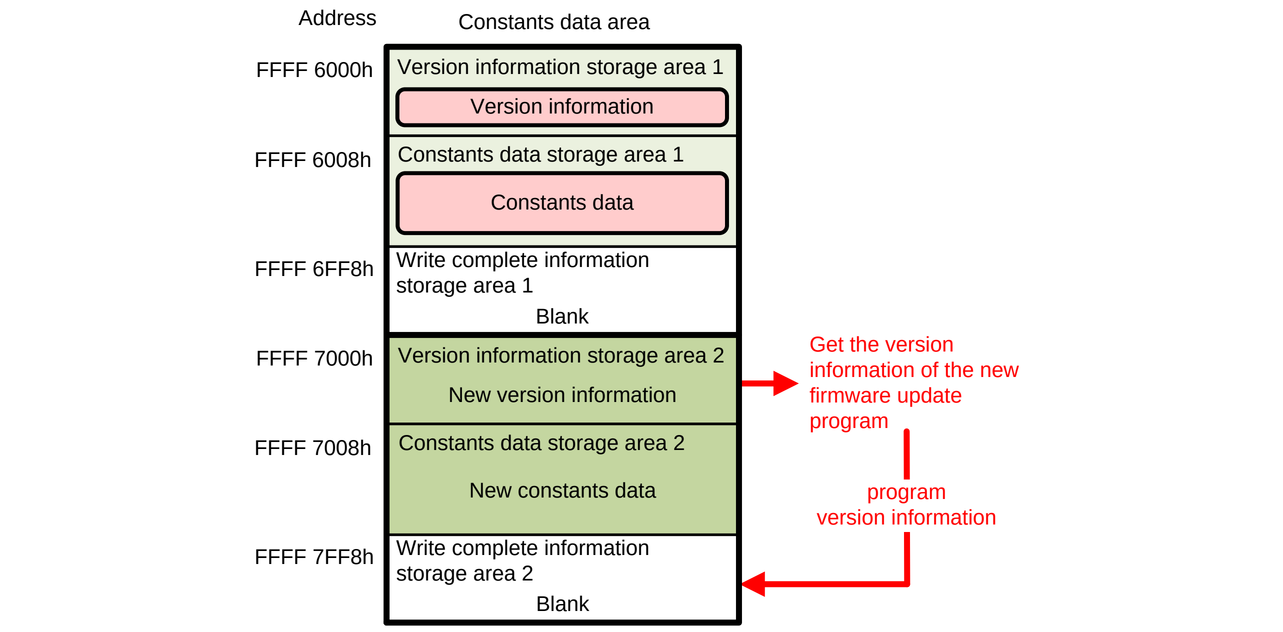

Address
Constants data area
FFFF 6000h
Version information storage area 1
Version information
FFFF 6008h
Constants data storage area 1
Constants data
FFFF 6FF8h
Write complete information storage area 1
Blank
FFFF 7000h
Version information storage area 2
New version information
FFFF 7008h
Constants data storage area 2
New constants data
FFFF 7FF8h
Write complete information storage area 2
Blank
Get the version information of the new firmware update program
program version information

Figure 5.21 Programming write complete information storage area (for RX231 group)

9. After the version information is programmed into the write complete information storage area, the firmware update program branches to firmware start-up processing. The firmware update program before being updated switches between the default area and the alternate area of the start-up program protection permanently according to the extra area setting, and then executes a software reset. The updated firmware update program is launched.

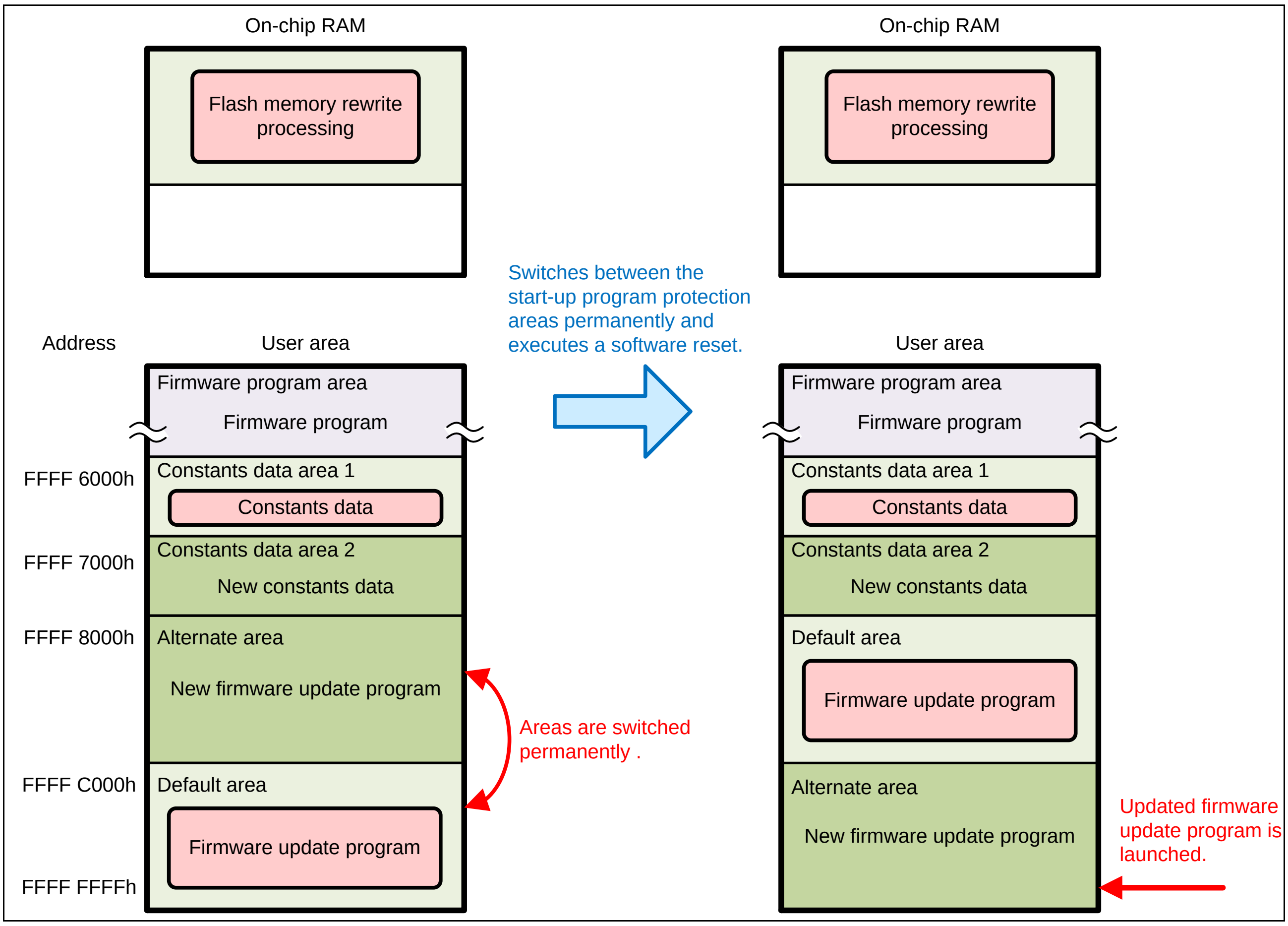


Figure 5.22 Software Reset and Launching the Updated Firmware Update Program (for RX231 Group)

5.2.5 Firmware Program Operation

This section describes flow of operation to confirm the firmware update program operation using the firmware program.

1. The firmware program is programmed as described in 5.2.2 Programming the Firmware.

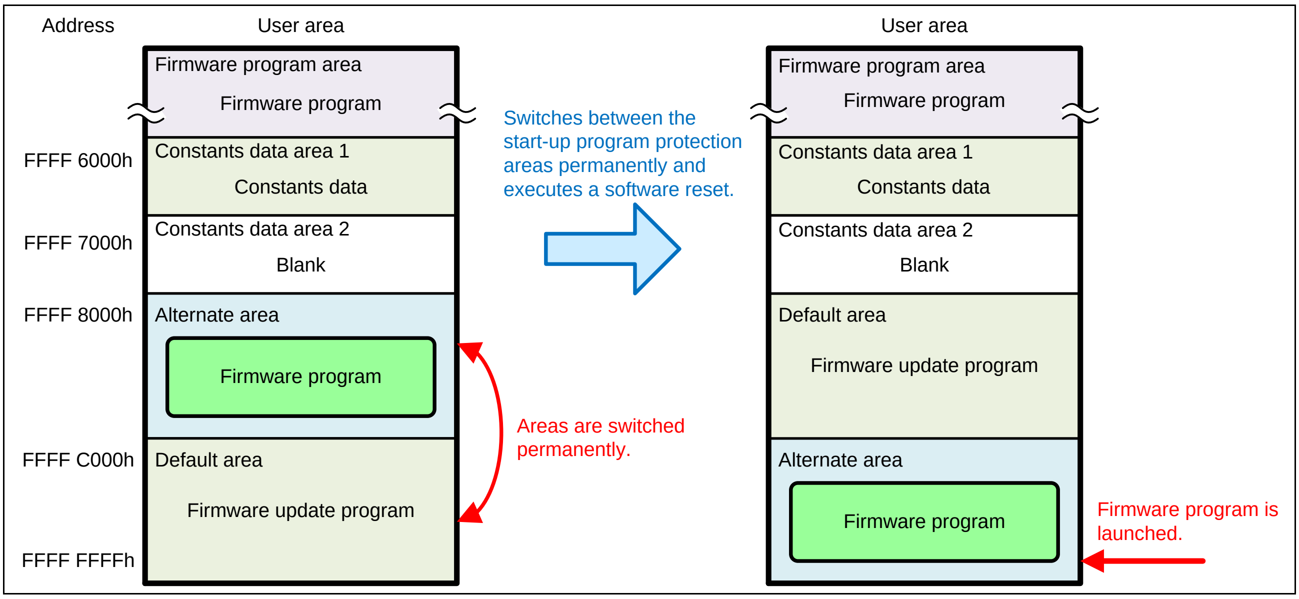


Figure 5.23 Programming and Launching the Firmware Program with the Firmware Update Program (for RX231 Group)

2. The firmware program is launched. The firmware program starts up the SCI and outputs the message to the terminal software on the host PC.

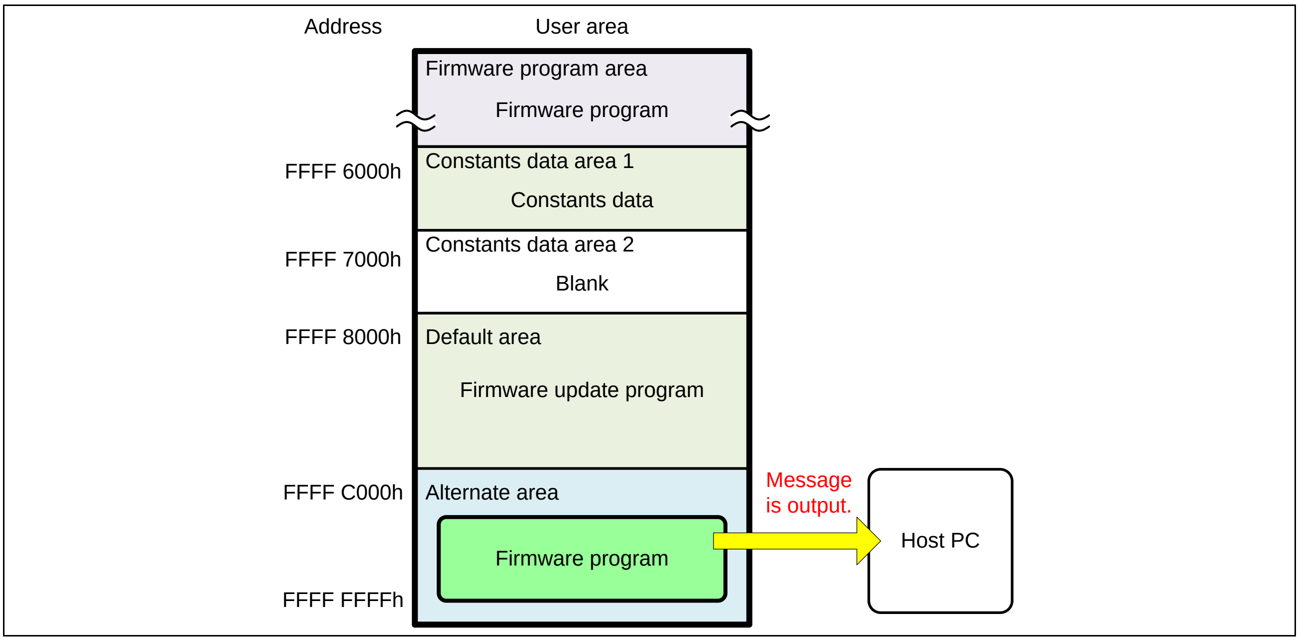


Figure 5.24 Outputting the Message (for RX231 Group)

3. The command is sent from the terminal software. The firmware program places processing for operation for the flash memory in the on-chip RAM so that switching the start-up program protection area and software reset are performed in the on-chip RAM.

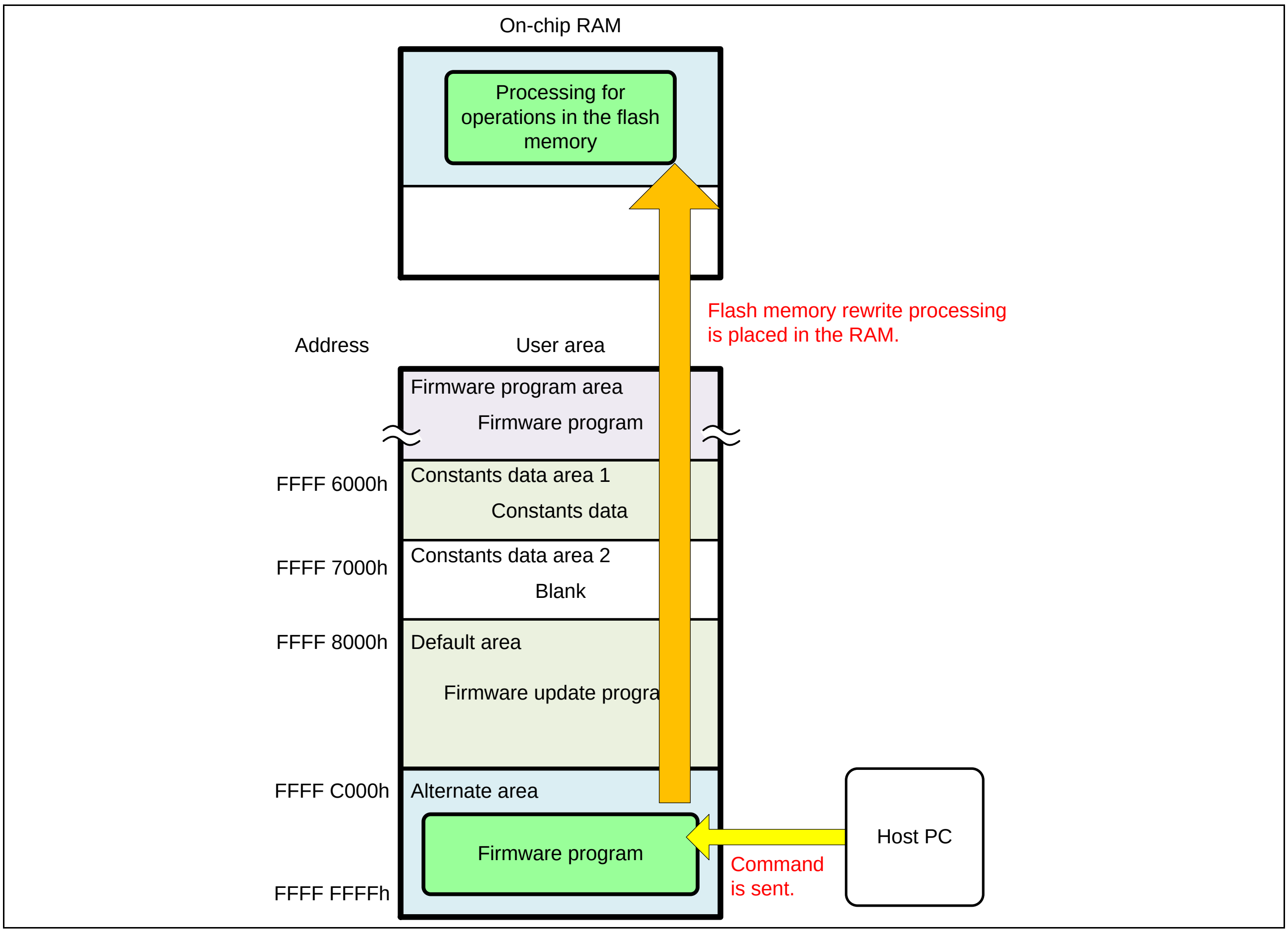


Figure 5.25 Receiving the Command (for RX231 Group)

4. The firmware program switches between the default area and the alternate area of the start-up program protection permanently according to the extra area setting, and then executes a software reset. The firmware update program is launched.

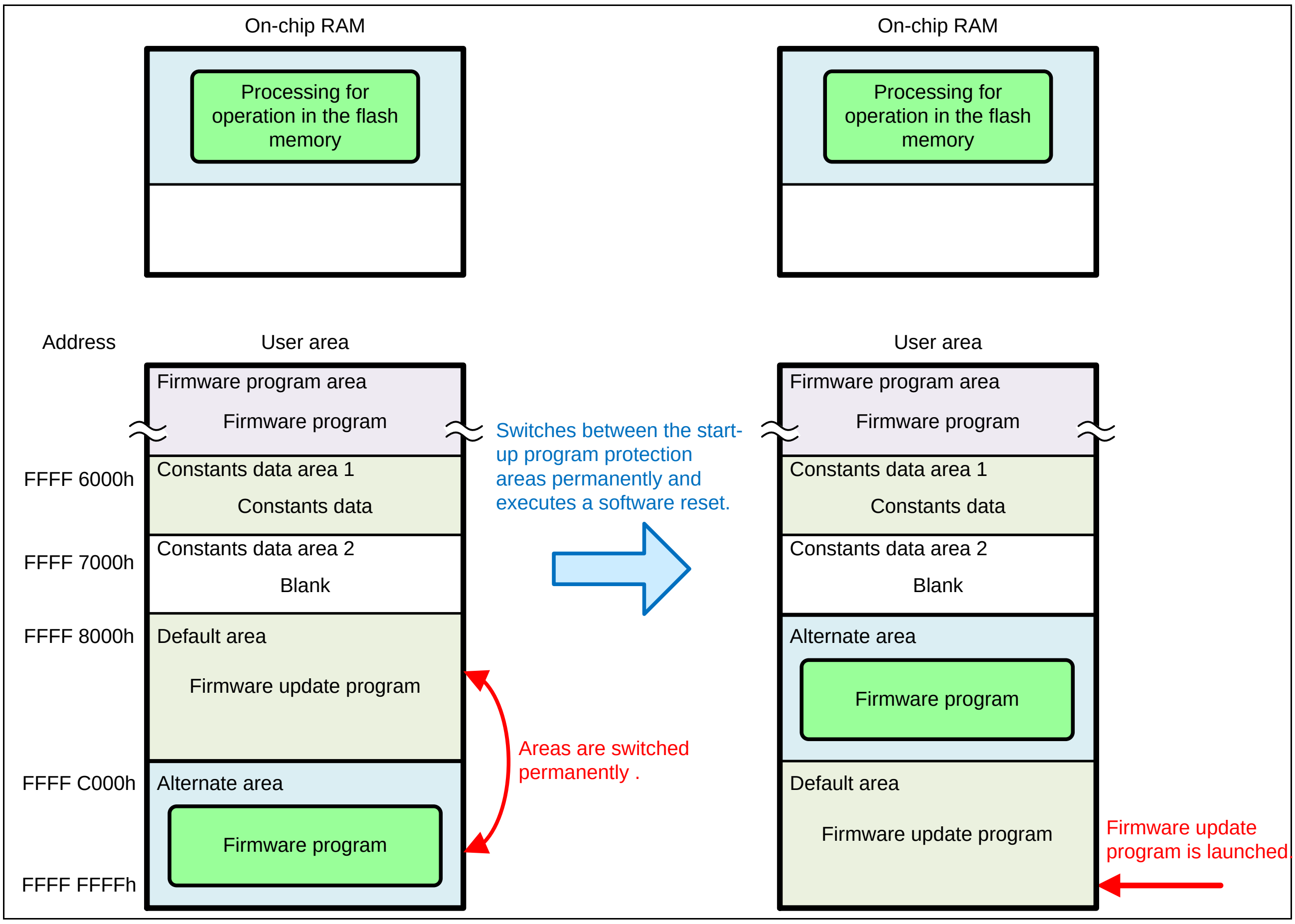


Figure 5.26 Software Reset and Launching the Firmware Update Program (for RX231 Group)

5.2.6 Recovery when Programming Failed

When programming the code flash memory using the firmware update program and if the programming failed, for example, due to temporary blackout, the programming can be performed again by resetting the device. The following explains how this works:

The firmware update program temporarily switches between the default area and the alternate area of the start-up program protection by the setting of the flash initial setting register (FISR) before erasing and programming the code flash memory. Then it programs the code flash memory except the default area. The setting of the flash initial setting register (FISR) is initialized by resetting the device. The default area and the alternate area after the register initialization are determined according to the start-up area setting in the extra area. When programming the code flash memory by the firmware update program, the start-up area setting for the extra area is set to launch from the default area. Therefore resetting the device causes the firmware update program in the default area to be launched and enables programming the code flash memory again.

5.2.7 How to Create a Firmware Update Program for Updating

When creating a firmware update program to be used when updating the firmware update program, change the items shown below.

(1) **Changing the section settings**

In the section settings, change the start address of each section. The firmware update program for updating needs to store constants data in the constants data area that is not used by the running firmware update program. Therefore, if the program before the update is using constants data area 2, set the start address as shown in Table 5.3. If the program before the update is using constants data area 1, set the start address as shown in Table 5.4.

Table 5.3 Start Address of Each Section When Constants Data is Stored in Constants Data Area 1

	Section		
Device	**FW_UP_VER**	**C_1**	**FW_UP_COMPLETE**
RX130 group	0xFFFF6800	0xFFFF6808	0xFFFF73F0
RX140 group RX231 group	0xFFFF6000	0xFFFF6008	0xFFFF6FF0

Table 5.4 Start Address of Each Section When Constants Data is Stored in Constants Data Area 2

	Section		
Device	**FW_UP_VER**	**C_1**	**FW_UP_COMPLETE**
RX130 group	0xFFFF7400	0xFFFF7408	0xFFFF7FF0
RX140 group RX231 group	0xFFFF7000	0xFFFF7008	0xFFFF7FF0

(2) **Changing the version information**

Change the version information of the firmware update program. To change the version information, change the value of the constant FW_UP_PROGRAM_VERSION in main.c. Version information should be 4-digit hexadecimal number from 0 to 9.

The following shows version information setting examples:

- For Ver3.05:

```
#define FW_UP_PROGRAM_VERSION (0x0305u)
```

- For Ver10.20

```
#define FW_UP_PROGRAM_VERSION (0x1020u)
```

5.3 Process Flowchart and Screen Output: Firmware Update Program

The firmware update program uses the serial communication to output the message to the terminal software on the host PC and branches to an appropriate processing according to the command input from the terminal software.

5.3.1 Main Processing

Main processing of the firmware update program initializes the SCI FIT module and the Flash FIT module, and uses the SCI to display the menu in the terminal software on the host PC. Then main processing waits for a key input from the terminal software and branches to an appropriate processing according to the key input.

(1) **Process flowchart**

Figure 5.27 shows the Flowchart of Main Processing.

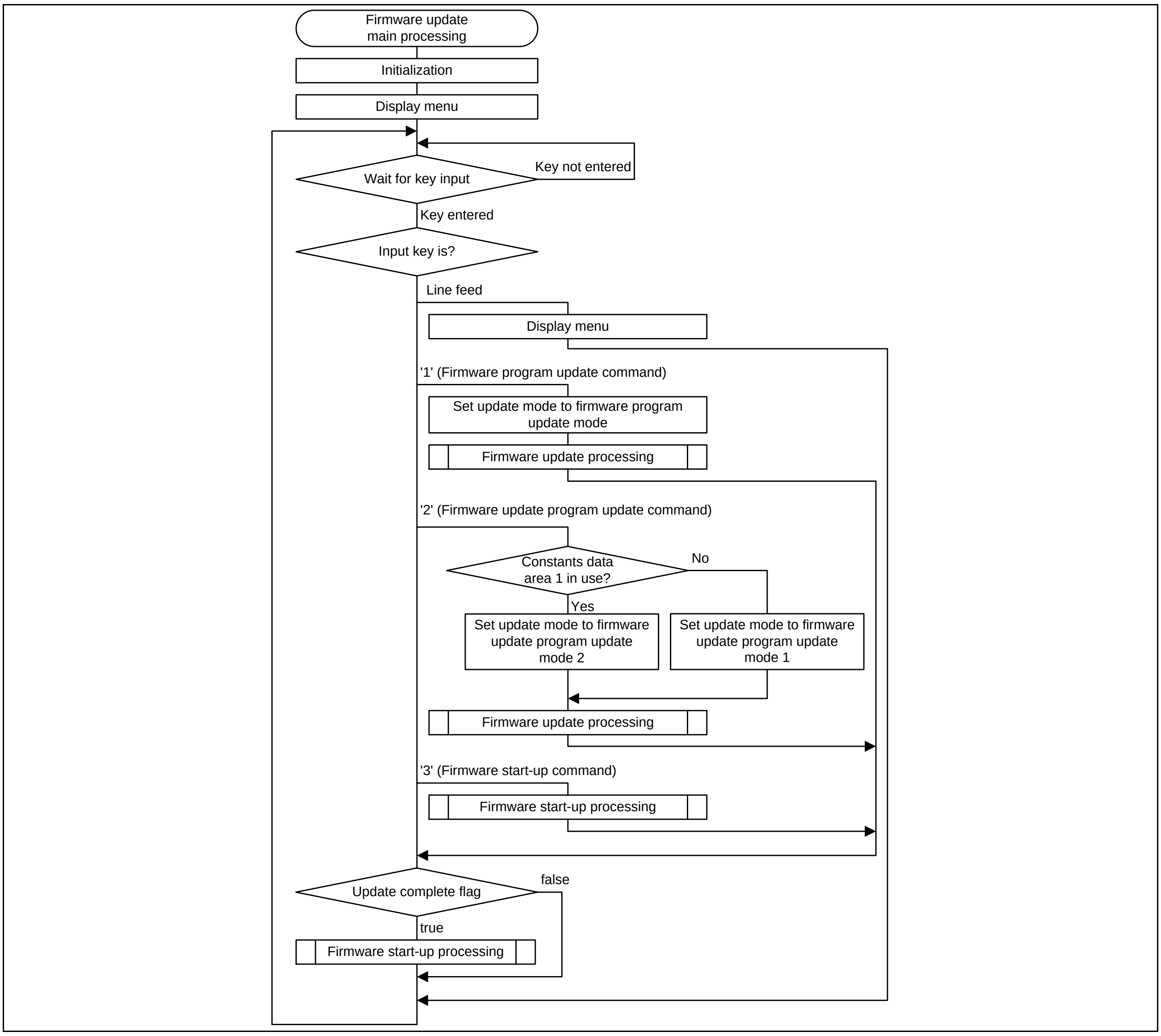

Firmware update main processing
Initialization
Display menu
Wait for key input
Key not entered
Key entered
Input key is?
Line feed
Display menu
'1' (Firmware program update command)
Set update mode to firmware program update mode
Firmware update processing
'2' (Firmware update program update command)
Constants data area 1 in use?
No
Yes
Set update mode to firmware update program update mode 2
Set update mode to firmware update program update mode 1
Firmware update processing
'3' (Firmware start-up command)
Firmware start-up processing
Update complete flag
false
true
Firmware start-up processing

Figure 5.27 Flowchart of Main Processing

(2) Screen output in the terminal software

After the firmware update program is launched, the following message is output.

RX130 group:

```
RX130 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

Figure 5.28 Screen Output by Main Processing (RX130 Group)

RX140 group:

```
RX140 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

Figure 5.29 Screen Output by Main Processing (RX140 Group)

RX231 group:

```
RX231 firmware update using Start-Up Program Protection menu ver1.00
1...Update firmware program
2...Update firmware update program
3...Execute firmware
>
```

Figure 5.30 Screen Output by Main Processing (RX231 Group)

When ‘1’ is entered, the firmware program is updated. The update mode is set to firmware program update mode. After the update mode is set, the process branches to the firmware update processing.

When ‘2’ is entered, the firmware update program is updated. The update mode is set to either firmware update program update mode 1 or firmware update program update mode 2. After the update mode is set, the process branches to the firmware update processing.

When ‘3’ is entered, the firmware start-up processing is executed.

When a line feed is entered, the menu is displayed again.

(3) Update Mode

When the firmware update program is used to update a program, the update mode shown in Table 5.5 is set to branch into the firmware update processing.

Table 5.5 Overview of the Update Mode

Update Mode	Description
Firmware program update mode	This mode updates the firmware program. Programming or erasing is performed only for the firmware program area and alternate area of the start-up program protection.
Firmware update program update mode 1	This mode updates the firmware update program. Programming or erasing is performed only for the constants data area 1 and alternate area of the start-up program protection.
Firmware update program update mode 2	This mode updates the firmware update program. Programming or erasing is performed only for the constants data area 2 and alternate area of the start-up program protection.

5.3.2 Firmware Update Processing

When ‘1’ or ‘2’ is entered in main processing, firmware update processing is executed. The update program is received through the serial communication using the XMODEM/SUM protocol and then programmed into the code flash memory.

(1) **Process flowchart**

Figure 5.31 shows the Flowchart of Firmware Update Processing.

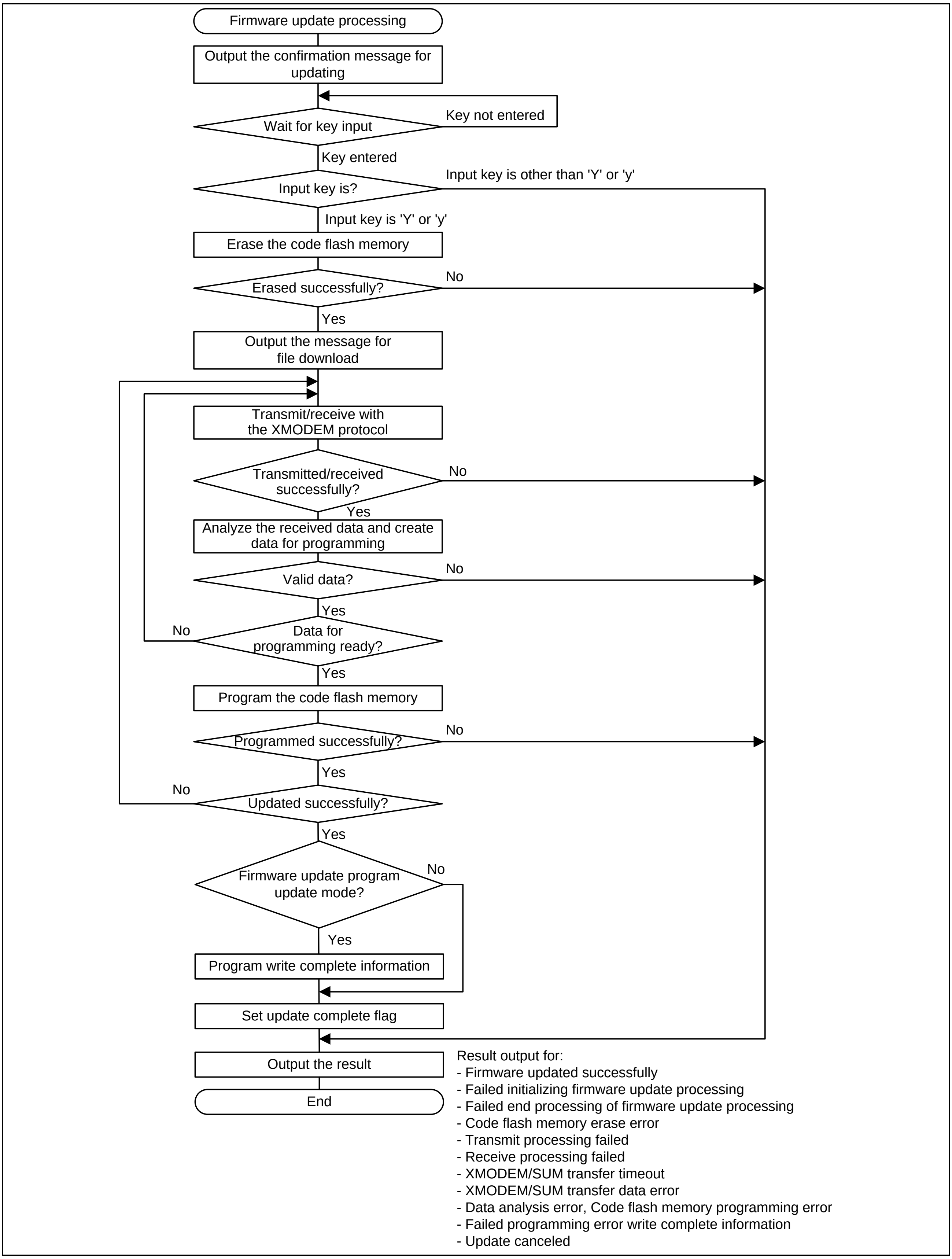

Firmware update processing
Output the confirmation message for updating
Key not entered
Wait for key input
Key entered
Input key is other than 'Y' or 'y'
Input key is?
Input key is 'Y' or 'y'
Erase the code flash memory
No
Erased successfully?
Yes
Output the message for file download
Transmit/receive with the XMODEM protocol
No
Transmitted/received successfully?
Yes
Analyze the received data and create data for programming
No
Valid data?
Yes
No
Data for programming ready?
Yes
Program the code flash memory
No
Programmed successfully?
Yes
No
Updated successfully?
Yes
Firmware update program update mode?
No
Yes
Program write complete information
Set update complete flag
Output the result
End
Result output for:
- Firmware updated successfully
- Failed initializing firmware update processing
- Failed end processing of firmware update processing
- Code flash memory erase error
- Transmit processing failed
- Receive processing failed
- XMODEM/SUM transfer timeout
- XMODEM/SUM transfer data error
- Data analysis error, Code flash memory programming error
- Failed programming error write complete information
- Update canceled

Figure 5.31 Flowchart of Firmware Update Processing

(2) Screen output in the terminal software

1. Confirmation message for updating

Firmware update processing outputs the following confirmation message for updating.

```
Erase flash memory and write firmware (Y/N)?
```

Figure 5.32 Screen Output to Confirm Updating

2. Downloading the file

```
When 'Y' or 'y' is entered, the processing erases the code flash memory, outputs
the message as shown Erasing has been done.
Start Xmodem download...
```

Figure 5.33, and waits for the firmware being received. Transmit the .mot file with the XMODEM/SUM protocol from the terminal software. For transmitting a file with the XMODEM/SUM protocol from the terminal software, refer to the user's manual for the terminal software.

```
Erasing has been done.
Start Xmodem download...
```

Figure 5.33 Screen Output for Downloading the File

3. Completion of the firmware update

When the firmware has been updated, the following message is output.

```
Updating firmware has been done.
>
```

Figure 5.34 Screen Output upon Completion of the Firmware Update

4. Error output

If an error occurs during firmware update, any of the following message is output according to the error.

```
Initialize update error.: Failed initializing firmware update processing
Finalize update error.   : Failed end processing of firmware update processing
Erasing error.           : Code flash memory erase error
Send error.             : Failed transmission
Receive error.          : Failed reception
Timeout.                : XMODEM/SUM transfer timeout
Data error              : XMODEM/SUM transfer data error
Block processing error.: Data analysis error, Code flash memory programming
                        : error
Set write complete information error.   : Failed programming
                        :write complete information
```

Figure 5.35 Screen Output upon Error Occurrence

5. Canceling the update

If a key other than ‘Y’ or ‘y’ is entered in response to the confirmation message for updating in 1. above, the following message is output and the update is canceled.

```
Command canceled.
>
```

Figure 5.36 Screen Output when Updating is Canceled

5.3.3 Firmware Start-up Processing

When ‘3’ is entered in main processing, firmware start-up processing is executed. The firmware is launched by executing a software reset after switching the start-up program protection area.

If the firmware program or firmware update program is updated, firmware start-up processing is executed without displaying the menu.

(1) Process flowchart

Figure 5.37 shows the Flowchart of Program Start-up Processing.

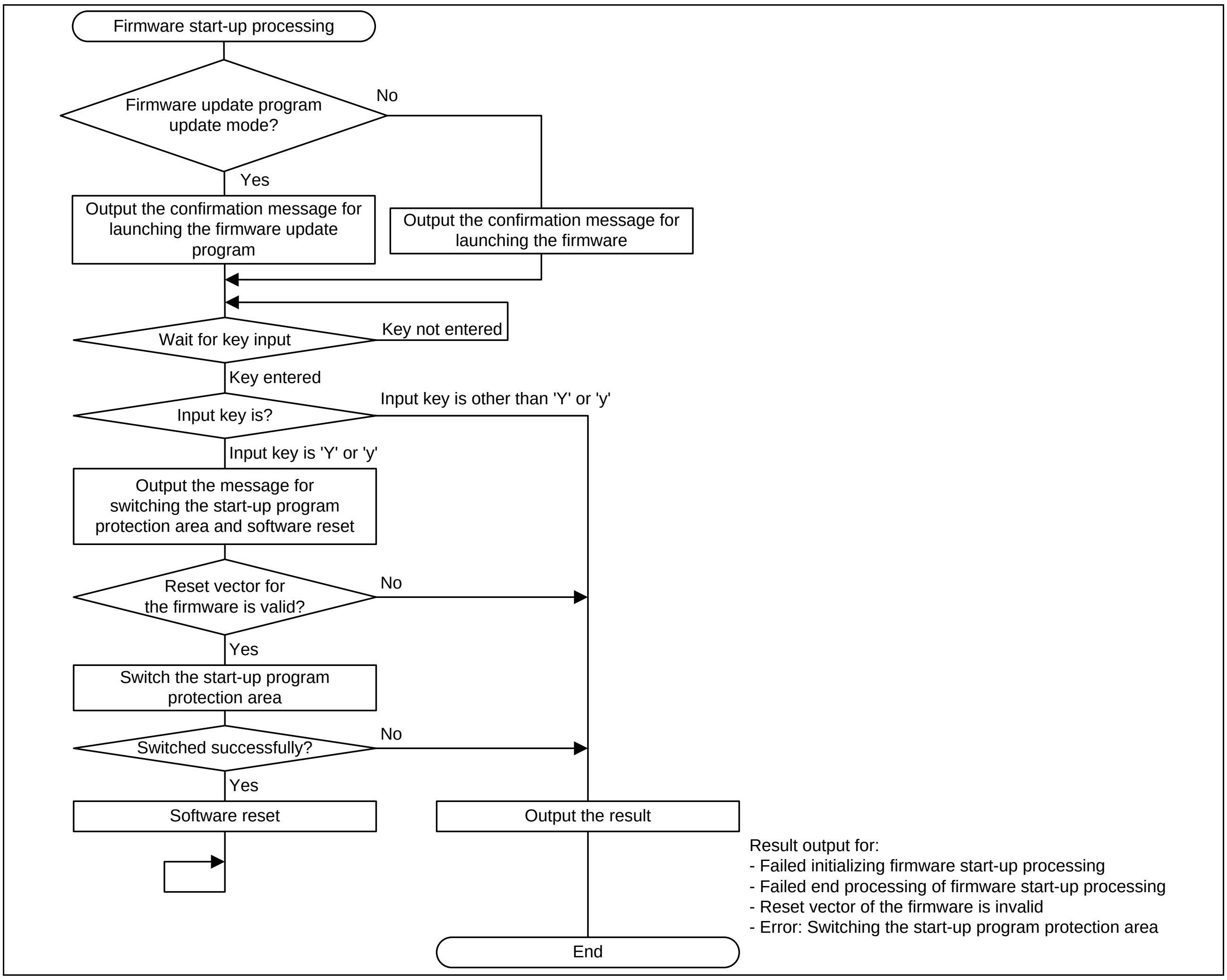

Firmware start-up processing
Firmware update program update mode?
No
Yes
Output the confirmation message for launching the firmware update program
Output the confirmation message for launching the firmware
Wait for key input
Key not entered
Key entered
Input key is?
Input key is other than 'Y' or 'y'
Input key is 'Y' or 'y'
Output the message for switching the start-up program protection area and software reset
Reset vector for the firmware is valid?
No
Yes
Switch the start-up program protection area
Switched successfully?
No
Yes
Software reset
Output the result
Result output for:
- Failed initializing firmware start-up processing
- Failed end processing of firmware start-up processing
- Reset vector of the firmware is invalid
- Error: Switching the start-up program protection area
End

Figure 5.37 Flowchart of Program Start-up Processing

(2) Screen output in the terminal software

1. Confirmation message for launching the program.

The firmware start-up processing outputs the following confirmation message for launching the firmware.

```
Execute firmware (Y/N)?
```

Figure 5.38 Screen Output for Launching the Firmware

2. Confirmation message for launching the firmware update program

The firmware start-up processing after the firmware update program is updated outputs the message as shown in Figure 5.39 to confirm the new firmware update program version.

```
Execute new firmware update program Ver[new-firmware-update-program-
version](Y/N)?
```

Figure 5.39 Screen Output for Launching the New Firmware Update Program

3. Launching the firmware

If ‘Y’ or ‘y’ is entered in response to the message above, the processing outputs the following message, switches the start-up program protection area, and executes a software reset.

```
Switch Start-Up area and do software reset.
```

Figure 5.40 Screen Output for Switching the Start-up Program Protection Area and Software Reset

4. Error output

If an error occurs during launching the firmware, any of the following message is output according to the error.

```
Initialize update error.: Failed initializing firmware start-up processing
Finalize update error.  : Failed end processing of firmware start-up processing
Reset vector of the firmware is invalid.: Reset vector of the firmware is invalid
Switching Start-Up area error. : Error: Switching the start-up program
                               : protection area
```

Figure 5.41 Screen Output upon Error Occurrence

5. Canceling the firmware start-up

If a key other than ‘Y’ or ‘y’ is entered in response to the confirmation message for launching the firmware in 1. above, the following message is output and the update is canceled.

```
Command canceled.
>
```

Figure 5.42 Screen Output when the Firmware Start-up is Canceled

5.4 Process Flowchart and Screen Output: Firmware Program

The firmware program uses the serial communication to output the message to the terminal software on the host. If a line feed is entered in the terminal software, the firmware switches the start-up program protection area, executes the software reset, and launch the firmware update program.

(1) Process flowchart

Figure 5.43 shows the Flowchart of the Firmware Program.

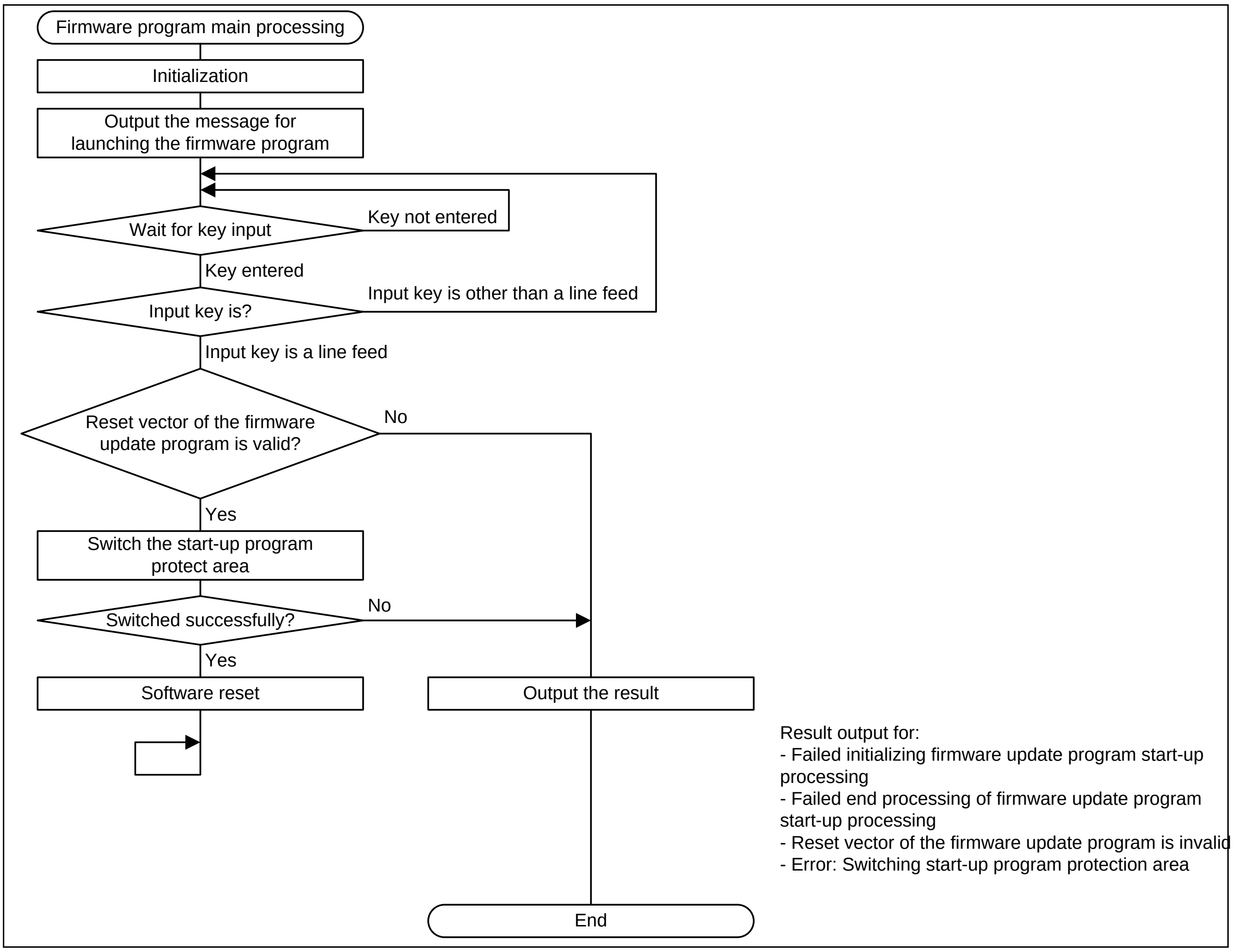

Firmware program main processing
Initialization
Output the message for launching the firmware program
Wait for key input
Key not entered
Key entered
Input key is?
Input key is other than a line feed
Input key is a line feed
Reset vector of the firmware update program is valid?
No
Yes
Switch the start-up program protect area
Switched successfully?
No
Yes
Software reset
Output the result
Result output for:
- Failed initializing firmware update program start-up processing
- Failed end processing of firmware update program start-up processing
- Reset vector of the firmware update program is invalid
- Error: Switching start-up program protection area
End

Figure 5.43 Flowchart of the Firmware Program

(2) Screen output in the terminal software

1. Firmware program start-up

The firmware program outputs the following message to inform that the firmware program is starting.

```
This program is the sample firmware.
Push Enter key to execute firmware update.
>
```

Figure 5.44 Screen Output when the Firmware Program is Starting-up

When a line feed is entered, the firmware switches the start-up program protection area, executes a software reset, and launch the firmware update program.

2. Error output

If an error occurs during launching the firmware update program, any of the following message is output according to the error.

```
Initialize update error.: Failed initializing firmware update program start-up
                         : processing
Finalize update error.   : Failed end processing of firmware update program
                         : start-up processing
Reset vector of the firmware is invalid. : Reset vector of the firmware update
                                         : program is invalid
Switching Start-Up area error. : Error: Switching the start-up program
                               : protection area
```

Figure 5.45 Screen Output upon Error Occurrence

5.5 Detailed Information of the Firmware Update Program

5.5.1 File Composition

Table 5.6 lists the Files Used in the Firmware Update Program and Table 5.7 lists the Standard Include Files Used in the Firmware Update Program. Files generated by the FIT module and files generated by the integrated development environment are not included in these tables.

Table 5.6 Files Used in the Firmware Update Program

File Name	Outline
main.c	Main source file
main.h	Main interface file
r_xmodem.c	XMODEM source file
r_xmodem_if.h	XMODEM interface file
r_fw_up_rx.c	Firmware update source file
r_fw_up_rx_if.h	Firmware update interface file
r_fw_up_rx_private.h	Firmware update header file
r_fw_up_buf.c	Source file to process the firmware data buffer
r_fw_up_buf.h	Header file to process the firmware data buffer

Table 5.7 Standard Include Files Used in the Firmware Update Program

File Name	Outline
stdbool.h	Defines macros regarding the Boolean type and the Boolean value.
stdint.h	Defines macros by declaring the integer type of the specified width.
stdlib.h	Library for standard C programming processing such as storage area management
string.h	Library for processing such as string comparison and copy.

5.5.2 Constants

Table 5.8 to Table 5.13 list constants used in the firmware update program.

Table 5.8 Constants Used in the Firmware Update Program (main.c)

Constant	Setting Value	Description
FW_UP_PROGRAM_VERSION	(0x0100u)	Version information
DUMMY_DATA	(0xAA55AA55AA55AA55)	Dummy data to be stored before the write complete information storage area
MASK_NUM	(0x0Fu)	Mask for getting lower 4 bits
ASCII_CODE_NUM	(0x30u)	Character code of ‘0’
ASCII_CODE_POINT	(0x2Eu)	Character code of ‘.’
DELAY_NUM	(1u)	Delay time to be passed as an argument to the R_BSP_SoftwareDelay function
TIMEOUT_NUM	(10000u)	Count value for determining the 10-second time out period
RECV_BYTE_SIZE	(1u)	1 byte size for receiving
SEND_BYTE_SIZE	(1u)	1 byte size for transmitting
COMMAND_UPDATE_FIRM	(‘1’)	Character code for the input command (for firmware program update command)
COMMAND_UPDATE_FIRM_UPDATE	(‘2’)	Character code for the input command (for firmware update program update command)
COMMAND_EXEC_PROGRAM	(‘3’)	Character code for the input command (for firmware start-up command)
COMMAND_YES_UPPER	(‘Y’)	Character code for the input command (“Y”)
COMMAND_YES_LOWER	(‘y’)	Character code for the input command (“y”)
COMMAND_CR	(‘\r’)	Character code for the input command (line feed)
STRING_MAX_SIZE	RX130 group: (SCI_CFG_CH1_TX_BUFSIZ) RX140 group: (SCI_CFG_CH1_TX_BUFSIZ) RX231 group: (SCI_CFG_CH5_TX_BUFSIZ)	Maximum size for an output string

Table 5.9 Constants Used in the Firmware Update Program (r_xmodem.c)

Constant	Setting Value	Description
XM_SOH	(0x01u)	XMODEM control code (SOH)
XM_EOT	(0x04u)	XMODEM control code (EOT)
XM_ACK	(0x06u)	XMODEM control code (ACK)
XM_NAK	(0x15u)	XMODEM control code (NAK)
XM_CAN	(0x18u)	XMODEM control code (CAN)
XM_HEADER_SIZE	(1+1+1)	Header size of the XMODEM data block (the number of bytes)
XM_DATA_SIZE	(128u)	Data size of the XMODEM data block (the number of bytes)
XM_SUM_SIZE	(1u)	Checksum size of the XMODEM data block (the number of bytes)
XM_BLOCK_SIZE	(XM_HEADER_SIZE + XM_DATA_SIZE + XM_SUM_SIZE)	XMODEM data block size (the number of bytes)
XM_RETRY_COUNT	(10u)	The number of retries upon XMODEM data transfer timeout
UINT8T_0	(0u)	0 in uint8_t
UINT8T_1	(1u)	1 in uint8_t
COMPLEMENT_CHECK	(0xFFu)	The numerical value to confirm the complement of the block number

Table 5.10 Constants Used in the Firmware Update Program (r_fw_up_rx_private.h) (1/3)

Constant	Setting Value	Description
FW_UP_BINARY_BUF_SIZE	(256u)	Buffer size for data to be programmed in the code flash memory
FW_UP_BINARY_BUF_NUM	(2u)	The number of buffers for data to be programmed in the code flash memory
FW_UP_BUF_NUM	(60u)	The number of arrays to store analyzed Motorola S record data
FW_UP_FIRM_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_127) RX140 group: (FLASH_CF_BLOCK_127) RX231 group: (FLASH_CF_BLOCK_255)	Start address of the area to program the firmware
FW_UP_FIRM_EN_ADDRESS	RX130 group: (FLASH_CF_BLOCK_37 – 1) RX140 group: (FLASH_CF_BLOCK_19 – 1) RX231 group: (FLASH_CF_BLOCK_19 – 1)	End address of the area to program the firmware
FW_UP_CONST_1_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_37) RX140 group: (FLASH_CF_BLOCK_19) RX231 group: (FLASH_CF_BLOCK_19)	Start address of the constants data area 1
FW_UP_CONST_1_FIN_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_34 - 256) RX140 group: (FLASH_CF_BLOCK_17 - 256) RX231 group: (FLASH_CF_BLOCK_17 - 256)	Start address to be used for final programming processing in constants data area 1
FW_UP_COMPLETE_1_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_34 - 8) RX140 group: (FLASH_CF_BLOCK_17 - 8) RX231 group: (FLASH_CF_BLOCK_17 - 8)	Start address of the write complete information storage area 1
FW_UP_CONST_1_EN_ADDRESS	RX130 group: (FLASH_CF_BLOCK_34 - 1) RX140 group: (FLASH_CF_BLOCK_17 - 1) RX231 group: (FLASH_CF_BLOCK_17 - 1)	End address of the constants data area 1

Table 5.11 Constants Used in the Firmware Update Program (r_fw_up_rx_private.h) (2/3)

Constant	Setting Value	Description
FW_UP_CONST_2_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_34) RX140 group: (FLASH_CF_BLOCK_17) RX231 group: (FLASH_CF_BLOCK_17)	Start address of the constants data area 2
FW_UP_CONST_2_FIN_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_31 - 256) RX140 group: (FLASH_CF_BLOCK_15 - 256) RX231 group: (FLASH_CF_BLOCK_15 - 256)	Start address to be used for final programming processing in constants data area 2
FW_UP_COMPLETE_2_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_31 - 8) RX140 group: (FLASH_CF_BLOCK_15 - 8) RX231 group: (FLASH_CF_BLOCK_15 - 8)	Start address of the write complete information storage area 2
FW_UP_CONST_2_EN_ADDRESS	RX130 group: (FLASH_CF_BLOCK_31 - 1) RX140 group: (FLASH_CF_BLOCK_15 - 1) RX231 group: (FLASH_CF_BLOCK_15 - 1)	End address of the constants data area 2
FW_UP_STUP_ST_ADDRESS	RX130 group: (FLASH_CF_BLOCK_15) RX140 group: (FLASH_CF_BLOCK_7) RX231 group: (FLASH_CF_BLOCK_7)	Start address of the default area of the start-up program protection
FW_UP_STUP_EN_ADDRESS	(FLASH_CF_BLOCK_END)	End address of the default area of the start-up program protection
FW_UP_FIRM_BLOCK_NUM	RX130 group: (90u) RX140 group: (108u) RX231 group: (236u)	The number of blocks for the area to program the firmware
FW_UP_CONST_BLOCK_NUM	RX130 group: (3u) RX140 group: (2u) RX231 group: (2u)	The number of blocks for the constants data area
FW_UP_STUP_BLOCK_NUM	RX130 group: (16u) RX140 group: (8u) RX231 group: (8u)	The number of blocks for the default area of the start-up program protection

Table 5.12 Constants Used in the Firmware Update Program (r_fw_up_rx_private.h) (3/3)

Constant	Setting Value	Description
FW_UP_FIRM_RESETVECT	RX130 group: (FLASH_CF_BLOCK_15 – 4) RX140 group: (FLASH_CF_BLOCK_7 – 4) RX231 group: (FLASH_CF_BLOCK_7 – 4)	Address of the reset vector of the firmware program
FW_UP_BLANK_VALUE	(0xFFFFFFFFu)	Read value when the code flash memory is blank.
PRCR_KEY	(0xA500u)	Key code of PRCR register
SET_PRC1	(0x0002u)	Value for setting the PRC1 bit of the PRCR register
MCU_RESET	(0xA501u)	A value that is set in the SWRR register to reset the MCU

Table 5.13 Constants Used in the Firmware Update Program (r_fw_up_buf.h)

Constant	Setting Value	Description
MOT_S_CHECK_SUM_FIELD	(0x02u)	The number of characters for the checksum field in the Motorola S-record format
ADDRESS_LENGTH_S1	(0x04u)	The number of characters for the address field in the Motorola S-record format (S1 type)
ADDRESS_LENGTH_S2	(0x06u)	The number of characters for the address field in the Motorola S-record format (S2 type)
ADDRESS_LENGTH_S3	(0x08u)	The number of characters for the address field in the Motorola S-record format (S3 type)
BUF_LOCK	(1u)	The specified buffer of Motorola S-record format is locked.
BUF_UNLOCK	(0u)	The specified buffer of Motorola S-record format is open.
MOT_RECORD_S0	(0u)	Record type in Motorola S-record format (S0 type)
MOT_RECORD_S1	(1u)	Record type in Motorola S-record format (S1 type)
MOT_RECORD_S2	(2u)	Record type in Motorola S-record format (S2 type)
MOT_RECORD_S3	(3u)	Record type in Motorola S-record format (S3 type)
MOT_RECORD_S7	(7u)	Record type in Motorola S-record format (S7 type)
MOT_RECORD_S8	(8u)	Record type in Motorola S-record format (S8 type)
MOT_RECORD_S9	(9u)	Record type in Motorola S-record format (S9 type)
MASK_LOWER_BYTE	(0x000000FFu)	Mask for getting lower 1 byte.
ASCII_CODE_0	(0x30u)	Character code of ‘0’.
ASCII_CODE_9	(0x39u)	Character code of ‘9’.
ASCII_CODE_UPPER_A	(0x41u)	Character code of ‘A’.
ASCII_CODE_UPPER_F	(0x46u)	Character code of ‘F’.
ASCII_CODE_LOWER_A	(0x61u)	Character code of ‘a’.
ASCII_CODE_LOWER_F	(0x66u)	Character code of ‘f’.
CONVERT_HEX_NUM	(0x0Fu)	Value for converting character code from 0 to 9 to hexadecimal.
CONVERT_HEX_UPPER_CHAR	(0x37u)	Value for converting character code from A to F to hexadecimal.
CONVERT_HEX_LOWER_CHAR	(0x57u)	Value for converting character code from a to f to hexadecimal.

5.5.3 Type Definitions

Figure 5.46 to Figure 5.49 show type definitions used in the firmware update program.

```
typedef enum e_xmodem_proc_stage
{
    XMODEM_PROC_END = 0,
    XMODEM_PROCESSING,
    XMODEM_SOH_RECEIVED
} e_xmodem_proc_stage_t;

typedef struct st_xmodem_states
{
    uint8_t                 retry_counter;
    uint8_t                 expected_block_number;
    uint8_t                 recv_buf_index;
    uint8_t                 can_counter;
    uint8_t               * p_recv_buf;
    e_xmodem_proc_stage_t   proc_stage;
    xm_recv_func_t          recv_func;
    xm_send_func_t          send_func;
    xm_exec_func_t          exec_func;
} st_xmodem_states_t;
```

Figure 5.46 Type Definitions Used in the Firmware Update Program (r_xmodem.c)

```
typedef enum e_xmodem_err
{
    XMODEM_SUCCESS,
    XMODEM_SEND_ERR,
    XMODEM_RECV_ERR,
    XMODEM_TIMEOUT,
    XMODEM_PROC_BLOCK_ERR,
    XMODEM_RECV_CAN,
    XMODEM_DATA_ERR
} e_xmodem_err_t;

typedef e_xmodem_err_t (*xm_recv_func_t)(uint8_t* p_arg);
typedef e_xmodem_err_t (*xm_send_func_t)(uint8_t arg);
typedef e_xmodem_err_t (*xm_exec_func_t)(const uint8_t* p_buf, uint16_t size);
```

Figure 5.47 Type Definitions Used in the Firmware Update Program (r_xmodem_if.h)

```
typedef enum e_update_mode_t
{
    UPDATE_FW,
    UPDATE_FW_UP_1,
    UPDATE_FW_UP_2,
} update_mode_t;

typedef enum e_fw_up_return_t
{
    FW_UP_SUCCESS,
    FW_UP_ERR_OPENED,
    FW_UP_ERR_NOT_OPEN,
    FW_UP_ERR_NULL_PTR,
    FW_UP_ERR_INVALID_RECORD,
    FW_UP_ERR_BUF_FULL,
    FW_UP_ERR_BUF_EMPTY,
    FW_UP_ERR_INITIALIZE,
    FW_UP_ERR_ERASE,
    FW_UP_ERR_WRITE,
    FW_UP_ERR_VERIFY,
    FW_UP_ERR_SWITCH_AREA,
    FW_UP_ERR_INVALID_ADDRESS,
    FW_UP_ERR_INVALID_RESETVECT,
    FW_UP_ERR_INTERNAL
} fw_up_return_t;

typedef struct st_fw_up_fl_data_t
{
    uint32_t src_addr;
    uint32_t dst_addr;
    uint32_t len;
    uint16_t count;
} fw_up_fl_data_t;
```

Figure 5.48 Type Definitions Used in the Firmware Update Program (r_fw_up_rx_if.h)

```
typedef enum fw_up_mot_s_cnt_t
{
    STATE_MOT_S_RECORD_MARK = 0,
    STATE_MOT_S_RECORD_TYPE,
    STATE_MOT_S_LENGTH_1,
    STATE_MOT_S_LENGTH_2,
    STATE_MOT_S_ADDRESS,
    STATE_MOT_S_DATA,
    STATE_MOT_S_CHKSUM_1,
    STATE_MOT_S_CHKSUM_2
} fw_up_mot_s_cnt_t;

typedef struct MotSBufS
{
    uint8_t           addr_length;
    uint8_t           data_length;
    uint8_t         * p_address;
    uint8_t         * p_data;
    uint8_t           type;
    uint8_t           act;
    struct MotSBufS * p_next;
} fw_up_mot_s_buf_t;

typedef struct WriteDataS
{
    uint32_t            addr;
    uint32_t            len;
    uint8_t             data[FW_UP_BINARY_BUF_SIZE];
    struct WriteDataS * p_next;
    struct WriteDataS * p_prev;
} fw_up_write_data_t;
```

Figure 5.49 Type Definitions Used in the Firmware Update Program (r_fw_up_buf.h)

5.5.4 Variables

Table 5.14 to Table 5.17 list static variables and Table 5.18 to Table 5.19 lists const variables.

Table 5.14 static Variables Used in the Firmware Update Program (main.c)

Type	Variable	Description	Function
static uint8_t	s_update_mode	Update mode	main block_proc_xm update_firmware exec_firmware set_write_complete_information
static uint8_t	s_update_complete_flag	Update complete flag	main update_firmware exec_firmware
static sci_hdl	s_sci_handle	SCI module control handle	main send_byte_xm recv_byte_xm update_firmware exec_firmware send_string_sci
static volatile bool	s_sci_send_end_flag	SCI transmit complete determination flag	sci_callback send_string_sci
static volatile int32_t	s_timeout_count	Timeout determination counter	recv_byte_xm

Table 5.15 static Variable Used in the Firmware Update Program (r_xmodem.c)

Type	Variable	Description	Function
static uint8_t	s_recv_buf[XM_BLOCK_SIZE]	XMODEM receive data buffer	exec_xmodem

Table 5.16 static Variable Used in the Firmwaer Update Program (r_fw_up_rx.c)

Type	Variable	Description	Function
static bool	s_is_opened	Firmware update initialization complete flag	fw_up_open fw_up_close fw_up_put_data fw_up_get_data erase_firmware write_firmware switch_start_up_and_reset

Table 5.17 static Variables Used in the Firmware Update Program (r_fw_up_buf.c)

Type	Variable	Description	Function
static fw_up_mot_s_buf_t	*sp_app_put_mot_s_buf	Pointer to the Motorola S-record data buffer currently used for Motorola S format analysis processing	fw_up_buf_init fw_up_put_mot_s
static fw_up_mot_s_buf_t	*sp_app_get_mot_s_buf	Pointer to the Motorola S-record data buffer currently used for processing to create data to be programmed into the code flash memory	fw_up_buf_init fw_up_get_binary
static fw_up_mot_s_buf_t	s_mot_s_buf[FW_UP_ BUF_NUM]	Buffer to store the contents of the Motorola S-record data	fw_up_buf_init fw_up_memory_init
static fw_up_write_data_t	*sp_app_write_buf	Pointer to the current data buffer for programming the code flash memory	fw_up_buf_init fw_up_get_binary
static fw_up_write_data_t	s_write_buf[FW_UP_ BINARY_BUF_NUM]	Buffer to store the data for programming the code flash memory	fw_up_buf_init
static fw_up_mot_s_cnt_t	s_mot_s_data_state	Analysis state of the Motorola S-record data	fw_up_buf_init fw_up_put_mot_s
static uint32_t	s_write_current_address	Current address to program in the code flash memory	fw_up_buf_init fw_up_get_binary
static bool	s_detect_terminal_flag	Detection flag for the endpoint of the record	fw_up_buf_init fw_up_put_mot_s fw_up_get_binary

Table 5.18 const Variables Used in the Firmware Update Program (main.c) (1/2)

Type	Variable	Description	Function
const uint32_t	g_program_version	Firmware update program version	show_menu_start_up
const uint64_t	g_dummy_data		—
static const uint8_t	s_string_menu0[]	RX130group: “RX130 firmware update using Start-Up Program Protection menu ver” RX140group: “RX140 firmware update using Start-Up Program Protection menu ver” RX231 group: “RX231 firmware update using Start-Up Program Protection menu ver”	show_menu_start_up
static const uint8_t	s_string_menu1[]	“1…Update firmware program\r\n”	show_menu_start_up
static const uint8_t	s_string_menu2[]	"2...Update firmware update program\r\n"	show_menu_start_up
static const uint8_t	s_string_menu3[]	“3…Execute firmware\r\n”	show_menu_start_up
static const uint8_t	s_string_input[]	“> ”	show_menu_start_up update_firmware exec_firmware
static const uint8_t	s_string_crlf[]	“\r\n”	main update_firmware exec_firmware
static const uint8_t	s_string_update[]	“Erase flash memory and write firmware (Y/N)?”	update_firmware
static const uint8_t	s_string_erase_success[]	“Erasing has been done.\r\n”	update_firmware
static const uint8_t	s_string_download []	“Start Xmodem download…\r\n”	update_firmware
static const uint8_t	s_string_finish_xmodem[]	“Updating firmware has been done.\r\n”	update_firmware
static const uint8_t	s_string_exec_firm[]	“Execute firmware (Y/N)?”	exec_firmware
static const uint8_t	s_string_reset[]	“Switch Start-Up area and do software reset.\r\n”	exec_firmware
static const_uin8_t	s_string_exec_firm_update[]	"Execute new firmware update program Ver"	exec_firmware
static const uint8_t	s_string_y_n[]	" (Y/N)?"	exec_firmware
static const uint8_t	s_string_cancel[]	“Command canceled.\r\n”	update_firmware exec_firmware

Table 5.19 const Variables Used in the Firmware Update Program (main.c) (2/2)

Type	Variable	Description	Function
static const uint8_t	s_string_flash_err[]	“Flash module error.\r\n”	main
static const uint8_t	s_string_erase_err[]	“Erasing error.\r\n”	update_firmware
static const uint8_t	s_string_set_info_err[]	"Set write complete information error.\r\n"	update_firmware
static const uint8_t	s_string_send_err[]	“Send error.\r\n”	update_firmware
static const uint8_t	s_string_recv_err[]	“Receive error.\r\n”	update_firmware
static const uint8_t	s_string_timeout[]	“Timeout.\r\n”	update_firmware
static const uint8_t	s_string_block_err[]	“Block processing error.\r\n”	update_firmware
static const uint8_t	s_string_data_err[]	“Data error.\r\n”	update_firmware
static const uint8_t	s_string_fin_update_err[]	“Finalize update error.\r\n”	update_firmware exec_firmware
static const uint8_t	s_string_init_update_err[]	“Initialize update error.\r\n”	update_firmware exec_firmware
static const uint8_t	s_string_resetvect_err[]	“Reset vector of the firmware is invalid.\r\n”	exec_firmware
static const uint8_t	s_string_switch_err[]	“Switching Start-Up area error.\r\n”	exec_firmware

5.5.5 Functions

Table 5.20 lists the Functions Used in the Firmware Update Program, Table 5.21 lists the FIT Module Functions Used in the Firmware Update Program, Table 5.22 to Table 5.24 lists the e^2 studio Smart Configurator Generated Function Used in the Firmware Update Program.

Table 5.20 Functions Used in the Firmware Update Program

Function	Description	Defined File
main	Main processing	main.c
show_menu_start_up	Displaying menu	main.c
sci_callback	Callback function for the SCI FIT module to check completion of an SCI transmission	main.c
send_byte_xm	Callback function for XMODEM protocol to transmit 1-byte data	main.c
recv_byte_xm	Callback function for XMODEM protocol to receive 1-byte data	main.c
block_proc_xm	Callback function for XMODEM protocol for data processing of 1-data block	main.c
update_firmware	Firmware update processing	main.c
exec_firmware	Firmware start-up processing	main.c
send_string_sci	Transmitting strings	main.c
set_write_complete_information	Programming the version information to the write complete information storage area	main.c
show_version	Display version	main.c
exec_xmodem	XMODEM protocol processing	r_xmodem.c
xmodem_recv_soh	Receiving the header of XMODEM protocol data block	r_xmodem.c
xmodem_check_eot	Checking the header of XMODEM protocol data block	r_xmodem.c
xmodem_recv_block	Receiving 1-data block of XMODEM protocol	r_xmodem.c
xmodem_analyze_block	Analyzing XMODEM protocol data block	r_xmodem.c
xmodem_proc_data	Processing data for 1 data block of XMODEM protocol	r_xmodem.c
xmodem_send_response	Response for XMODEM protocol	r_xmodem.c
fw_up_open_flash	Flash FIT module initialization	r_fw_up_rx.c
fw_up_open	Firmware update initialization	r_fw_up_rx.c
fw_up_close	Completing firmware update	r_fw_up_rx.c
copy_update_ramprog	Copying RAM program	r_fw_up_rx.c
analyze_and_write_data	Analyzing receive data and programming code flash memory	r_fw_up_rx.c
fw_up_put_data	Analyzing receive data	r_fw_up_rx.c
fw_up_get_data	Obtaining programming data for the code flash memory	r_fw_up_rx.c
erase_firmware	Erasing the code flash memory	r_fw_up_rx.c
write_firmware	Programming the code flash memory	r_fw_up_rx.c
switch_start_up_and_reset	Switching the start-up program protection area and software reset	r_fw_up_rx.c
fw_up_buf_init	Initializing buffer for firmware update	r_fw_up_buf.c
fw_up_memory_init	Initializing pointer to the buffer	r_fw_up_buf.c
fw_up_put_mot_s	Analyzing Motorola S-record data	r_fw_up_buf.c
fw_up_get_binary	Obtaining programming data for the code flash memory	r_fw_up_buf.c
fw_up_ascii_to_hexbyte	Converting ASCII to binary	r_fw_up_buf.c

Table 5.21 FIT Module Functions Used in the Firmware Update Program

Function	FIT Module	Application	Function (Used in)
R_FLASH_Open	Flash FIT module	Initializing the Flash FIT module	fw_up_open_flash
R_FLASH_Erase	Flash FIT module	Erasing the code flash memory	erase_firmware
R_FLASH_Write	Flash FIT module	Programming the code flash memory	write_firmware set_write_complete_information
R_FLASH_Control	Flash FIT module	Switching the start-up program protection area	erase_firmware write_firmware switch_start_up_and_reset
R_SCI_Open	SCI FIT module	Starting up the SCI	main
R_SCI_Control	SCI FIT module	Enabling the transmit end interrupt	main
R_SCI_Send	SCI FIT module	Transmitting the SCI data	send_byte_xm send_string_sci
R_SCI_Receive	SCI FIT module	Receiving the SCI data	main recv_byte_xm update_firmware exec_firmware

Table 5.22 e^2 studio Smart Configurator Generated Function Used in the Firmware Update Program (RX130 Group)

Function	FIT Module	Application	Function (Used in)
R_SCI_PinSet_SCI1	SCI FIT module	Pin setting for the SCI	main

Table 5.23 e^2 studio Smart Configurator Generated Function Used in the Firmware Update Program (RX140 Group)

Function	FIT Module	Application	Function (Used in)
R_SCI_PinSet_SCI1	SCI FIT module	Pin setting for the SCI	main

Table 5.24 e^2 studio Smart Configurator Generated Function Used in the Firmware Update Program (RX231 Group)

Function	FIT Module	Application	Function (Used in)
R_SCI_PinSet_SCI5	SCI FIT module	Pin setting for the SCI	main

5.6 Detailed Information of the Firmware

5.6.1 File Composition

Table 5.25 lists the Files Used in the Firmware and Table 5.26 lists the Standard Include Files Used in the Firmware. Files generated by the FIT module and files generated by the integrated development environment are not included in these tables.

Table 5.25 Files Used in the Firmware

File Name	Outline
main.c	Main source file
main.h[*1]	Main interface file
r_fw_up_rx.c[*1]	Firmware update source file
r_fw_up_rx_if.h[*1]	Firmware update interface file
r_fw_up_rx_private.h[*1]	Firmware update header file
r_fw_up_buf.c[*1]	Source file to process the firmware data buffer
r_fw_up_buf.h[*1]	Header file to process the firmware data buffer

Note: 1. This is the same file used in the firmware update program.

Table 5.26 Standard Include Files Used in the Firmware

File Name	Outline
stdbool.h	Defines macros regarding the Boolean type and the Boolean value.
stdint.h	Defines macros by declaring the integer type of the specified width.
stdlib.h	Library for standard C programming processing such as storage area management
string.h	Library for processing such as string comparison and copy.

5.6.2 Constants

Table 5.27 lists Constants Used in the Firmware (main.c). For constants defined in the same file as the firmware update program, refer to 5.5.2 Constants.

Table 5.27 Constants Used in the Firmware (main.c)

Constant	Setting Value	Description
RECV_BYTE_SIZE	(1)	1 byte size for receiving
COMMAND_CR	(‘\r’)	Character code for the input command: line feed
STRING_MAX_SIZE	RX130 group: (SCI_CFG_CH1_TX_BUFSIZ) RX140 group: (SCI_CFG_CH1_TX_BUFSIZ) RX231 group: (SCI_CFG_CH5_TX_BUFSIZ)	Maximum size for an output string

5.6.3 Type Definitions

For type definitions, refer to 5.5.3 Type Definitions.

5.6.4 Variables

Table 5.28 lists static Variables Used in the Firmware (main.c) and Table 5.29 lists const Variables Used in the Firmware (main.c). For variables defined in the same file as the firmware update program, refer to 5.5.4 Variables.

Table 5.28 static Variables Used in the Firmware (main.c)

Type	Variable	Description	Function
static sci_hdl	s_sci_handle	SCI module control handle	main send_string_sci
static volatile bool	s_sci_send_end_flag	SCI transmit complete determination flag	sci_callback send_string_sci

Table 5.29 const Variables Used in the Firmware (main.c)

Type	Variable	Description	Function
static const uint8_t	s_string_menu0[]	"This program is the sample firmware.\r\n"	show_menu_start_up
static const uint8_t	s_string_menu1[]	"Push Enter key to execute firmware update.\r\n"	show_menu_start_up
static const uint8_t	s_string_input[]	"> "	show_menu_start_up
static const uint8_t	s_string_crlf[]	"\r\n"	main
static const uint8_t	s_string_reset[]	"Switch Start-Up area and do software reset.\r\n"	main
static const uint8_t	s_string_flash_err[]	"Flash module error.\r\n"	main
static const uint8_t	s_string_switch_err[]	"Switching Start-Up area error.\r\n"	main
static const uint8_t	s_string_init_update_err[]	"Initialize update error.\r\n"	main
static const uint8_t	s_string_fin_update_err[]	"Finalize update error.\r\n"	main
static const uint8_t	s_string_resetvect_err[]	"Reset vector of the firmware update is invalid.\r\n"	main

5.6.5 Functions

Table 5.30 lists the Functions Used in the Firmware, Table 5.31 lists the FIT Module Functions Used in the Firmware, Table 5.32 to Table 5.34 lists the e^2 studio Smart Configurator Generated Function Used in the Firmware. These tables do not include functions that are defined in the same file as the firmware update program and are not used by the firmware.

Table 5.30 Functions Used in the Firmware

Function	Description	Defined File
main	Main processing	main.c
show_menu_start_up	Displaying menu	main.c
sci_callback	Callback function for the SCI FIT module to check completion of an SCI transmission	main.c
send_string_sci	Transmitting strings	main.c
fw_up_open_flash	Flash FIT module initialization	r_fw_up_rx.c
fw_up_open	Firmware update initialization	r_fw_up_rx.c
fw_up_close	Completing firmware update	r_fw_up_rx.c
copy_update_ramprog	Copying RAM program	r_fw_up_rx.c
switch_start_up_and_reset	Switching the start-up program protection area and software reset	r_fw_up_rx.c
fw_up_buf_init	Initializing buffer for firmware update	r_fw_up_buf.c
fw_up_memory_init	Initializing pointer to the buffer	r_fw_up_buf.c

Table 5.31 FIT Module Functions Used in the Firmware

Function	FIT Module	Application	Function (Used in)
R_FLASH_Open	Flash FIT module	Initializing the Flash FIT module	fw_up_open_flash
R_FLASH_Control	Flash FIT module	Switching the start-up program protection area	switch_start_up_and_reset
R_SCI_Open	SCI FIT module	Starting up the SCI	main
R_SCI_Control	SCI FIT module	Enabling the transmit end interrupt	main
R_SCI_Send	SCI FIT module	Transmitting the SCI data	send_string_sci
R_SCI_Receive	SCI FIT module	Receiving the SCI data	main

Table 5.32 e^2 studio Smart Configurator Generated Function Used in the Firmware (RX130 Group)

Function	FIT Module	Application	Function (Used in)
R_SCI_PinSet_SCI1	SCI FIT module	Pin setting for the SCI	main

Table 5.33 e^2 studio Smart Configurator Generated Function Used in the Firmware (RX140 Group)

Function	FIT Module	Application	Function (Used in)
R_SCI_PinSet_SCI1	SCI FIT module	Pin setting for the SCI	main

Table 5.34 e^2 studio Smart Configurator Generated Function Used in the Firmware (RX231 Group)

Function	FIT Module	Application	Function (Used in)
R_SCI_PinSet_SCI5	SCI FIT module	Pin setting for the SCI	main

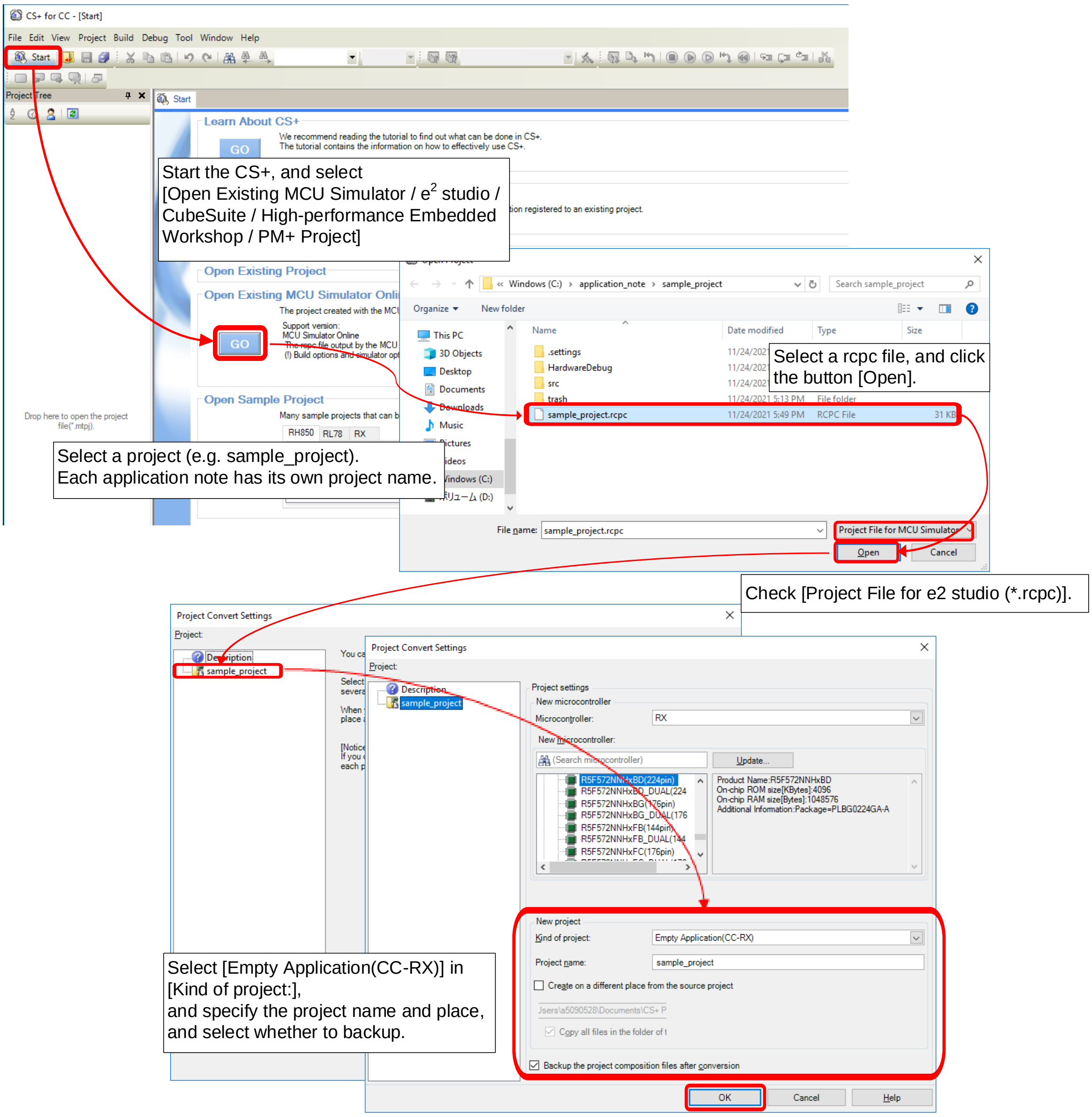
6. Import a Project

The sample programs are distributed in e^2 studio project format. This section shows how to import a project into CS+. After importing the sample project, make sure to confirm build and debugger setting.

6.1 Importing a Project into CS+

To use sample programs in CS+, follow the steps below to import them into CS+. In projects managed by CS +, do not use space codes, multibyte characters, and symbols such as "$", "#", "%" in folder names or paths to them.

(Note that depending on the version of CS+ you are using, the interface may appear somewhat different from the screenshots below.)



Start the CS+, and select [Open Existing MCU Simulator / e^2 studio / CubeSuite / High-performance Embedded Workshop / PM+ Project]

Select a project (e.g. sample_project).
Each application note has its own project name.

Select a rcpc file, and click the button [Open].

Check [Project File for e2 studio (*.rcpc)].

Select [Empty Application(CC-RX)] in [Kind of project:], and specify the project name and place, and select whether to backup.

File name: sample_project.rcpc

Microcontroller: RX

Kind of project: Empty Application(CC-RX)

Project name: sample_project

Figure 6.1 Importing a Project into CS+

7. Reference Documents

- RX110 Group User’s Manual: Hardware (R01UH0421)
- RX111 Group User’s Manual: Hardware (R01UH0365)
- RX113 Group User’s Manual: Hardware (R01UH0448)
- RX130 Group User’s Manual: Hardware (R01UH0560)
- RX140 Group User’s Manual: Hardware (R01UH0905)
- RX230 Group, RX231 Group User’s Manual: Hardware (R01UH0496)
  (The latest version can be downloaded from the Renesas Electronics website.)

- Technical Update/Technical News
  (The latest information can be downloaded from the Renesas Electronics website.)

- C compiler manual
- RX Family C/C++ compiler Package
  (The latest version can be downloaded from the Renesas Electronics website.)

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Sep.25.17	—	First edition issued
1.10	Sep.26.18	—	RX100 series added
1.20	Mar.24.22	All	Added the following to the target devices. • RX140 series Changed the specifications of the sample programs for the following reasons: To reduce the size of the firmware update program. • Removed the CMT module • Separated the constants part from the program part. To improve convenience when updating a program. • Changed firmware start-up procedure after programming is complete. Revised Table 1.2. Modified the setting items in Table 1.5. Modified URLs in “2.2 Compiler Package” and “2.3 Renesas Flash Programmer”. Updated “3. Setting Up the Project” and “4. Operation Confirmation”. Added “5.1 Configuration of the Firmware Update Program”. Changed the figures used in “5.2 Operation Overview”. Added “5.2.7 How to Create a Firmware Update Program for Updating”. Changed the figures used in “5.3 Process Flowchart and Screen Output: Firmware Update Program” and “5.4 Process Flowchart and Screen Output: Firmware Program”. Updated “5.5 Detailed Information of the Firmware Update Program” and “5.6 Detailed Information of the Firmware”. Revised “6. Import a Project”. Added “7. Reference Documents”.

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/.

© 2022 Renesas Electronics Corporation. All rights reserved.