RX Family

Firmware Update Sample Program with Dual Bank Function, and Flash Module and SCI Module Firmware Integration Technology

Introduction

This application note describes updating the on-chip code flash memory using the dual bank function in the RX Family MCUs. Serial communication is used for application software control and data transfer.

Target Devices

RX651 Group and RX65N Group	ROM capacity: 1.5 MB to 2 MB
RX72M Group	ROM capacity: 2 MB to 4 MB

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

Related Documents

RX Family Board Support Package Firmware Integration Technology Module (R01AN1685)

RX Family Flash Module Using Firmware Integration Technology (R01AN2184)

RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)

RX Family CMT Module Using Firmware Integration Technology (R01AN1856)

RX Family SCI Multi-Mode Module Using Firmware Integration Technology (R01AN1815)

Contents

1. Overview 4
1.1 About This Application Note 4
1.2 Operation Confirmation Environment 5
1.3 Module Configuration 6
1.4 File Structure 7
1.5 Project 8

2. Obtaining the Development Environment 9
2.1 e2 studio 9
2.2 Compiler Package 9
2.3 Renesas Flash Programmer 9
2.4 RSK USB Serial Driver 9

3. Importing the Project 10
3.1 Creating a Workspace 10
3.2 Importing from the Smart Browser 12
3.2.1 Importing a Project 12
3.3 Importing from the Zip File 17
3.3.1 Importing the Project into the Workspace 17
3.4 Changed Setting Information 20
3.4.1 Changes to Software component configuration 20
3.4.2 Pin configuration 25
3.4.3 Changes to “C/C++ Build” Settings 26

4. Operation Confirmation 29
4.1 Building the Project 29
4.2 Preparing Debugging 30
4.2.1 Preparing Devices 30
4.2.2 Host PC Settings 31
4.2.3 Presetting the MCU 32
4.3 Debug Configuration 35
4.4 Debugging 41
4.5 mot File for Update Confirmation 43

5. Overview of the Sample Program 44
5.1 Operation Overview 44
5.1.1 Updating the Firmware 45
5.2 Process Flowchart and Screen Output 48
5.2.1 Main Processing 48
5.2.2 Firmware Update Processing 50
5.2.3 Changing the Startup Bank Processing 52

5.3 Sample Program 54
5.3.1 File Composition 54
5.3.2 Constants 55
5.3.3 Type Definitions 57
5.3.4 Variables 60
5.3.5 Functions 63

6. Appendices 65
6.1 Troubleshooting 65
6.1.1 Operation Terminated Abnormally when Programming a mot File with the Renesas Flash Programmer 65
6.1.2 File Transfer Canceled During XMODEM Transfer 65
6.1.3 Nothing Displayed Even Though XMODEM Transfer Completed 65
6.1.4 FIT Module Setting Items Not Displayed in Software component configuration 66
6.1.5 SCI FIT Module Not Displayed in “New Component” Dialog Box 71
6.2 FAQ 73
6.2.1 Q: How do I change the device? 73
6.2.2 Q: How can I change the endianness to big endian? 80
6.2.3 Q: How can I change the interface to an interface other than the SCI? 82
6.2.4 Q: How can I erase the option-setting memory? 82
6.2.5 Q: How can I specify to read the other bank of the startup bank? 83
6.2.6 Q: How can I check the bank currently specified as the startup bank? 83
6.2.7 Q: How can I import the sample program into CS+? 84

7. Others 85
7.1 Notes on Using the Evaluation Version of C/C++ Compiler Package for RX Family 85

Revision History 86

1. Overview

1.1 About This Application Note

This application note describes the method to safely update the code flash memory using the dual bank function. The MCU 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 serial communication software on the host PC must be capable of XMODEM/SUM transfer.

This application note is accompanied by two versions of the sample program, one for RX65N Group and RX651 Group MCUs with 2 MB of ROM, and one for RX72M Group MCUs with 4 MB of ROM. To use a different device, refer to 6.2.1, Q: How do I change the device?

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
Compare match timer	Timer for checking serial communication timeout

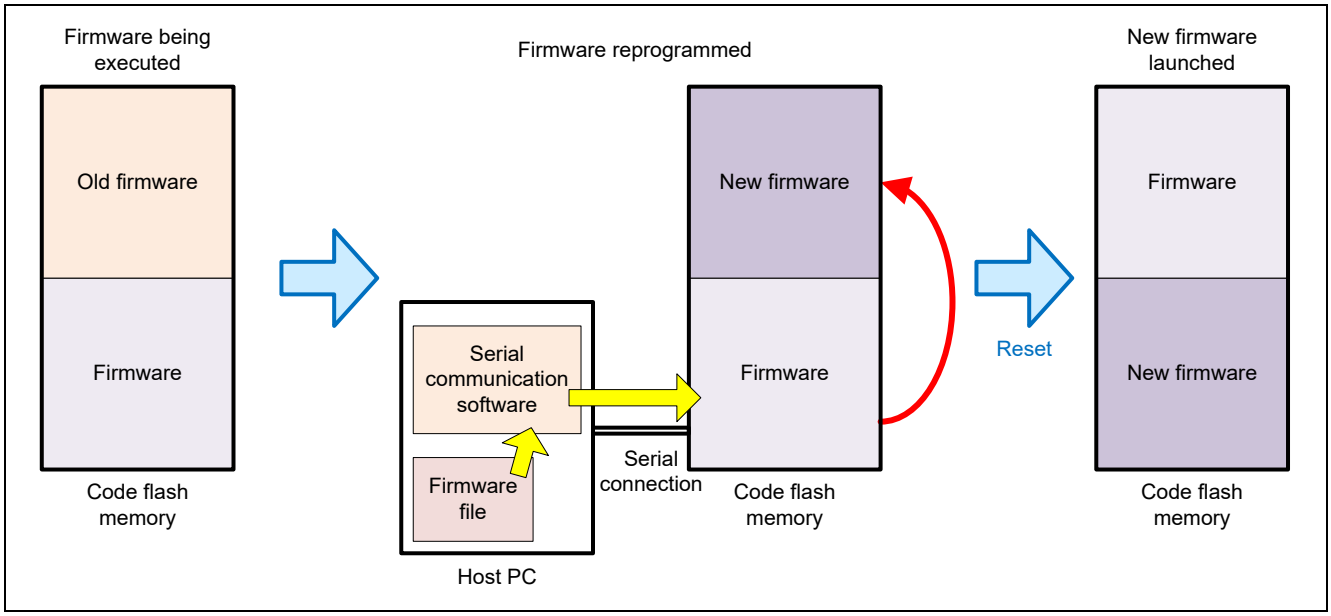


Figure 1.1 Operation Overview

1.2 Operation Confirmation Environment

The operation of the sample program in this application note has been confirmed under the following conditions.

Table 1.2 Operation Confirmation Conditions

Item	Contents
MCU used	R5F565NEDDFC (RX65N Group)
Operating frequency	Main clock: 24 MHz PLL: 240 MHz (main clock ×1/1 ×10) System clock (ICLK): 120 MHz (PLL ×1/2) Flash interface clock (FCLK): 60 MHz (PLL ×1/4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V.3.01.00
	Compiler option -lang = c99
iodefine.h version	Version 2.3
Endian	Little endian, big endian*1
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample program version	Version 1.20
Flash programmer	Renesas Flash Programmer V3.06.00
Board used	Renesas Starter Kit+ for RX65N-2MB (product No.: RTK50565N2C01001BR) (hereinafter referred to as RSK+RX65N-2MB)

Item	Contents
MCU used	R5F572MNDDBD (RX72M Group)
Operating frequency	Main clock: 24 MHz PLL: 240 MHz (main clock ×1/1 ×10) System clock (ICLK): 240 MHz (PLL ×1/1) Flash interface clock (FCLK): 60 MHz (PLL ×1/4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V.3.01.00
	Compiler option -lang = c99
iodefine.h version	Version 1.0C
Endian	Little endian, big endian*1
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample program version	Version 1.20
Flash programmer	Renesas Flash Programmer V3.06.00
Board used	Renesas Starter Kit+ for RX72M (product No.: RTK5572MNDC00000BJ) (hereinafter referred to as RSK+RX72M)

Note: 1. The endianness used in the sample program is little endian. To change the endianness to big endian, refer to 6.2.2, Q: How can I change the endianness to big endian?.

1.3 Module Configuration

Figure 1.2 shows the module configuration of the dual bank firmware update SCI project and Table 1.3 lists the FIT modules implemented in the project.

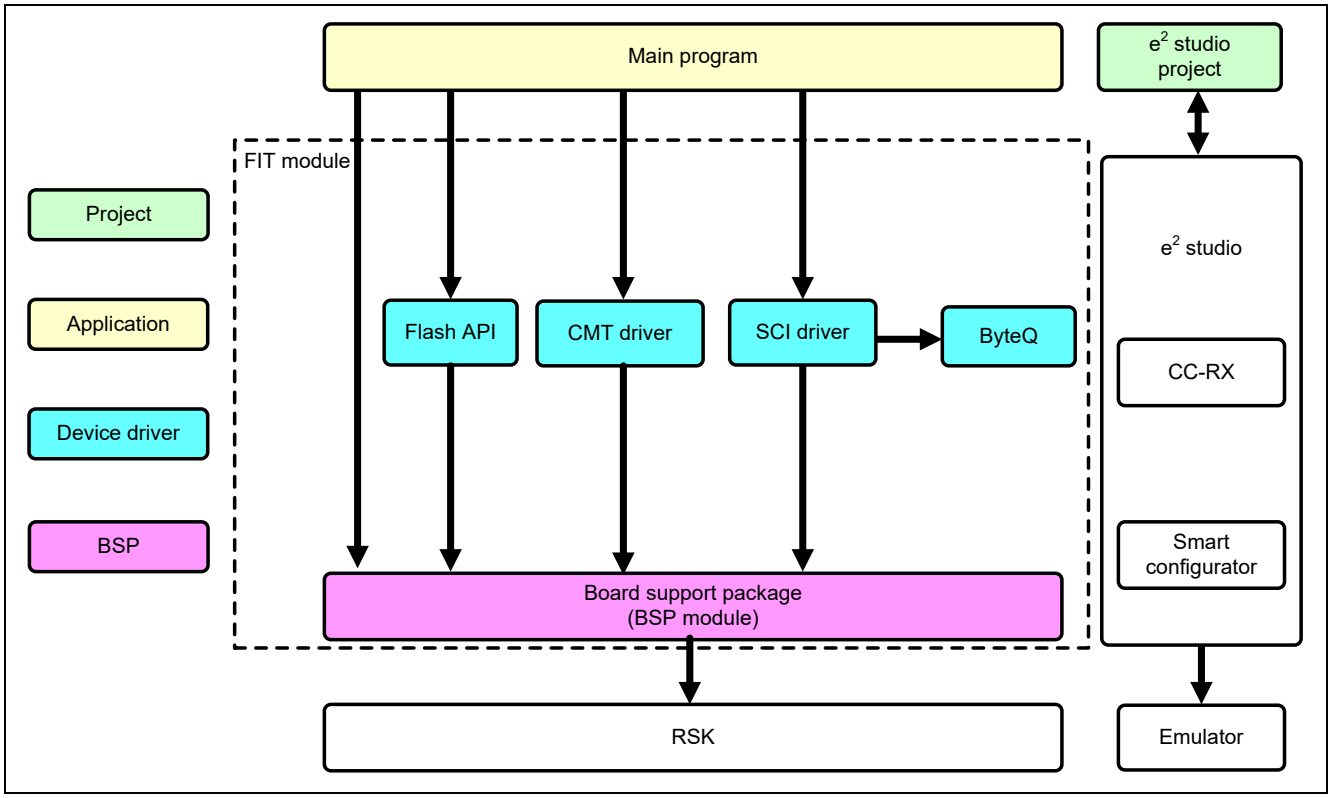


Figure 1.2 Module Configuration

Table 1.3 Module List

Category	Application Note (Document No.)	FIT Module Name	Rev.
BSP	RX Family Board Support Package Firmware Integration Technology Module (R01AN1685)	r_bsp	5.2
Device driver	RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)	r_byteq	1.60
	RX Family CMT Module Using Firmware Integration Technology (R01AN1856)	r_cmt_rx	3.20
	RX Family Flash Module Using Firmware Integration Technology (R01AN2184)	r_flash_rx	4.20
	RX Family SCI Multi-Mode Module Using Firmware Integration Technology (R01AN1815)	r_sci_rx	2.0
Application	Main program	—	1.20

1.4 File Structure

Figure 1.3 shows the file structure of this application note.

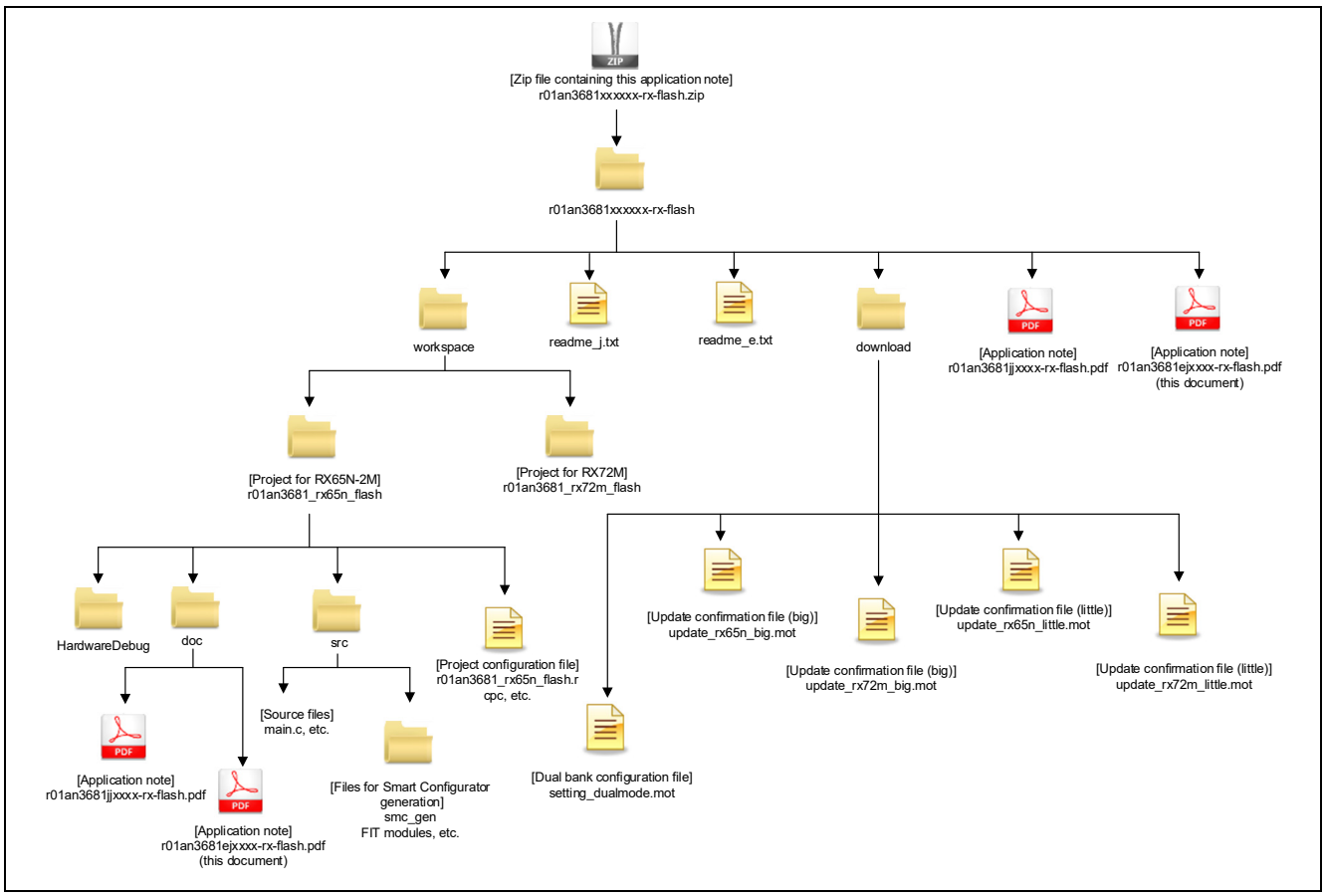


Figure 1.3 File Structure

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.

The “r01an3681_rx65n_flash” and “r01an3681_rx72m_flash” folders contain the project files for building the sample program described in this application note. The operation of the sample program can be confirmed by importing the appropriate project into the workspace of e^2 studio.

1.5 Project

This application note includes the e^2 studio project for building and evaluating the sample program. The project has 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.

Table 1.4 Project Configuration

Configuration Example		Description
Build configuration	HardwareDebug (Debug on hardware)	Configuration to generate a load module with debug information
	Release (Release - No Debug)	Configuration for release without debug information
Debug configuration	r01an3681_rx65n_flash HardwareDebug	Performs hardware debugging via E2 emulator Lite using the load module generated with HardwareDebug (Debug on hardware).
	r01an3681_rx72m_flash HardwareDebug	

The following table lists target specific settings.

Table 1.5 Target Specific Settings

Item	Setting
Toolchain version	V3.01.00
Debug hardware	E2 emulator Lite
Endianness	Little-endian data
Target selection	R5F565NEDxFC_DUAL
	R5F572MNDxBD_DUAL
Renesas RTOS support	None

2. Obtaining the Development Environment

2.1 e² studio

Visit the following URL and download the e² studio.

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

This document assumes that V7.5.0 or later version of e² studio is used. If a version earlier than V7.5.0 is used, some features of e² studio may not be supported. Make sure to download the latest version of e² studio on the website.

2.2 Compiler Package

Visit the following URL and download the RX Family C/C++ compiler package.

https://www.renesas.com/en-us/products/software-tools/tools/compiler-assembler/compiler-package-for-rx-family.html

2.3 Renesas Flash Programmer

Visit the following URL and download the Renesas flash programmer.

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

2.4 RSK USB Serial Driver

Visit the following URL and download the RSK USB serial driver.

Refer to the RSK user's manual for details on the installation.

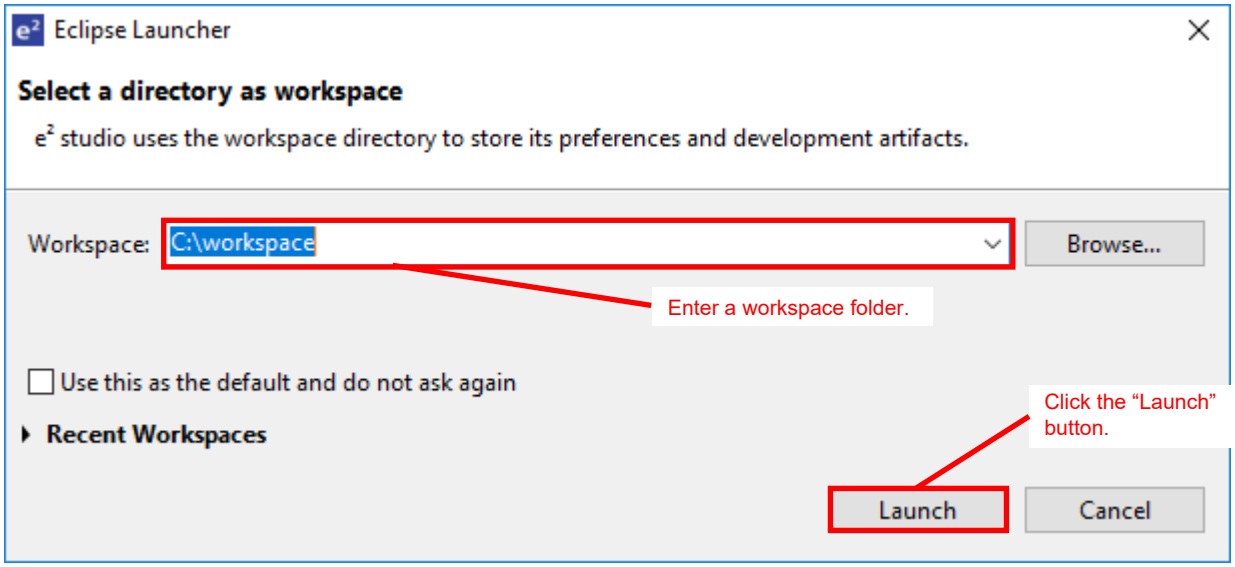
https://www.renesas.com/en-us/software/D6000699.html

3. Importing the Project

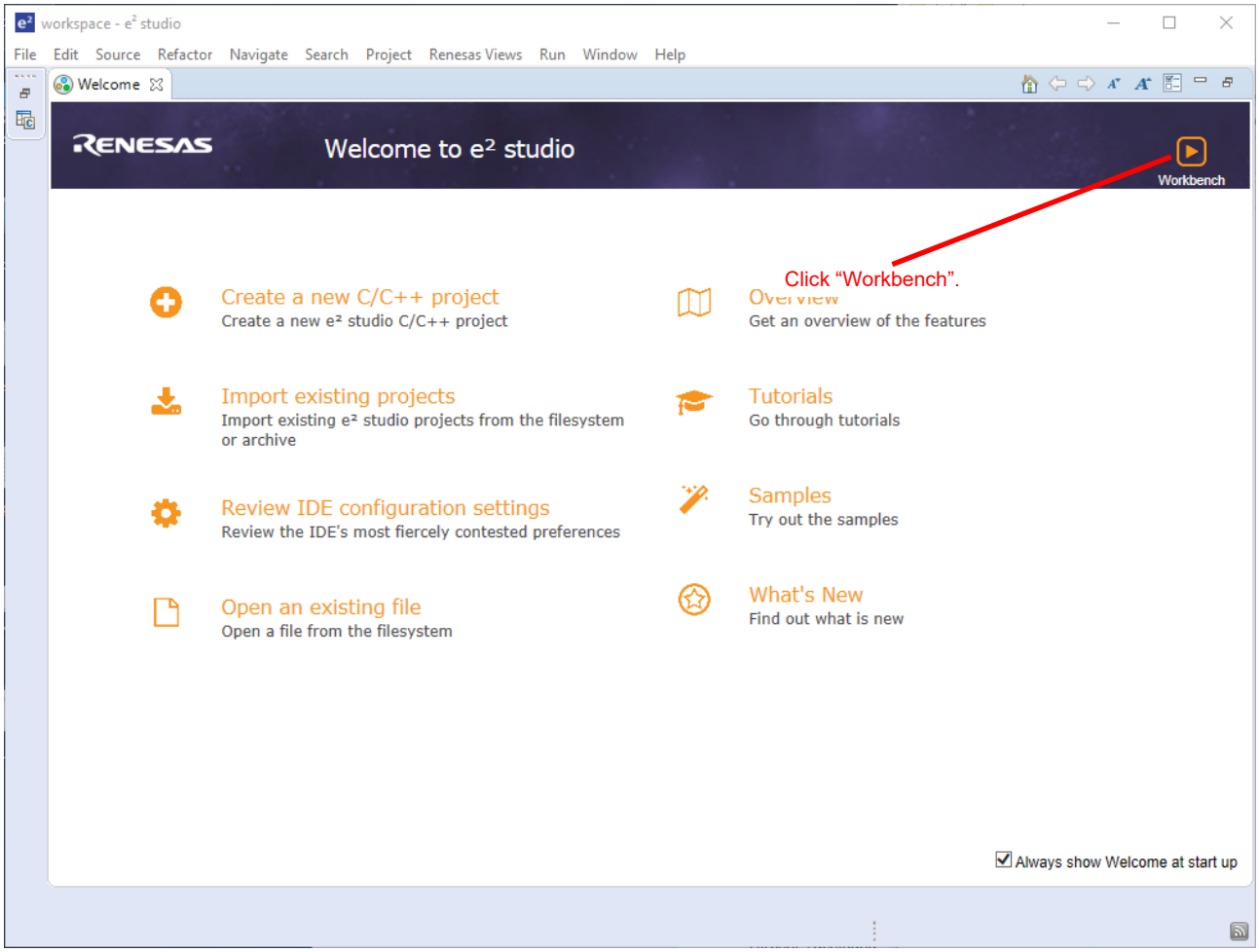
This section describes the procedure to import the project, for building the sample program, into the e² studio.

3.1 Creating a Workspace

1. Start the e² studio. The dialog to select a workspace opens. If the dialog does not open, select “File” >> “Switch workspace” >> “Others”.
2. Enter a workspace and click the “Launch” button. If the specified folder is not currently a workspace, then a new workspace is created.



3. If the workspace is new, the Welcome dialog opens. Click “Workbench”.

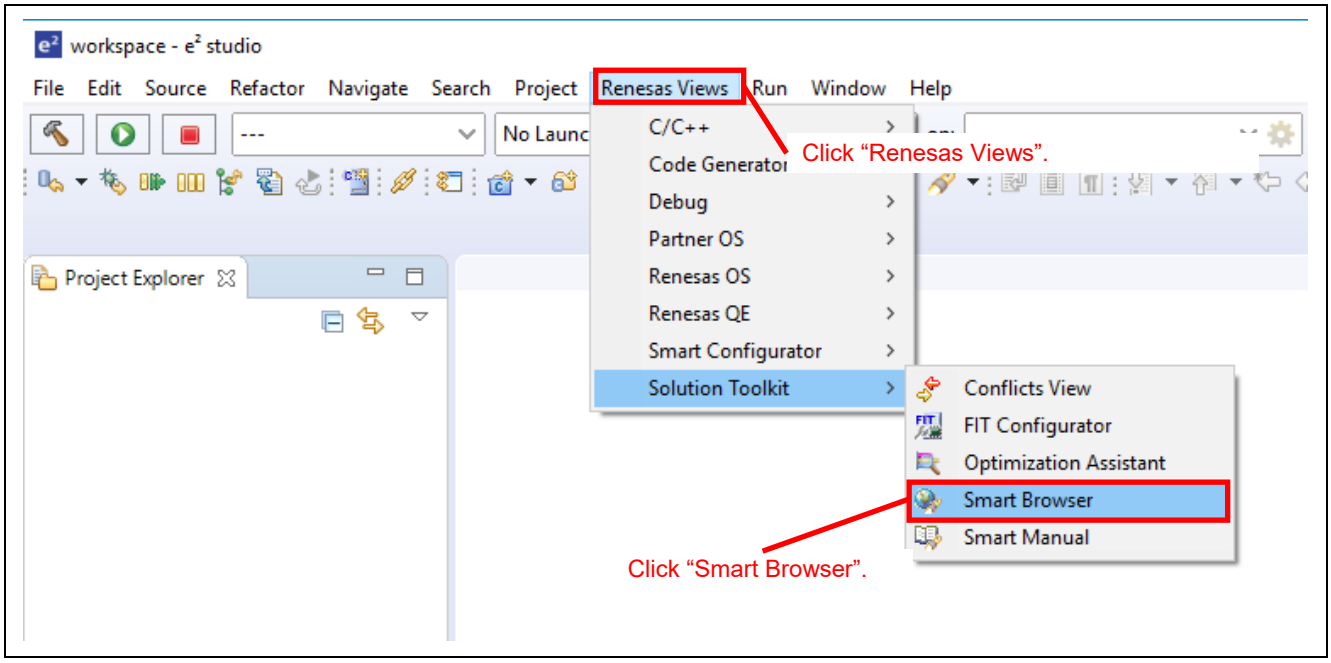


3.2 Importing from the Smart Browser

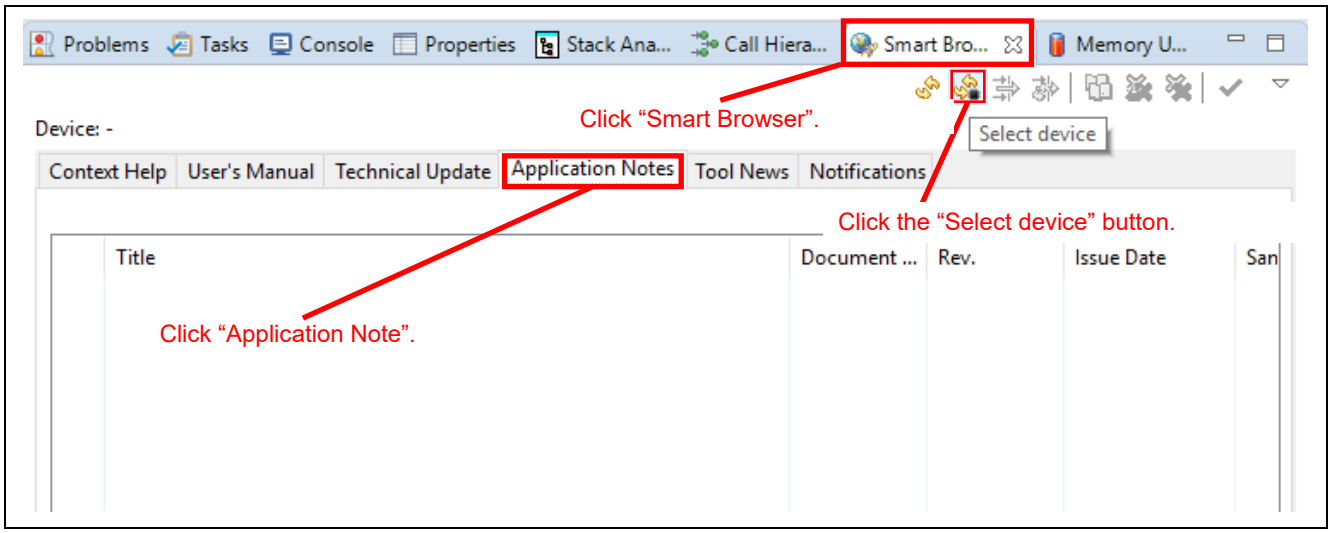
This section describes the procedure to import the project using the Smart Browser in the e² studio.

3.2.1 Importing a Project

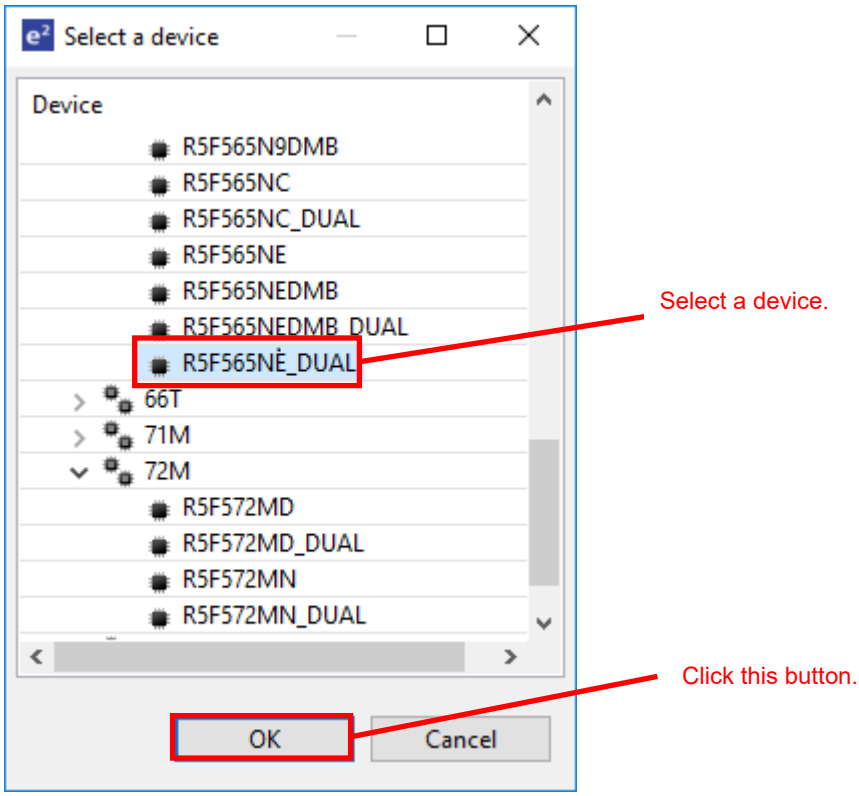
1. Click the “Smart Browser” tab to switch to it.
2. If the “Smart Browser” tab is not displayed, click “Renesas Views” on the main menu bar. Then select “Solution Toolkit” >> “Smart Browser” from the menu that appears.



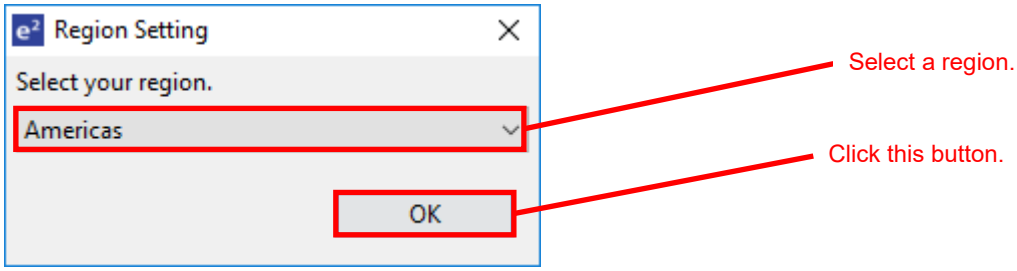
3. Click the “Application Note” tab to switch to it.
4. Click the “Select device” button.



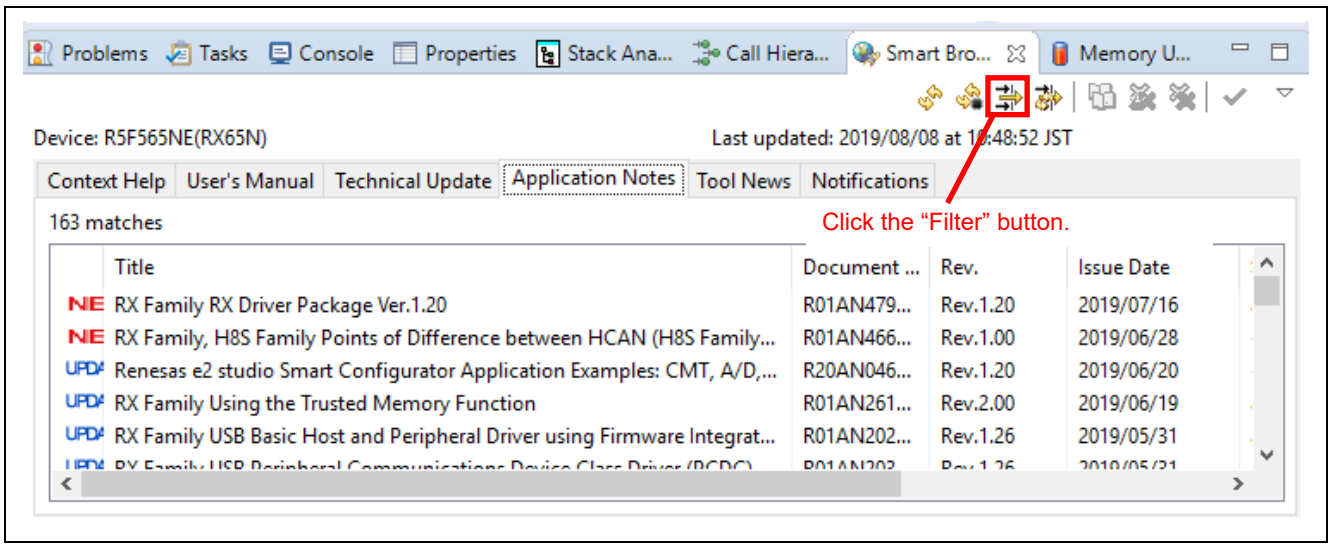
5. The “Select a device” dialog box is displayed. Select the device you wish to use.
6. Click the “OK” button.



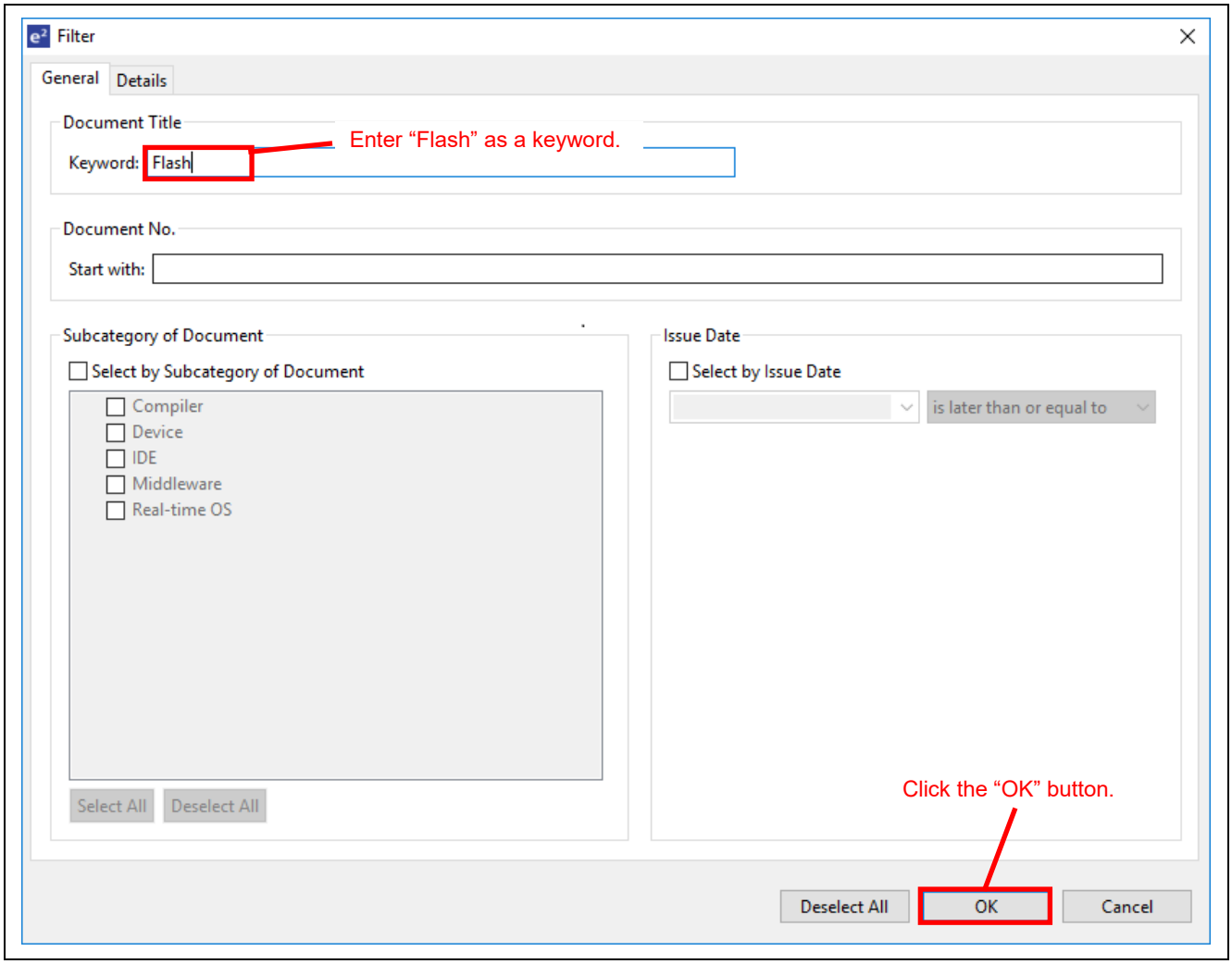
7. If the Smart Browser is being opened for the first time, the “Region Setting” dialog box is displayed.
8. Select the region where you are located.
9. Click the “OK” button.



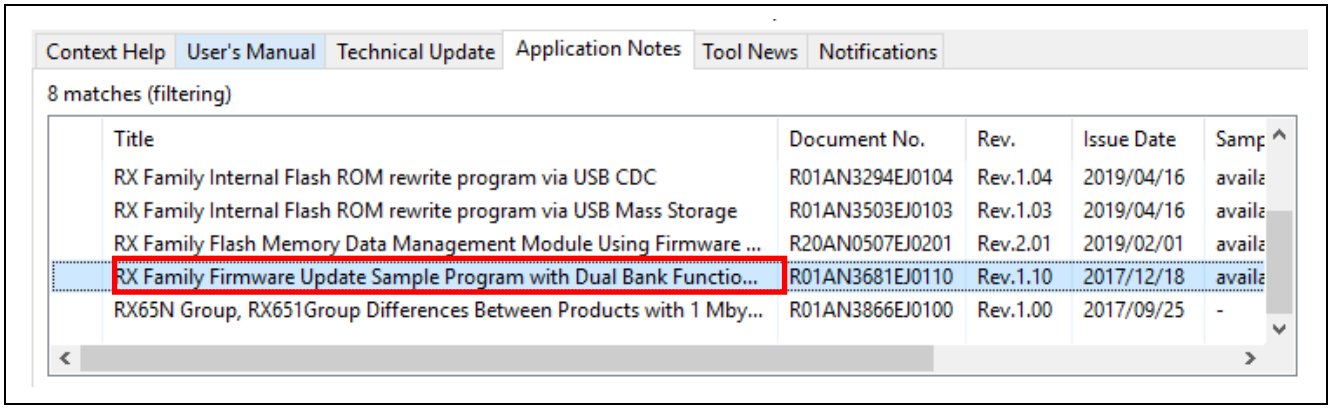
10. After a few moments, a list of application notes is displayed.
11. Click the “Filter” button.



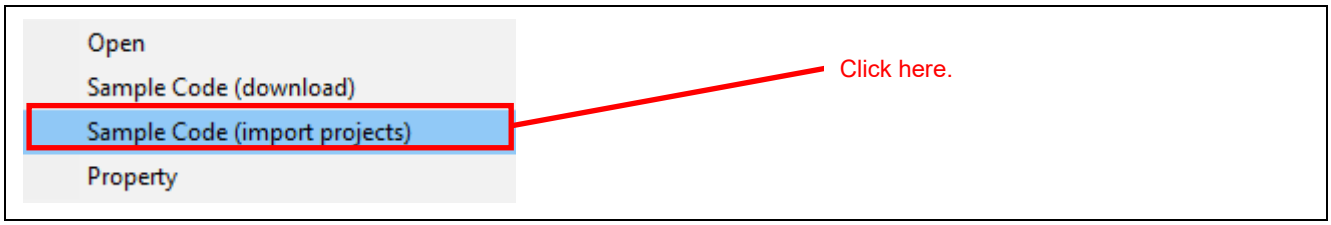
12. Enter “Flash” as a keyword under Document Title.
13. Click the “OK” button.



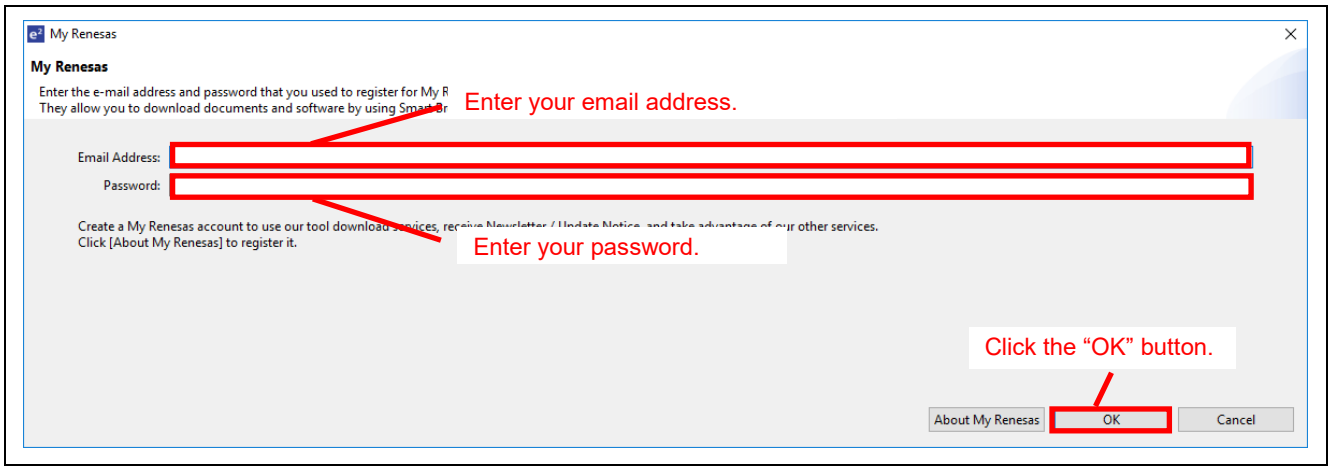
14. Select this application note, and right-click on it.



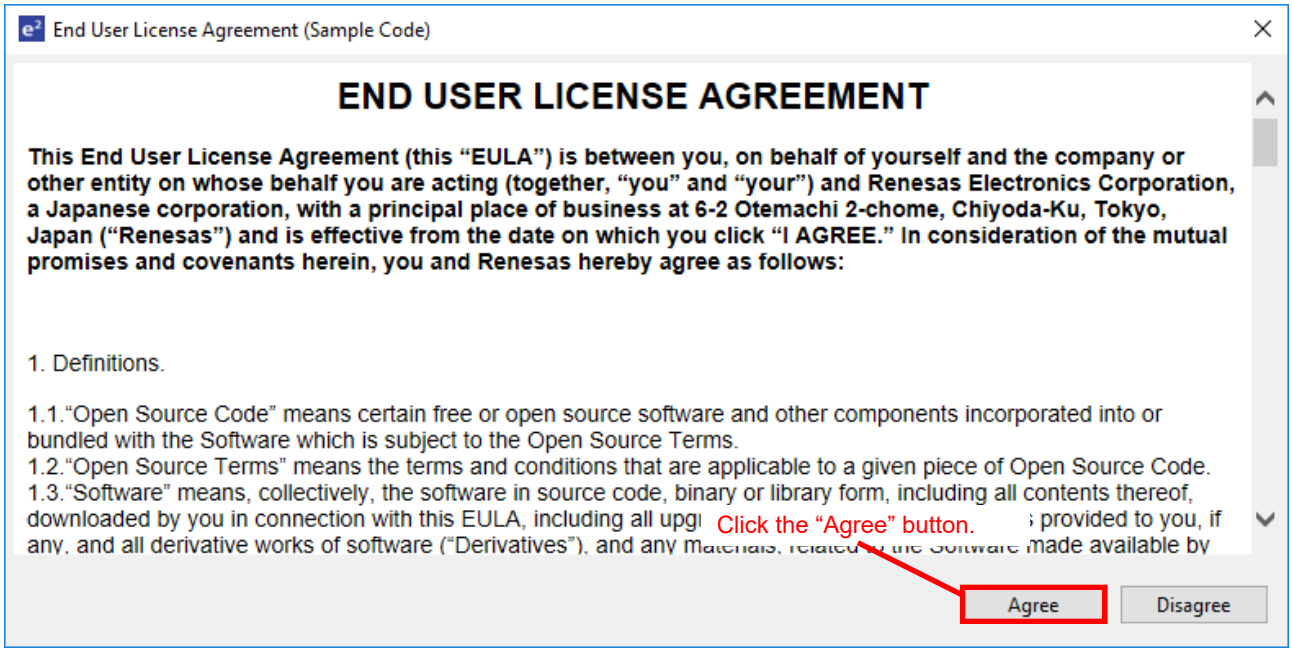
15. On the context menu, click “Sample Code (import project)”.



16. A file save dialog box appears. Select the save location and click the “Save” button.
17. If your computer is not authenticated with My Renesas, the “My Renesas” dialog box appears before the file download occurs. Enter the email address and password you have registered on the Renesas website.
18. Click the “OK” button.



19. To accept the terms of the disclaimer, click the “Agree” button.



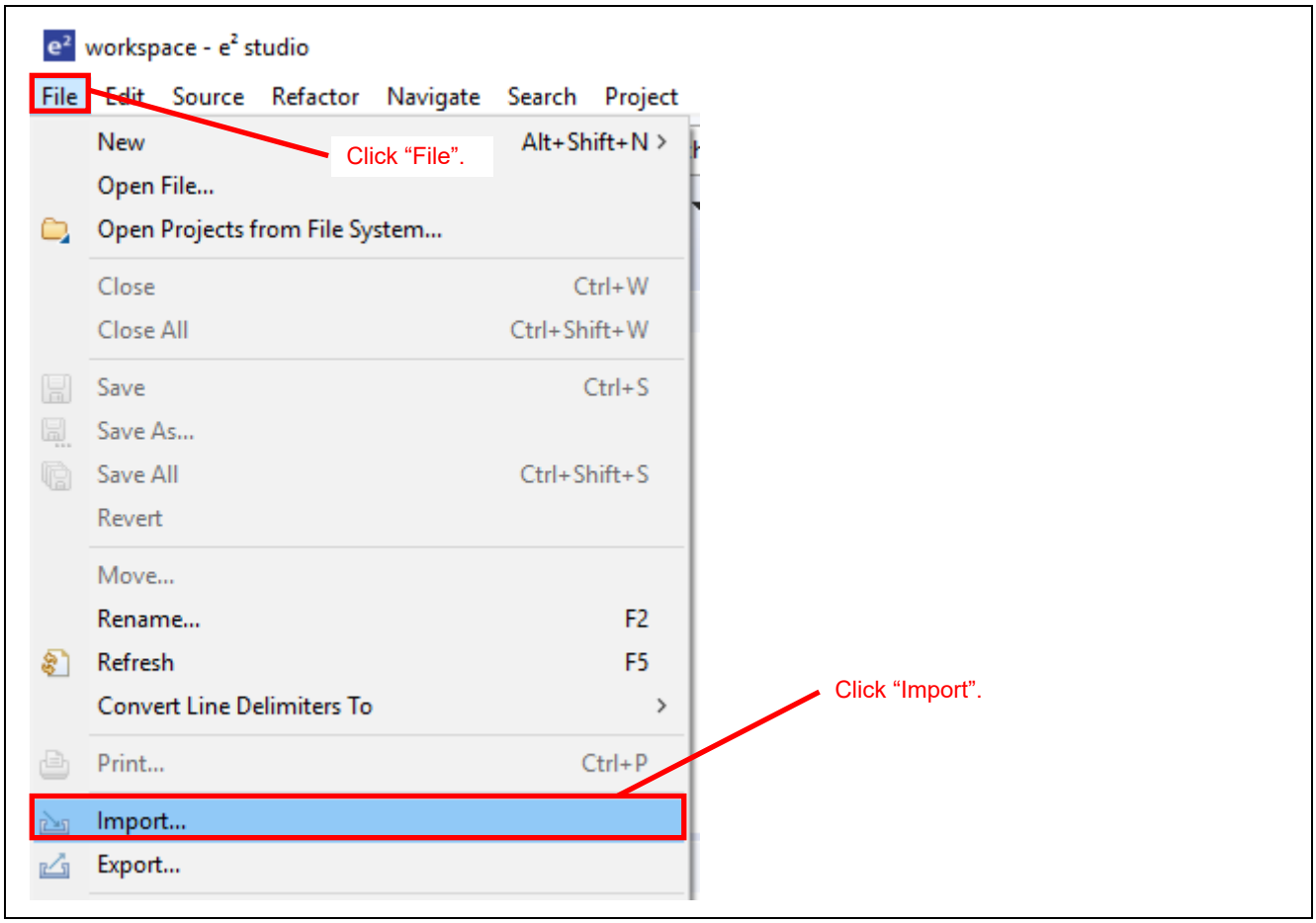
20. The “Import” dialog box appears. Select either “r01an3681_rx65n_flash” or “r01an3681_rx72m_flash”, displayed under “Projects:”, whichever matches your device.
21. Click the “Finish” button.

3.3 Importing from the Zip File

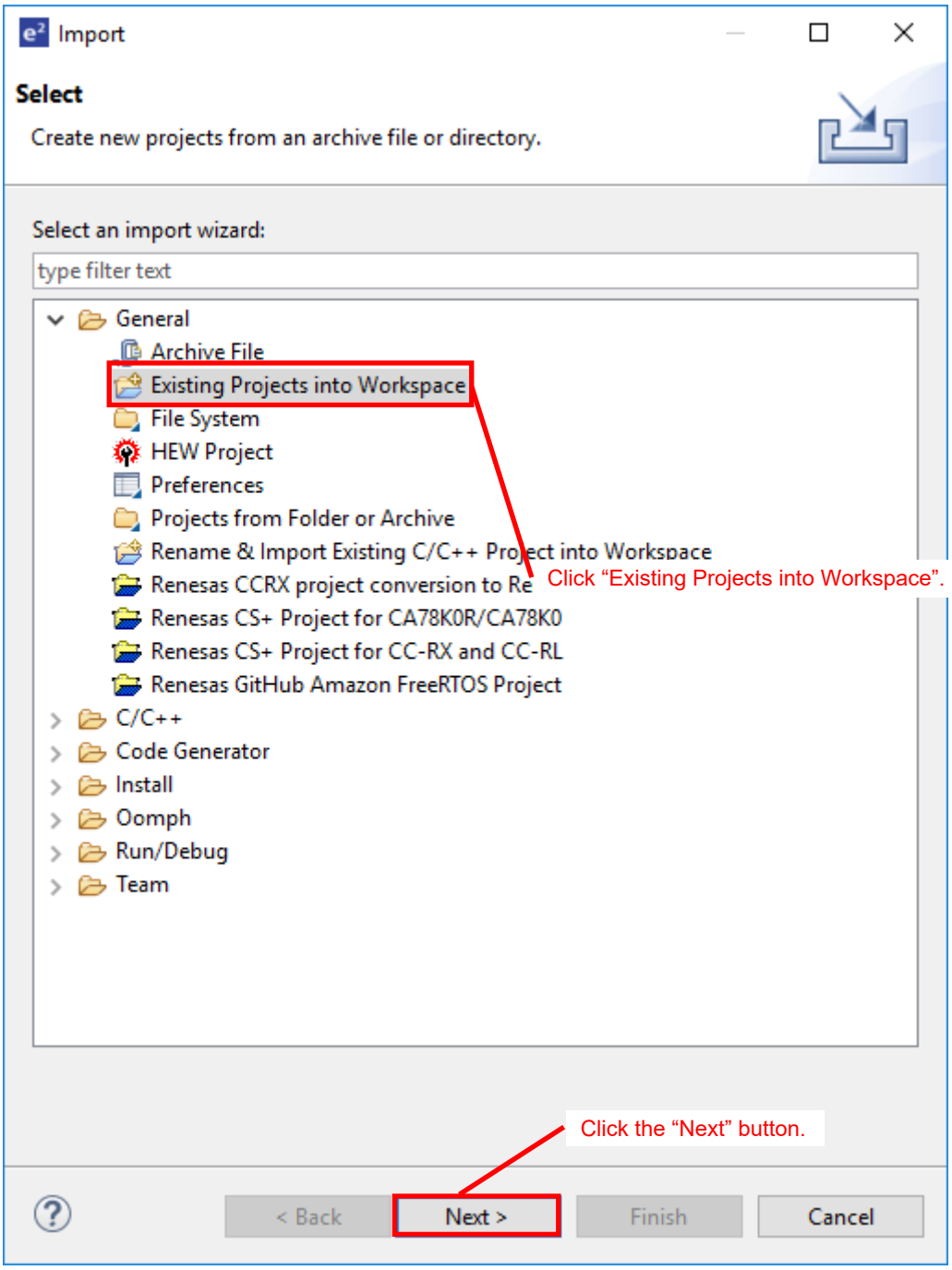
The project can be imported from the accompanying zip file of this application note.

3.3.1 Importing the Project into the Workspace

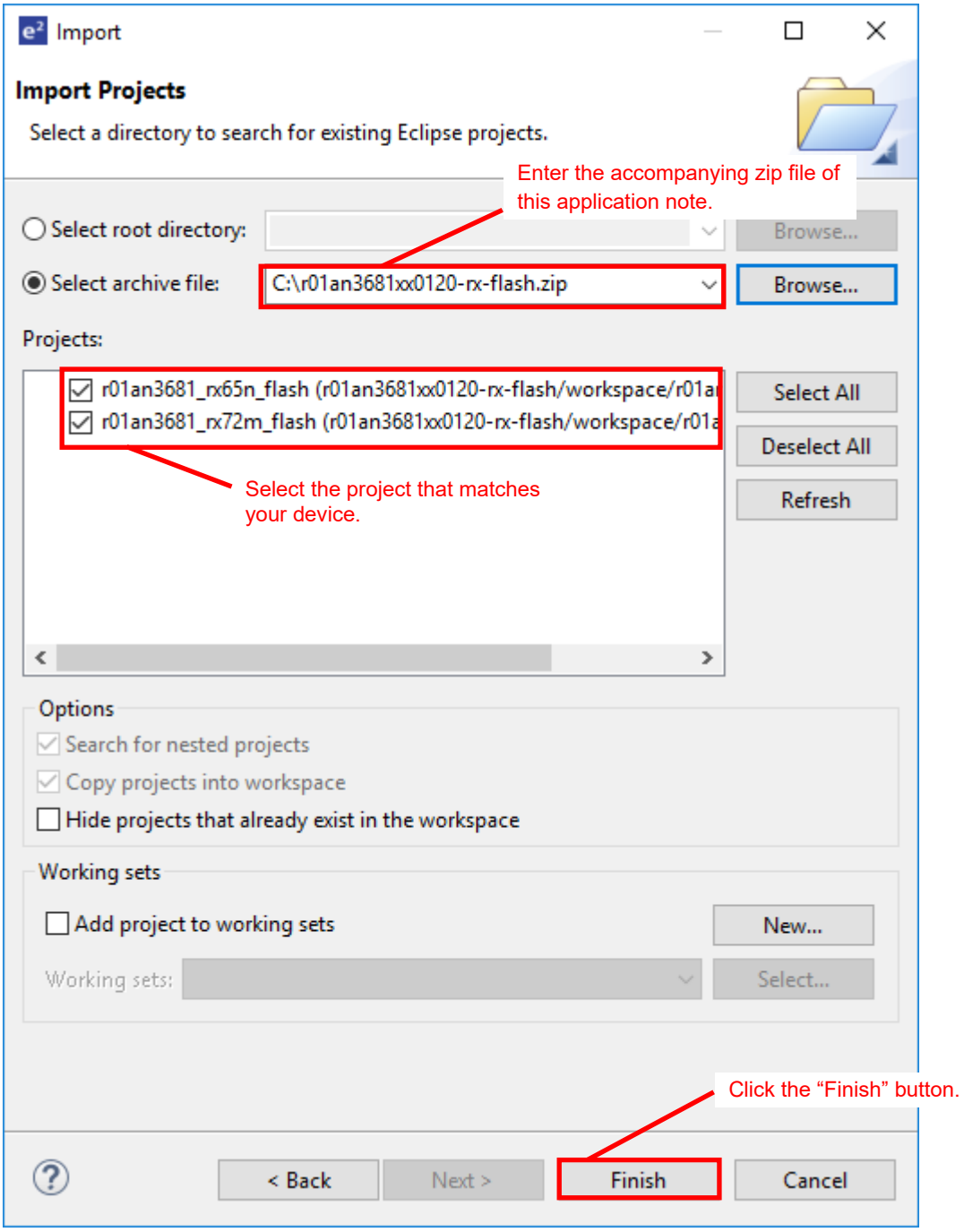
1. Select “File” >> “Import”.



2. Select “Existing Projects into Workspace” under “General”.
3. Click the “Next” button.



4. Enter the accompanying zip file of this application note in the combo box for the “Select archive file” section.
5. Select either “r01an3681_rx65n_flash” or “r01an3681_rx72m_flash”, displayed under “Projects:”, whichever matches your device.
6. Click the “Finish” button.



3.4 Changed Setting Information

In the project of the sample program, changes have been made to the software component configuration and “C/C++ Build” settings. Also, pin settings are performed. The details are described below.

The information provided in this section can be referred when setting up a new project. When using the imported project, go to 4, Operation Confirmation.

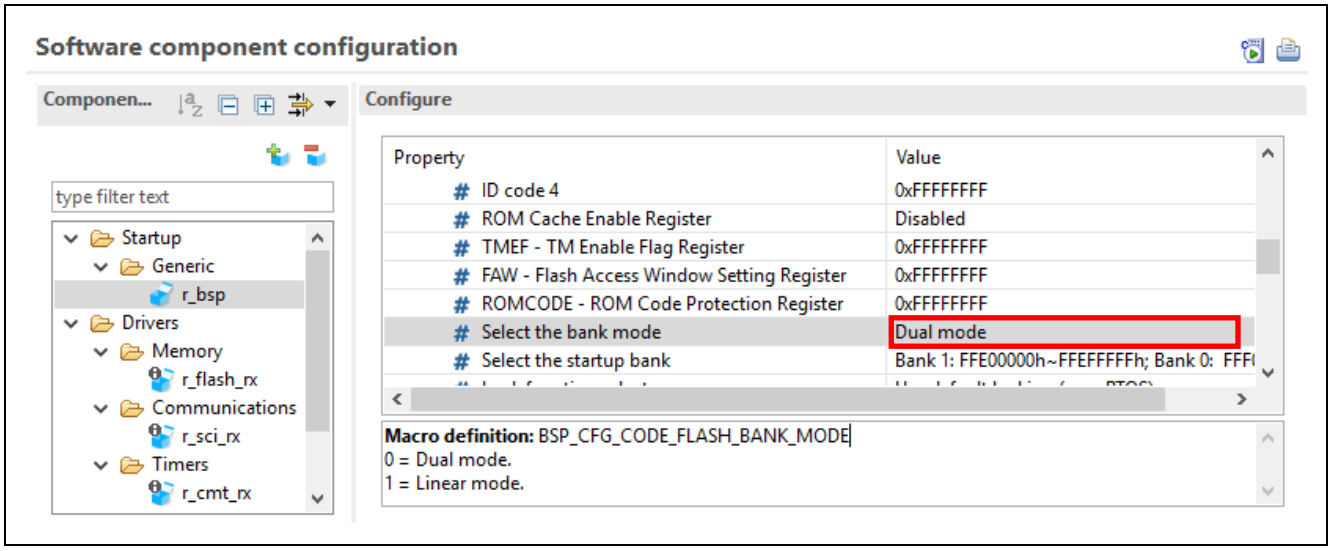
3.4.1 Changes to Software Component Configuration

(1) Changes to BSP Configuration

[src/smc_gen/r_config/r_bsp_config.h]

Startup >> Generic >> r_bsp

Change the bank mode to “Dual mode”.



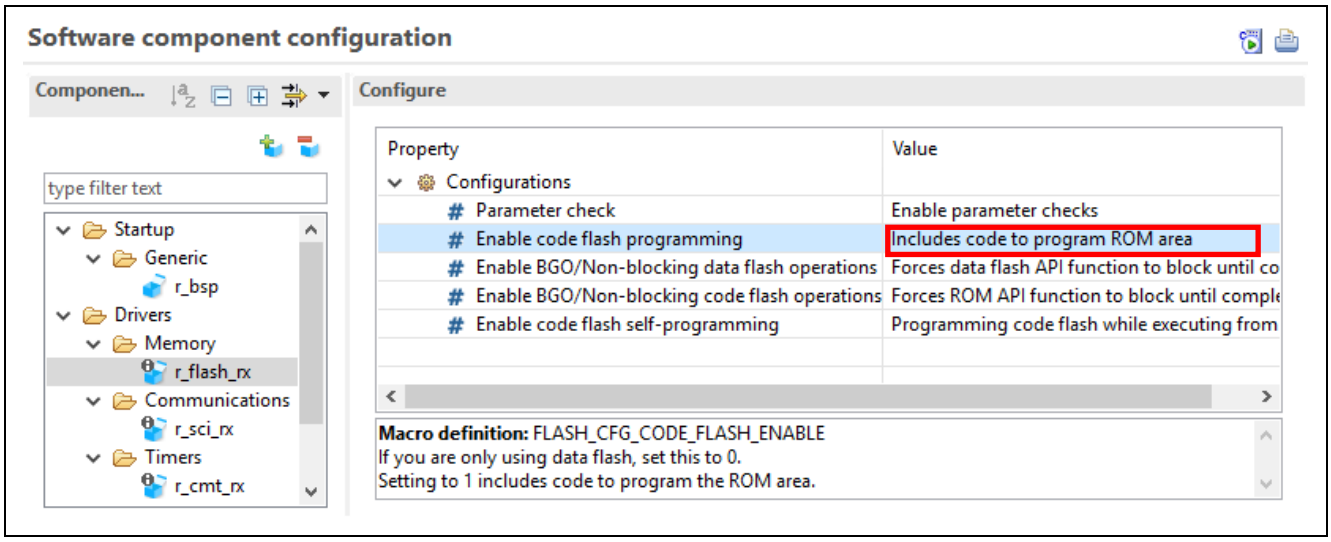
(2) Changes to Flash API Configuration

The following Flash API configuration are changed.

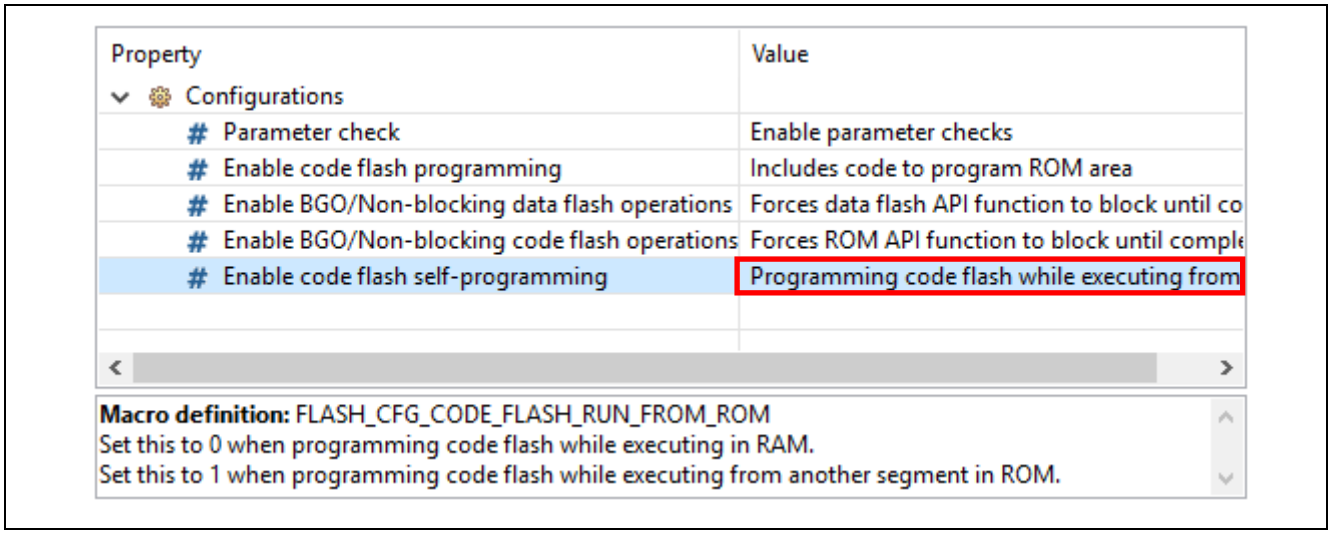
[src/smc_gen/r_config/r_flash_rx_config.h]

Drivers >> Memory >> r_flash_rx

The following change is made to allow the Flash API to program the code flash memory.



The following change allows the program to be run from the ROM.

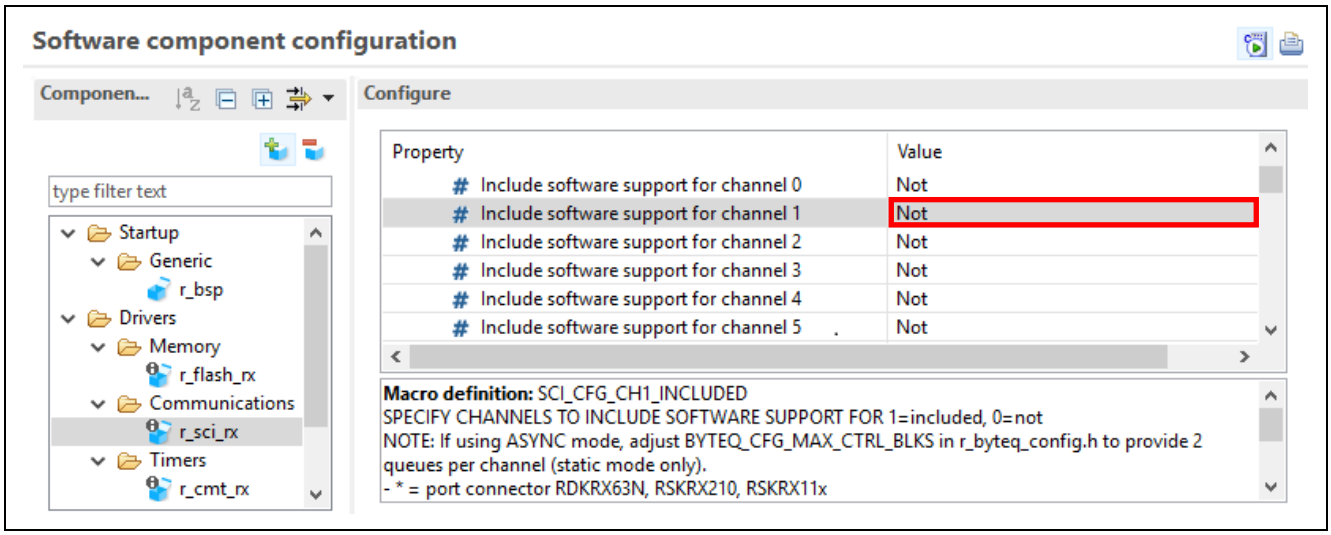


(3) Changes to SCI API Configuration

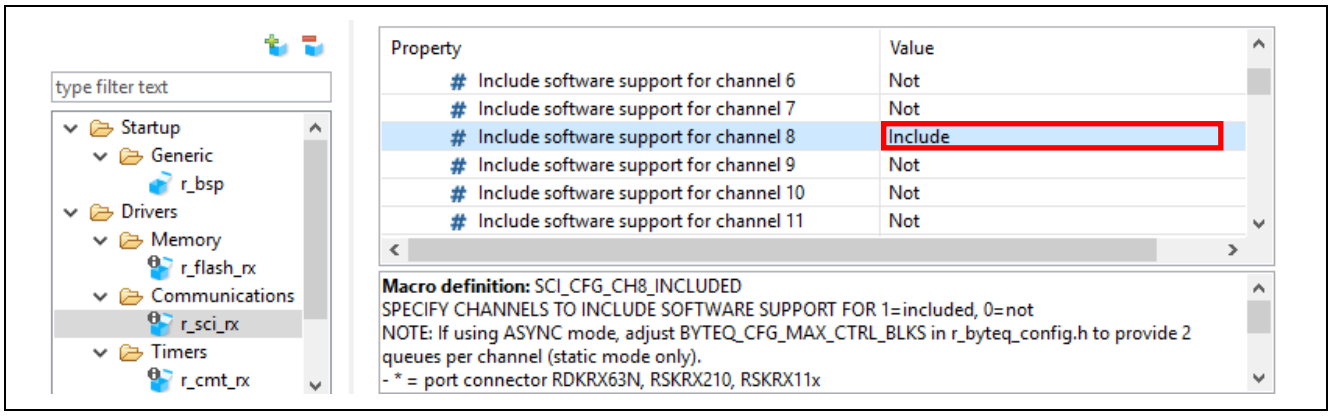
[src/smc_gen/r_config/r_sci_rx_config.h]

Drivers >> Communication >> r_sci_rx

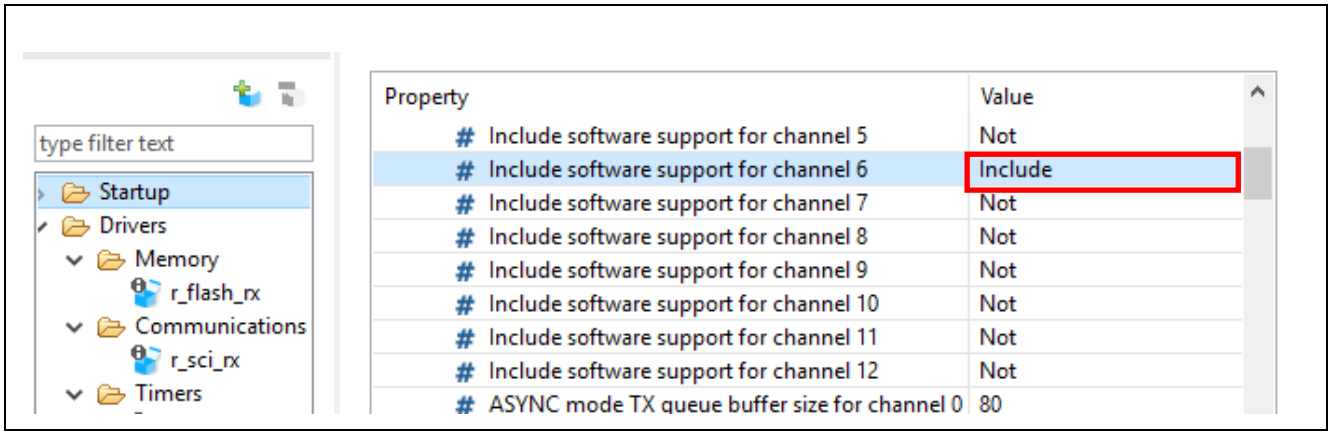
The SCI channel is changed to enable use of the USB serial port of the RSK+. Change the configuration for channel 1 to “Not”.



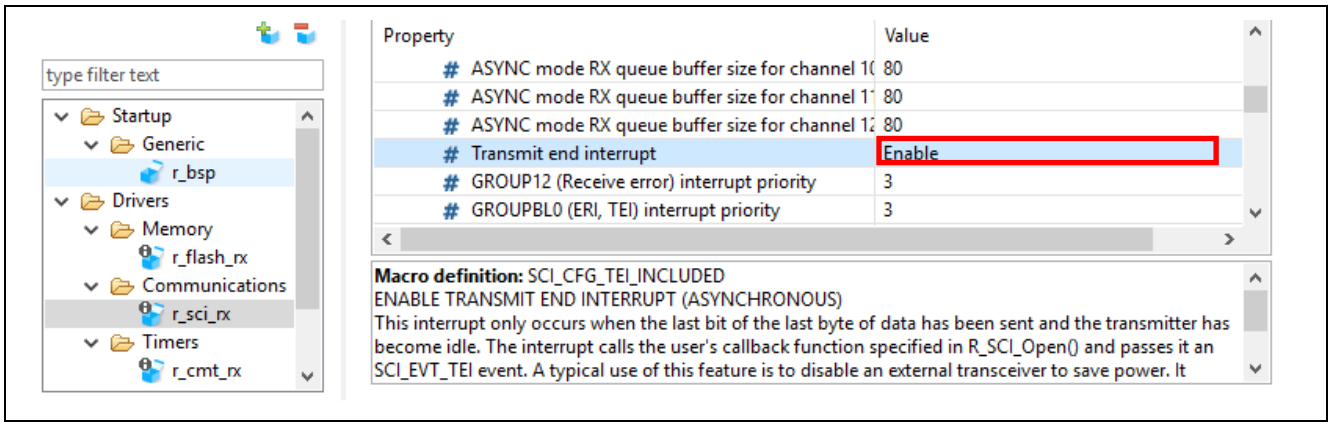
On the RX65N, change the configuration for channel 8 to “Include”.



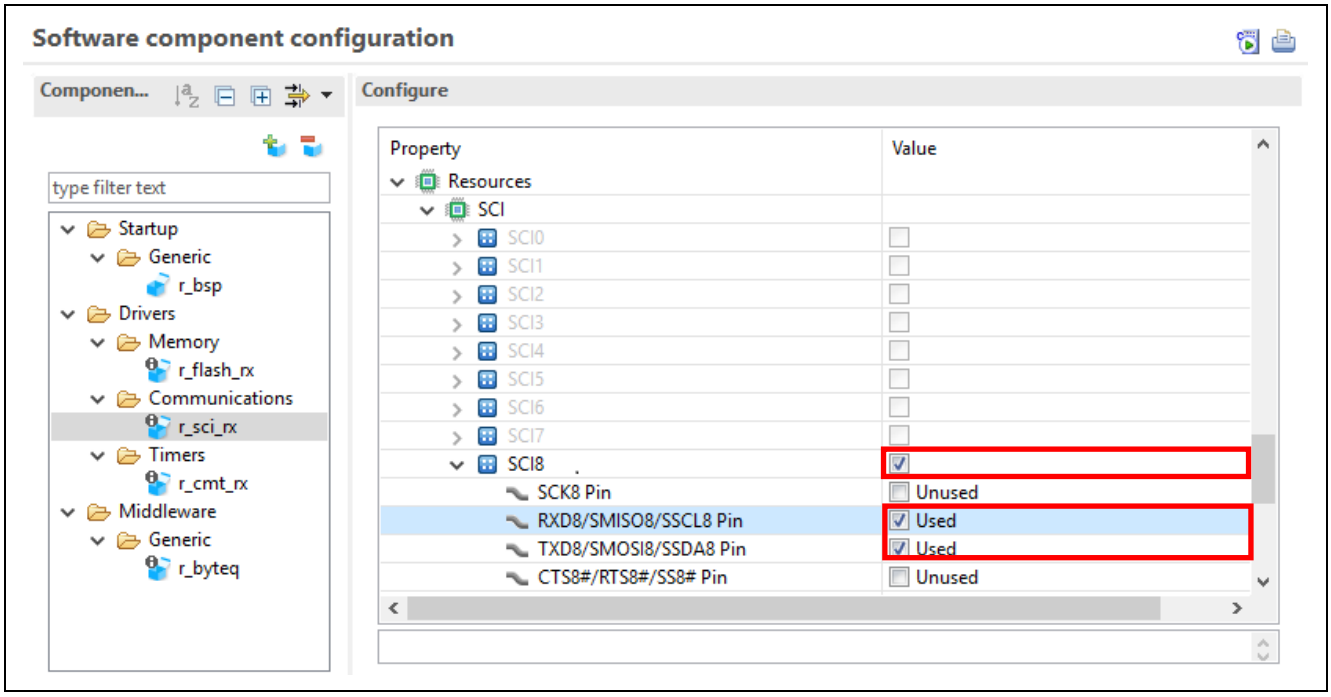
On the RX72M, change the configuration for channel 6 to “Include”.



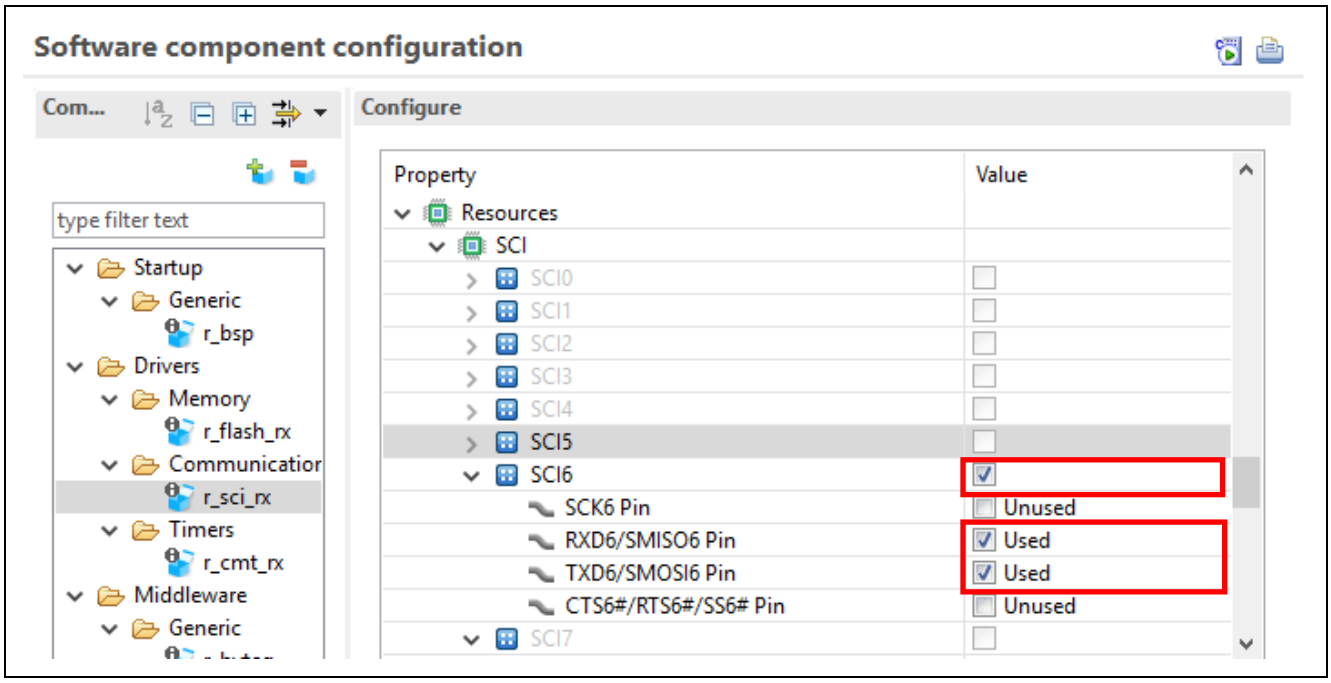
Change the transmit end interrupt configuration to “Enable”.



On the RX65N, change the settings for RXD8 and TXD8 under Resources to “Use”.



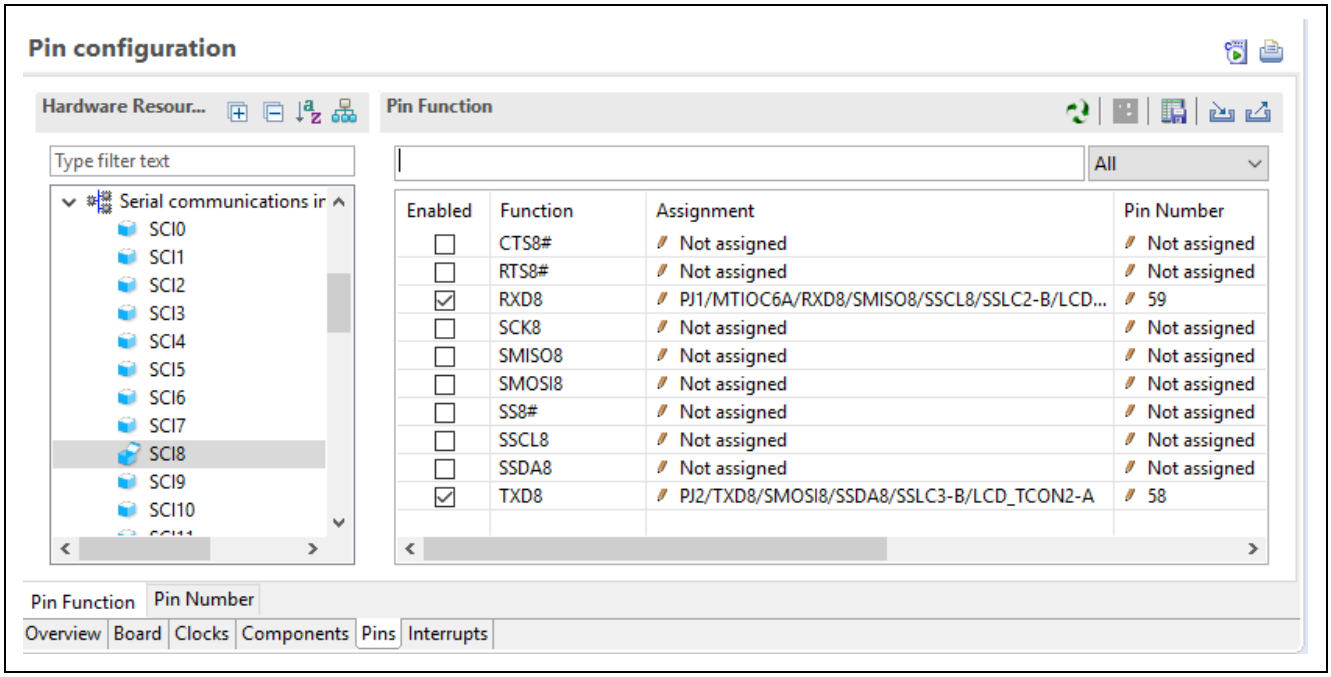
On the RX72M, change the configuration for RXD6 and TXD6 under Resources to “Use”.


Software component configuration
Com...
Configure
type filter text
Startup
Generic
r_bsp
Drivers
Memory
r_flash_rx
Communicatior
r_sci_rx
Timers
r_cmt_rx
Middleware
Generic
Property
Value
Resources
SCI
SCI0
SCI1
SCI2
SCI3
SCI4
SCI5
SCI6
SCK6 Pin
Unused
RXD6/SMISO6 Pin
Used
TXD6/SMOSI6 Pin
Used
CTS6#/RTS6#/SS6# Pin
Unused
SCI7

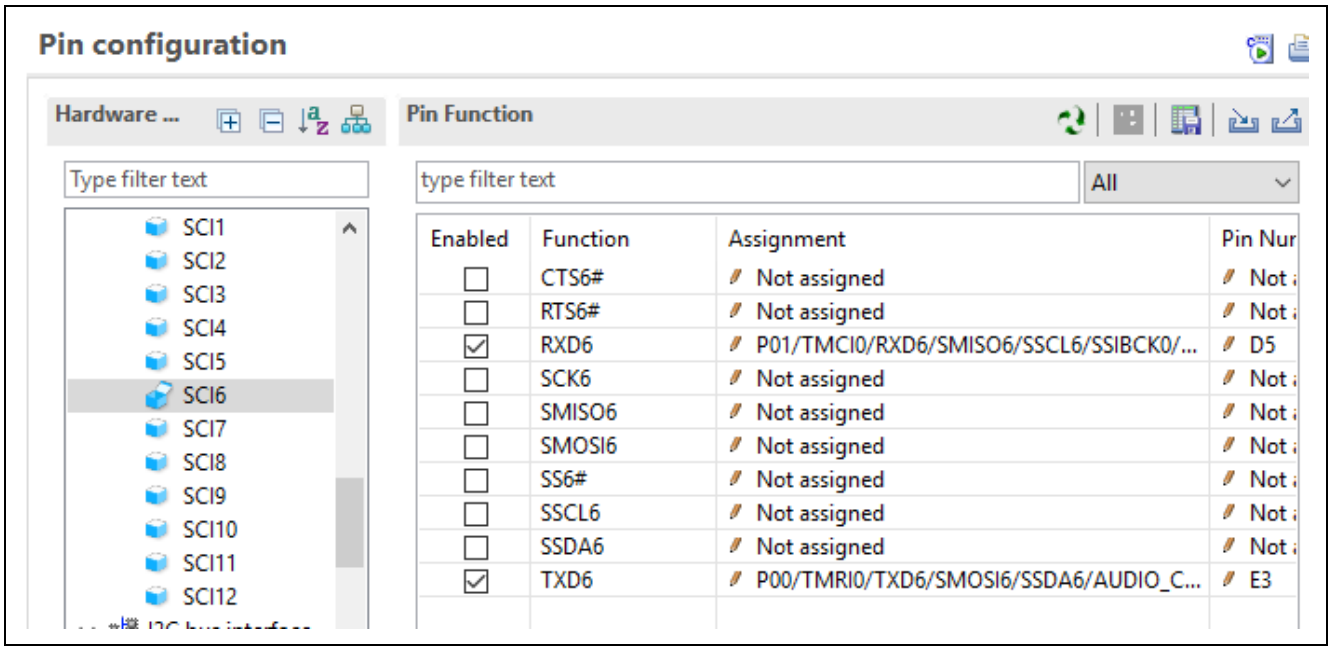
3.4.2 Pin Configuration

Click the “Pins” tab to display the “Pin configuration” screen.

On the RX65N, the pins assigned to RXD8 and TXD8 are specified. When RXD8 and TXD8 are set to “Use” under Resources, the corresponding boxes are checked automatically.



On the RX72M, the pins assigned to RXD6 and TXD6 are specified. When RXD6 and TXD6 are set to “Use” under Resources, the corresponding boxes are checked automatically.

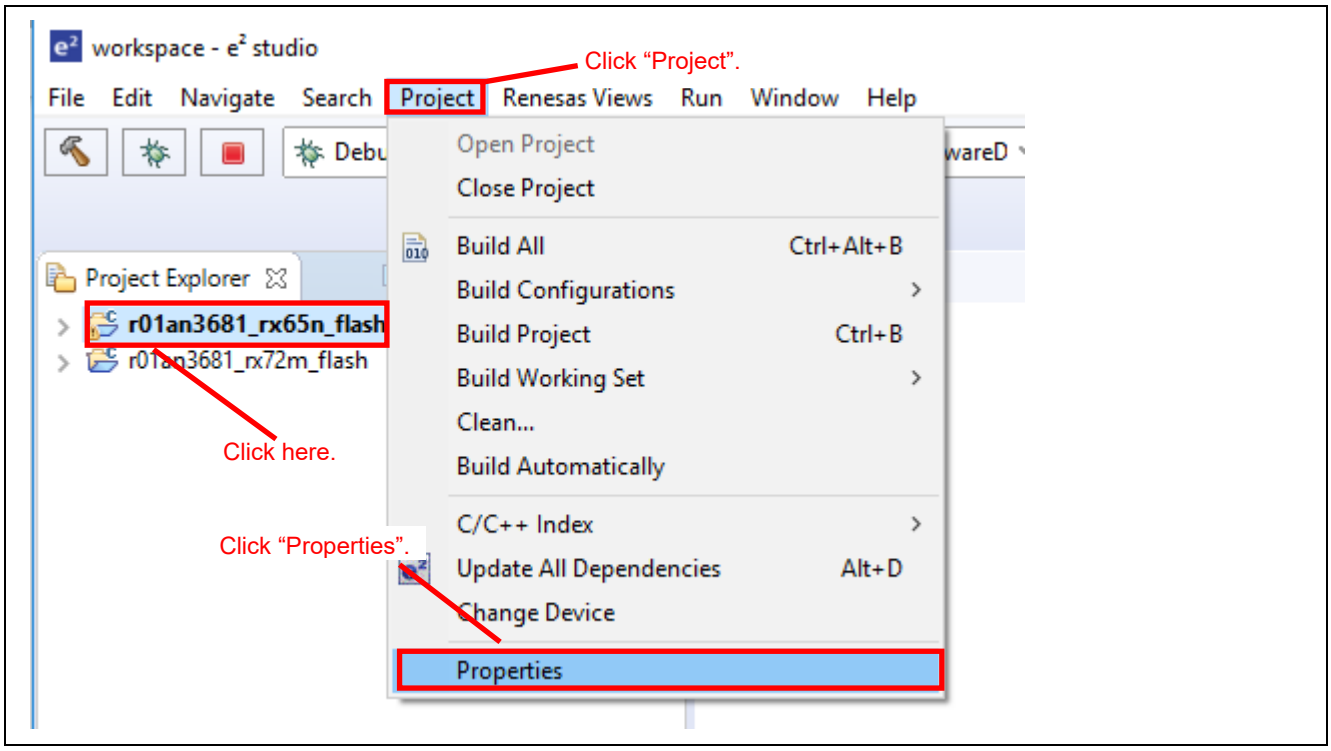


3.4.3 Changes to “C/C++ Build” Settings

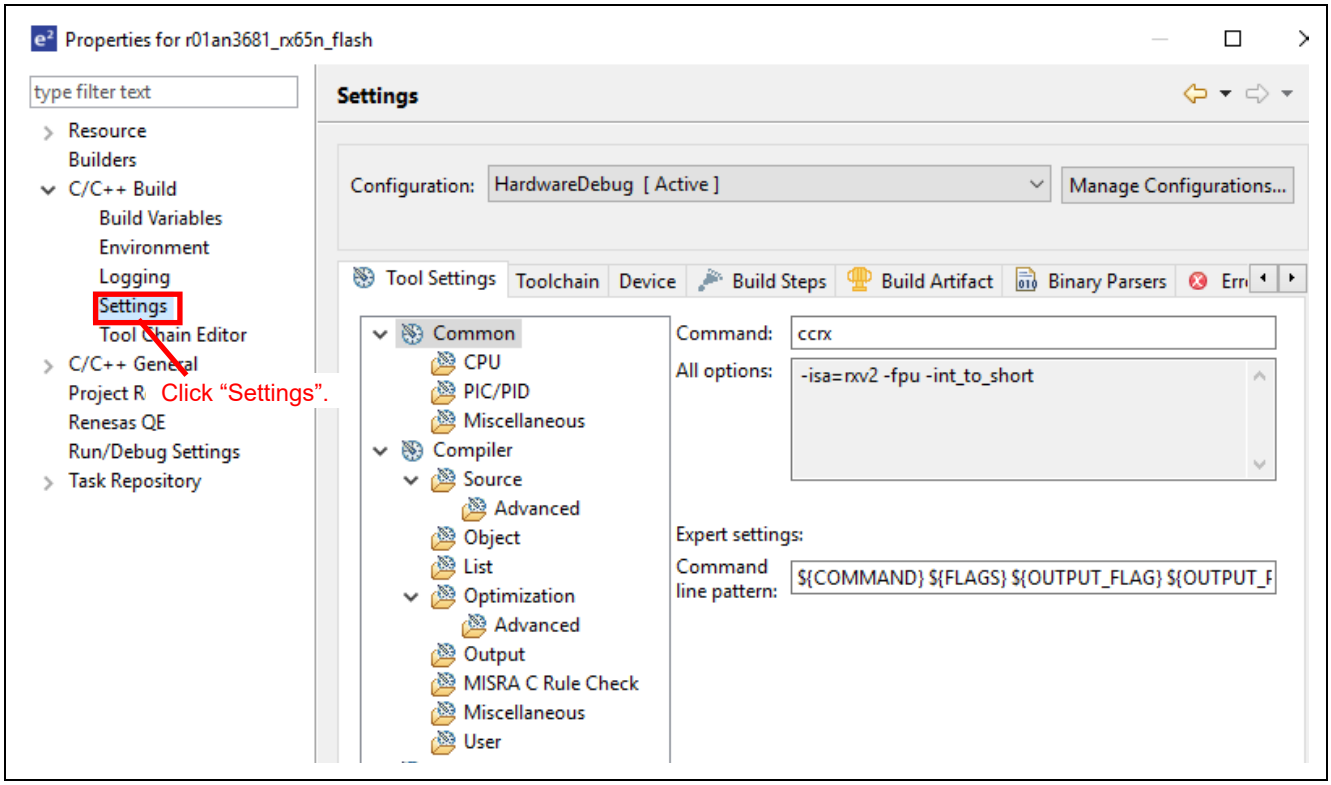
The default build-time settings in the project have been changed to values described in Table 3.1, Changed Build Settings of the Project.

You can confirm the settings changed with the following procedure:

1. In Project Explorer, click the target project.
2. Select “Project” >> “Properties”.

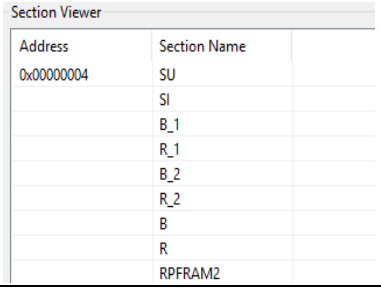
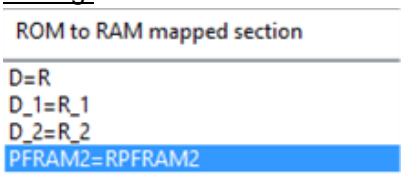
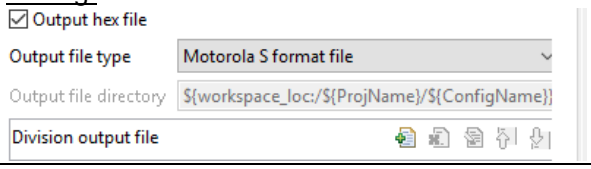


3. Select “C/C++ Build” and then “Settings”.



4. In the “Tool Settings” tab, see the settings have been changed as listed in Table 3.1, Changed Build Settings of the Project.

Table 3.1 Changed Build Settings of the Project

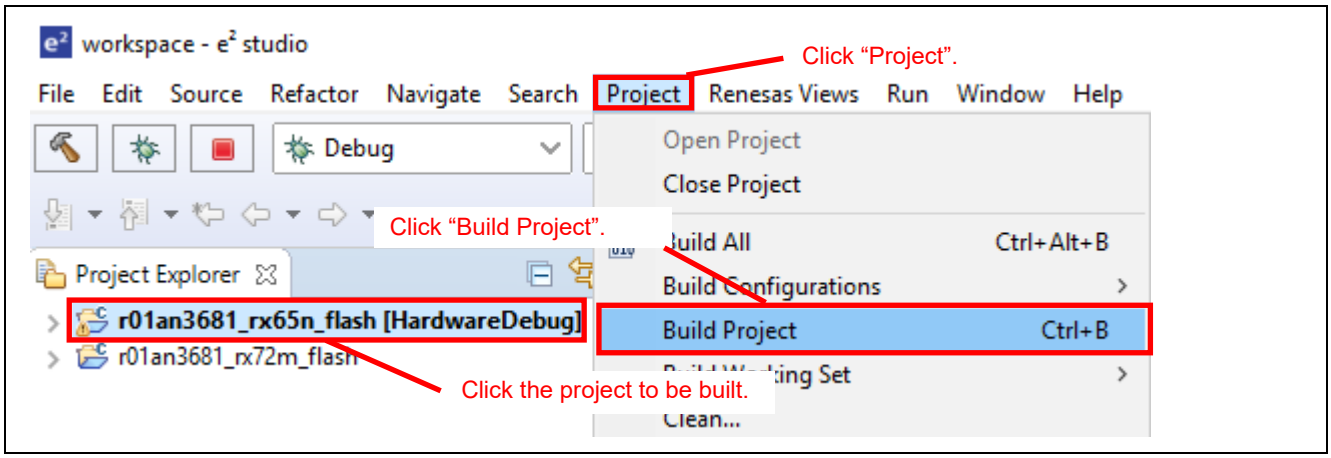
Item	Changed Item	Description
Compiler Source	src is added to the “Include file directories” section.	Add to include path to allow header files in src to be accessed from folders within src.
Linker Section	The RPFRAM2 section is added to the RAM area. Setting:  	Specify the RAM area to be used by the Flash API.
Linker Section Symbol file	“PFRAM2=RPFRAM2” is added to the “ROM to RAM mapped section” section. Setting:  	Add the ROM to RAM mapping since the code to switch the startup bank with the Flash FIT API needs to be executed on the RAM.
Converter Output	Check “Output hex file”. Setting:  	Output the mot file.

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.
2. Select “Project” >> “Build Project”.



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

```
    -nomessage
rlink   "r01an3681_rx65n_flash.abs" -subcommand="Converterr01an3681_rx65n_flash.tmp"

Renesas Optimizing Linker Completed
'Finished building target:'

'Build complete.'

14:03:04 Build Finished. 0 errors, 0 warnings. (took 21s.932ms)
```

4.2 Preparing Debugging

4.2.1 Preparing Devices

The evaluation board needs to be prepared before debugging.

Table 4.1 lists required devices and configurations.

Table 4.1 Devices and Configurations

No.	Device	Remarks
1	Development PC	PC used for development
2	Evaluation board (Renesas Starter Kit+ for RX65N-2MB) (Renesas Starter Kit+ for RX72M)	In this application note, power is supplied to the board from AC adapter.
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	See Figure 4.1 for an example of the environment.

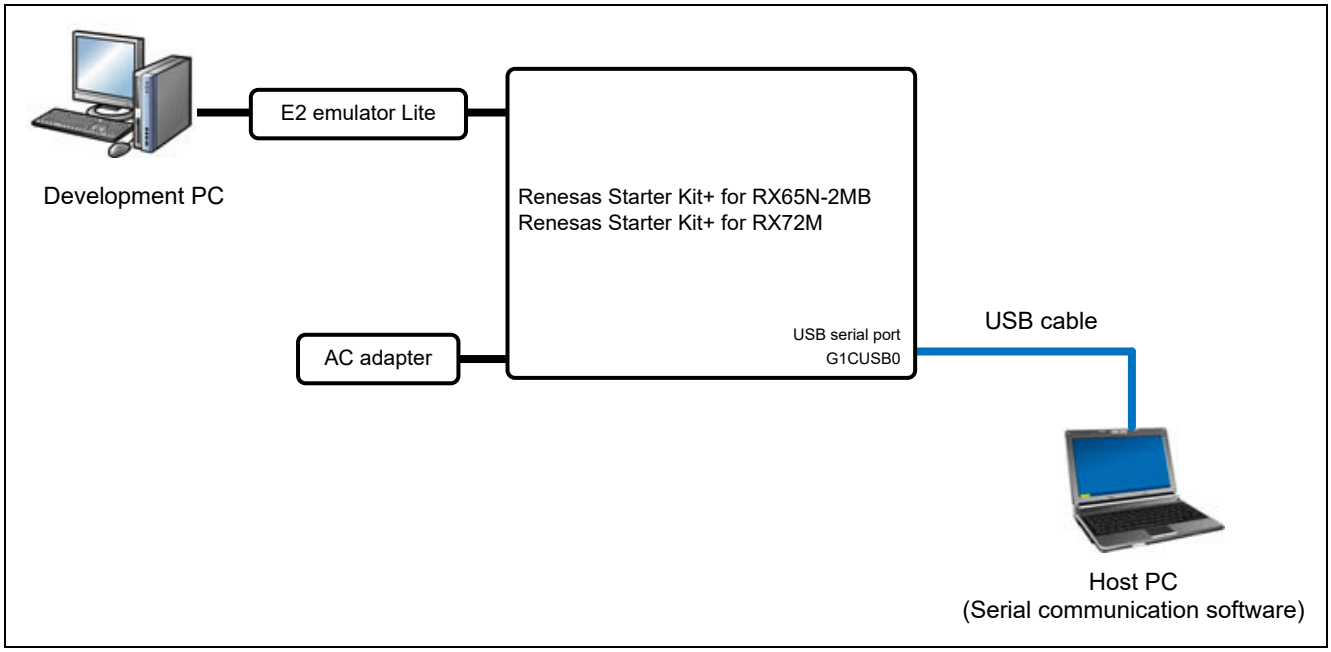


Figure 4.1 Debug Configuration

4.2.2 Host PC Settings

Table 4.2 lists the communication specification for the serial communication software. For serial communication software configuration, refer to the document for the serial communication software.

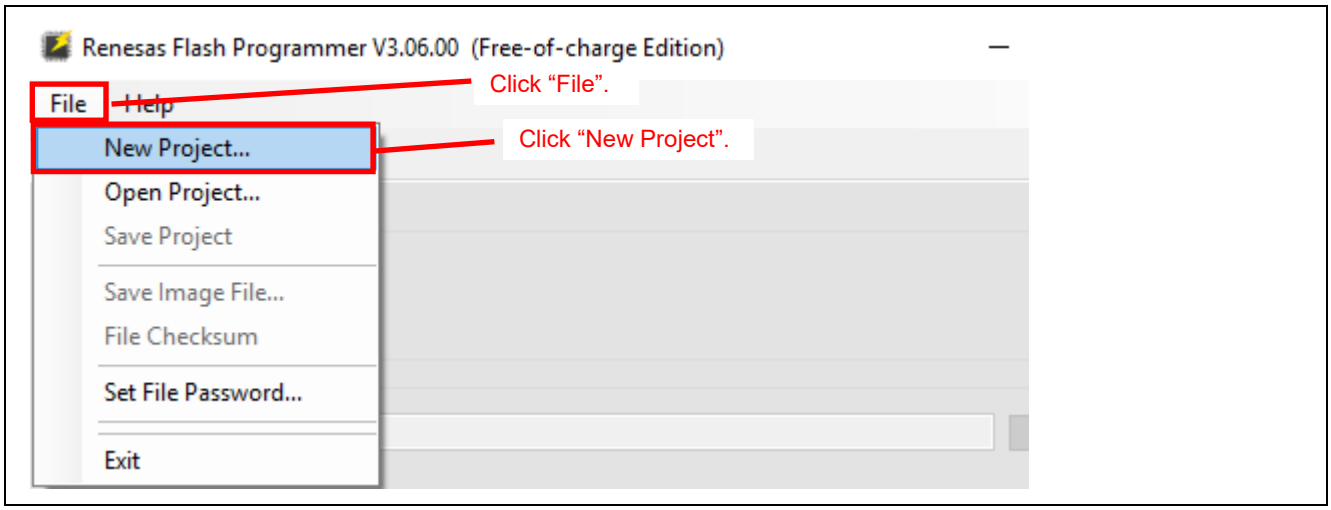
Table 4.2 Communication Specification

Item	Description
Communication method	Asynchronous communication
Communication bit rate	115,200 bps
Data length	8 bits
Parity	None
Stop bit	1 bit
Flow control	None

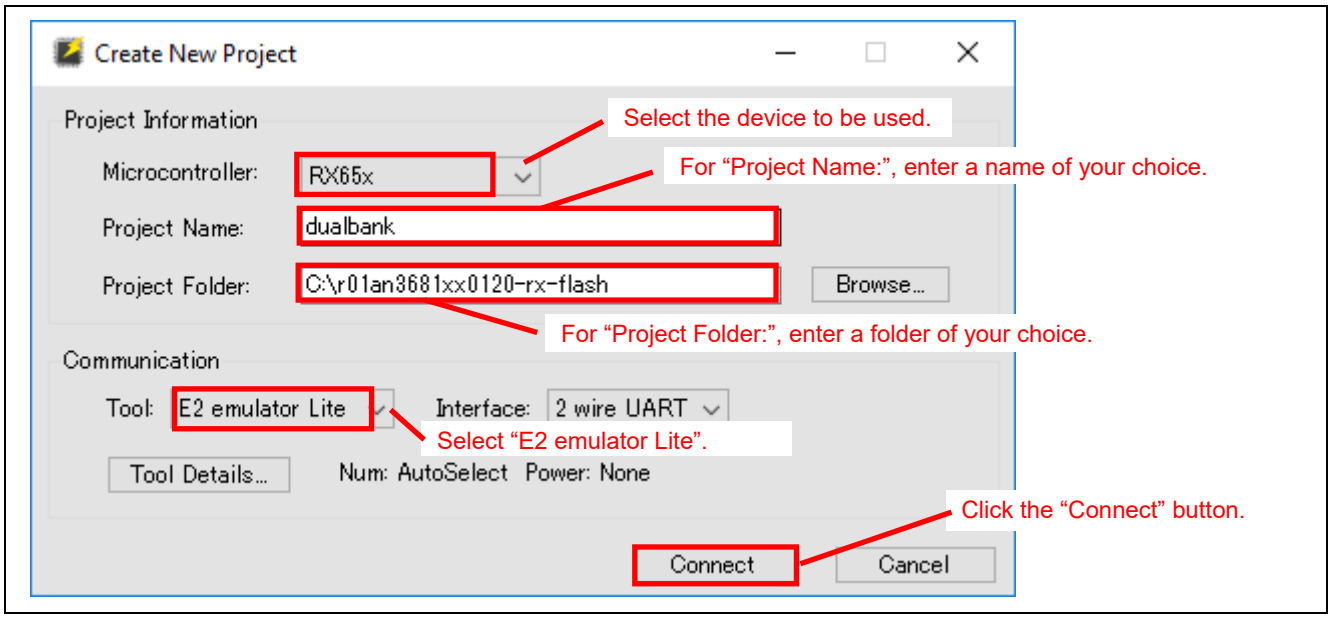
4.2.3 Presetting the MCU

To use the dual bank function, first the MCU has to be set to dual mode. Follow the procedure below for setting dual mode.

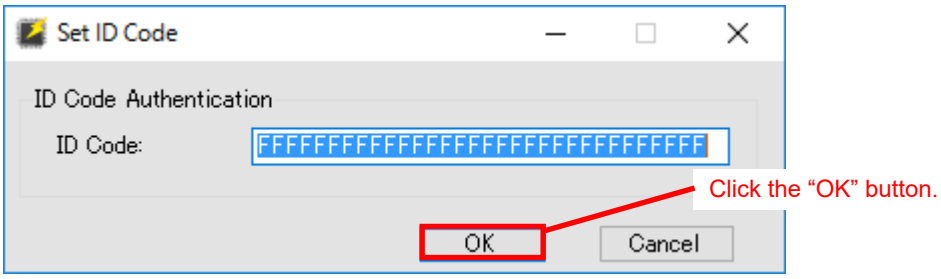
1. Start up the Renesas Flash Programmer.
2. Select “File” >> “New Project”.



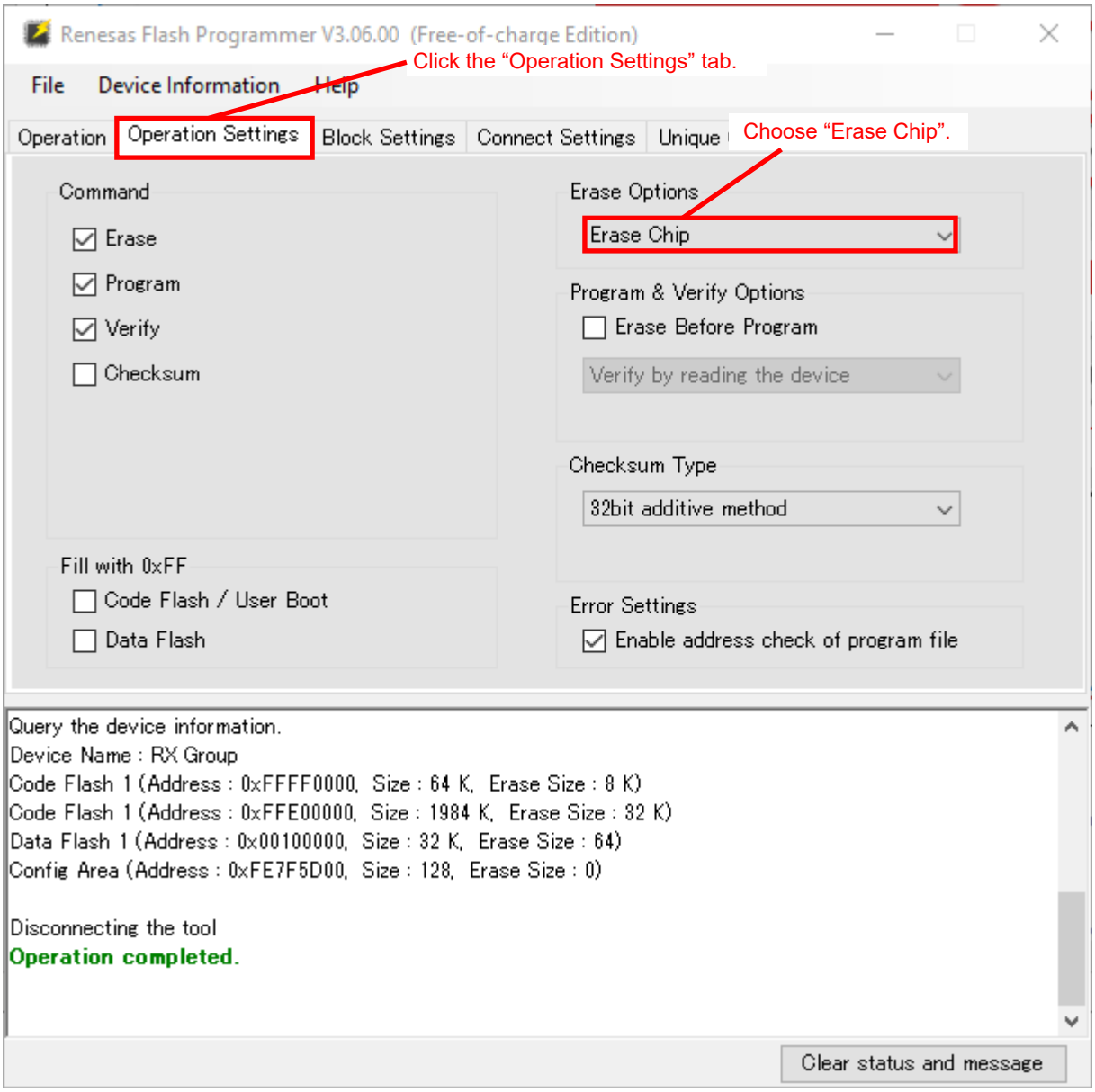
3. The “Create New Project” dialog opens.
4. For “Microcontroller:”, select the device to be used.
5. Enter a project name in the “Project Name” field.
6. Enter a folder in the “Project Folder” field.
7. For “Tool:”, select “E2 emulator Lite”.
8. Click the “Connect” button.



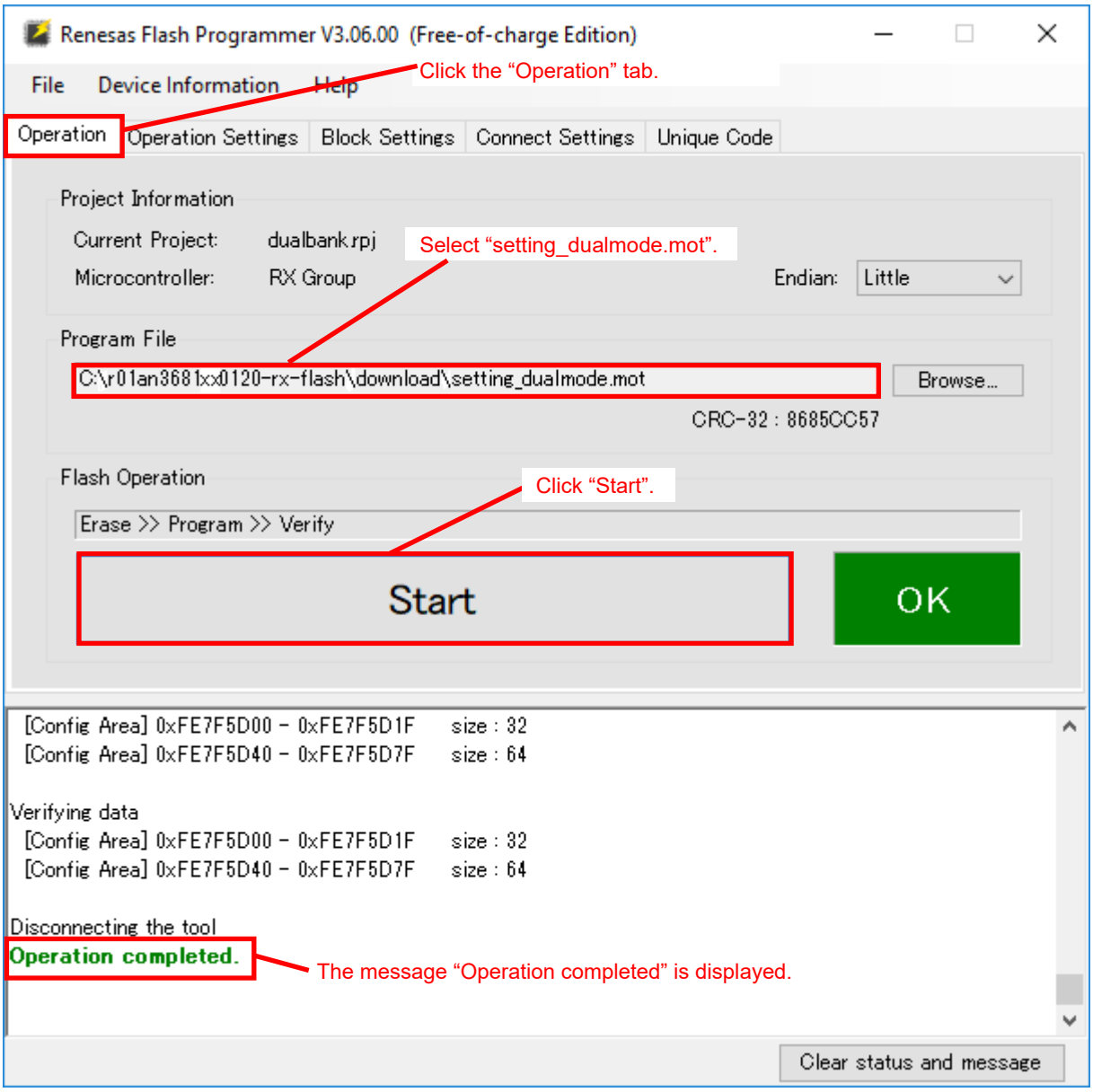
9. The “ID Code Setting” dialog appears.
10. Click the “OK” button.



11. Click the “Operation Settings” tab.
12. Choose “Erase Chip” for the “Erase Option” section.



13. Click the “Operation” tab.
14. Select the “setting_dualmode.mot” file in the “Program File” field. The file is located in the “download” folder under the unzipped folder of this application note.
15. Click the “Start” button.
16. The message “Operation completed.” is displayed. Now the MCU has been set to dual mode.



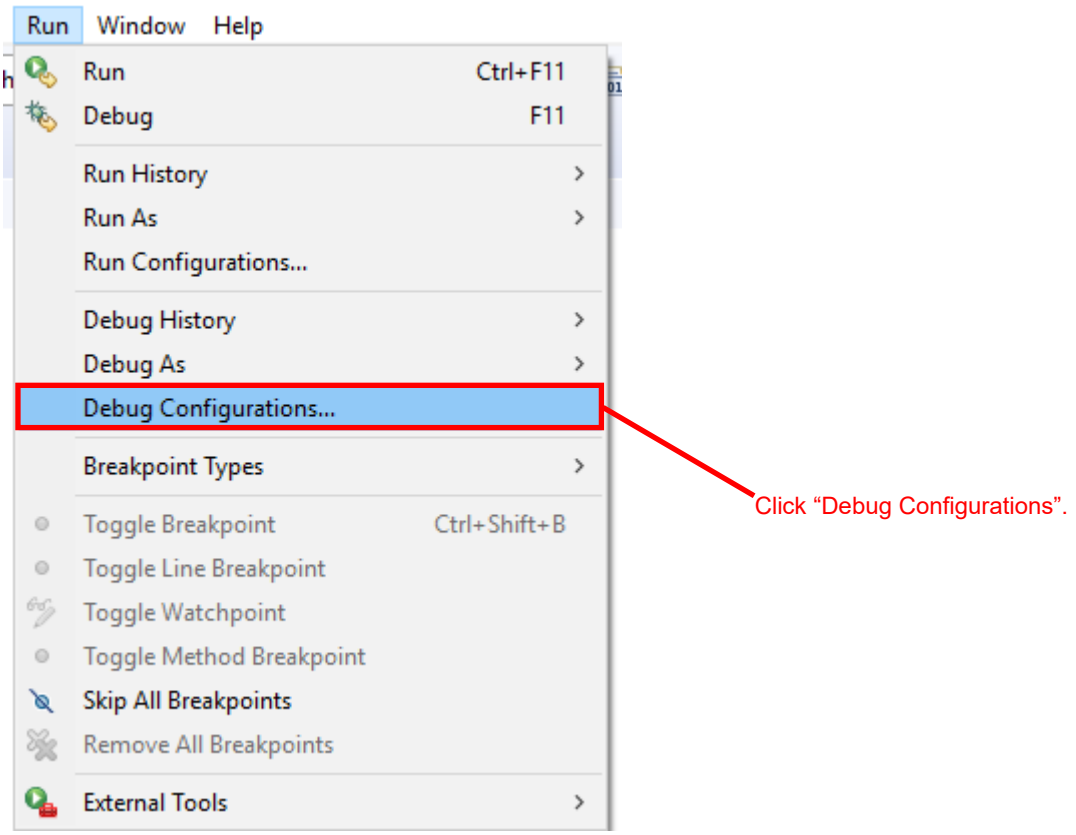
The Renesas Flash Programmer requires projects for dual mode and linear mode separately. The project for linear mode cannot be connected in dual mode. If another connection needs to be made using the Renesas Flash Programmer, create a project again.

4.3 Debug Configuration

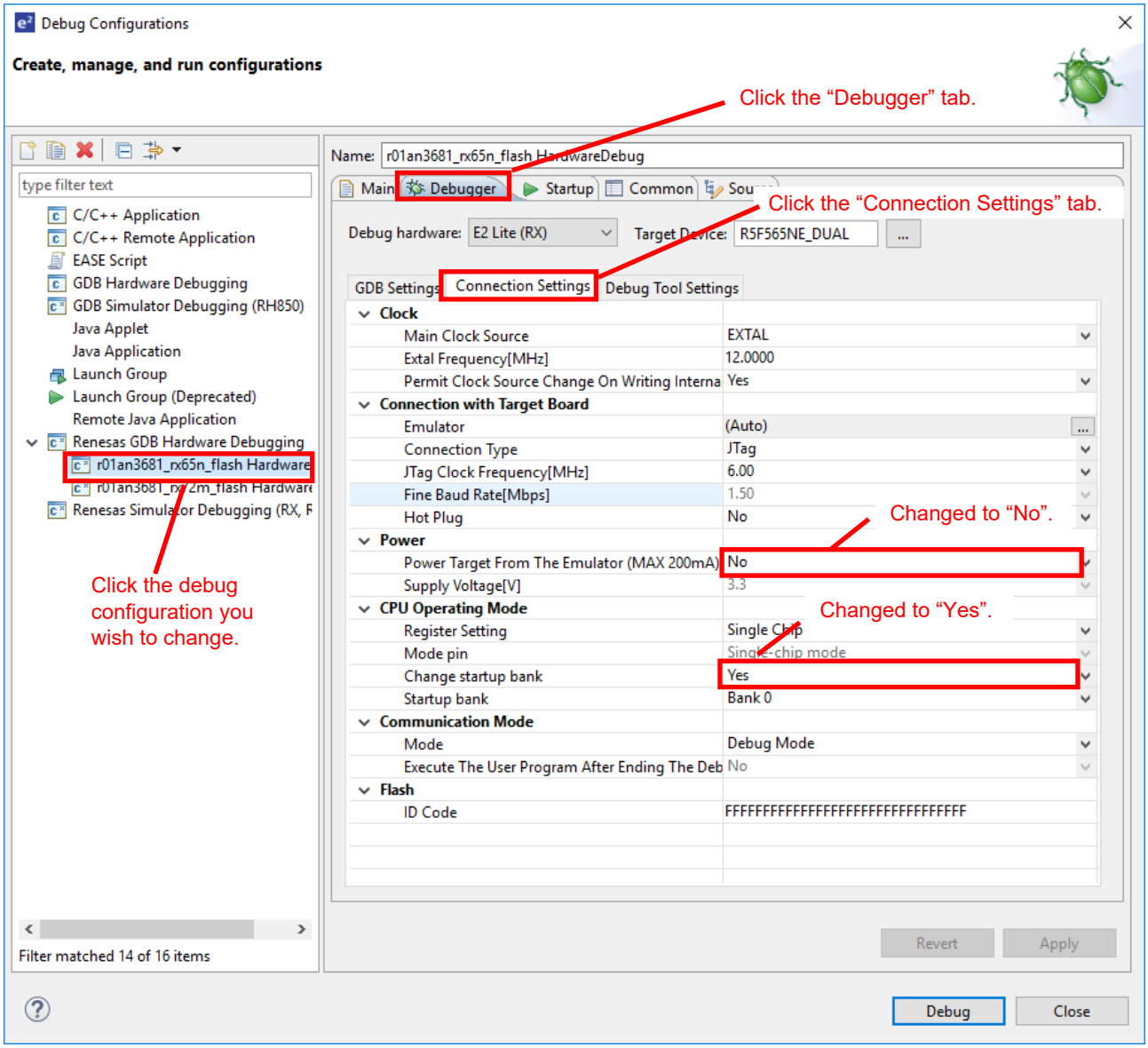
The debug configuration has been changed in the sample program. You can confirm the debug configuration changed in the project with the procedure below.

The procedure can be used to change the settings when setting up a new project. When performing debug execution, go to 4.4, Debugging.

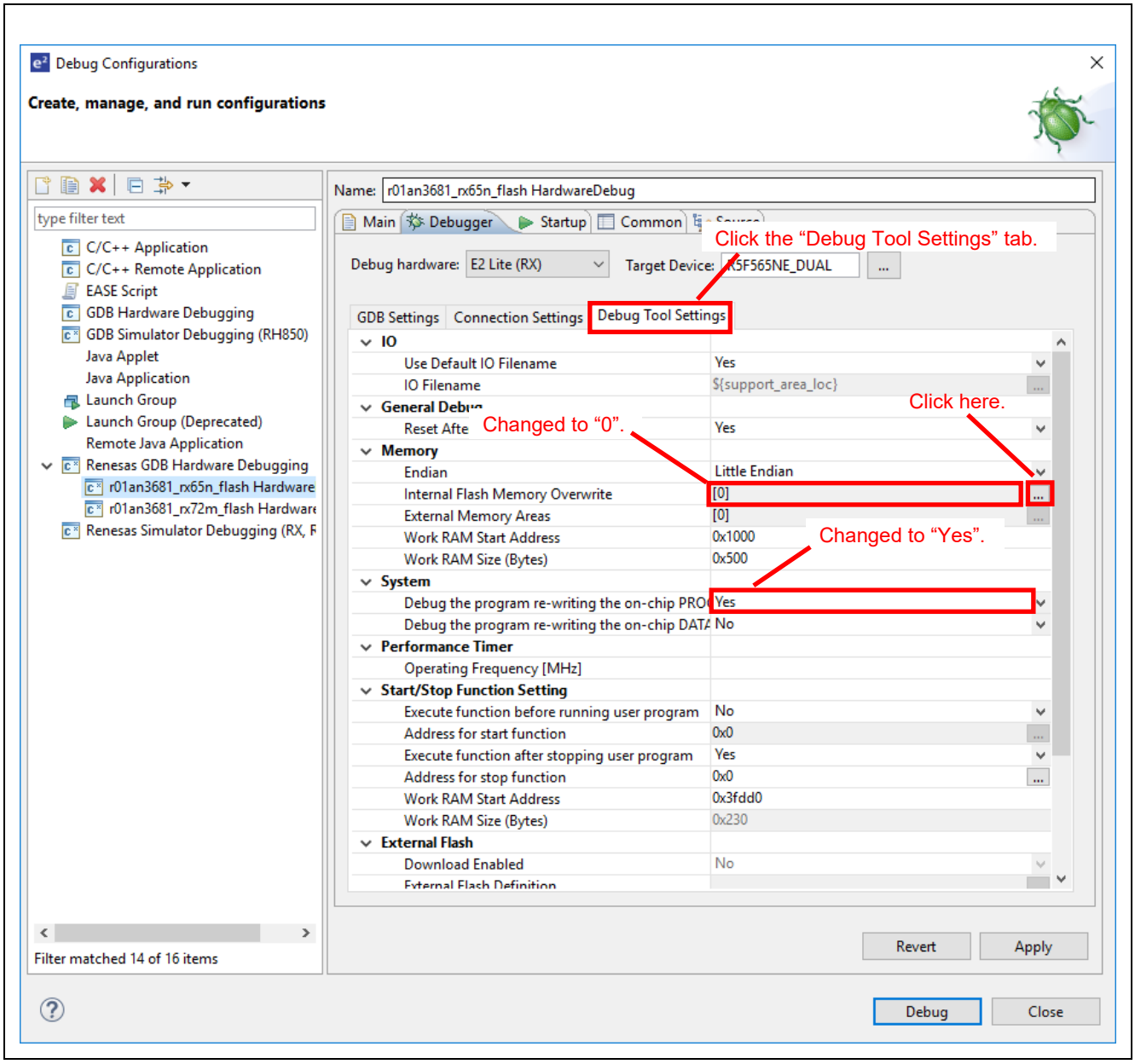
1. Select “Run” >> “Debug Configurations” in the e^2 studio.



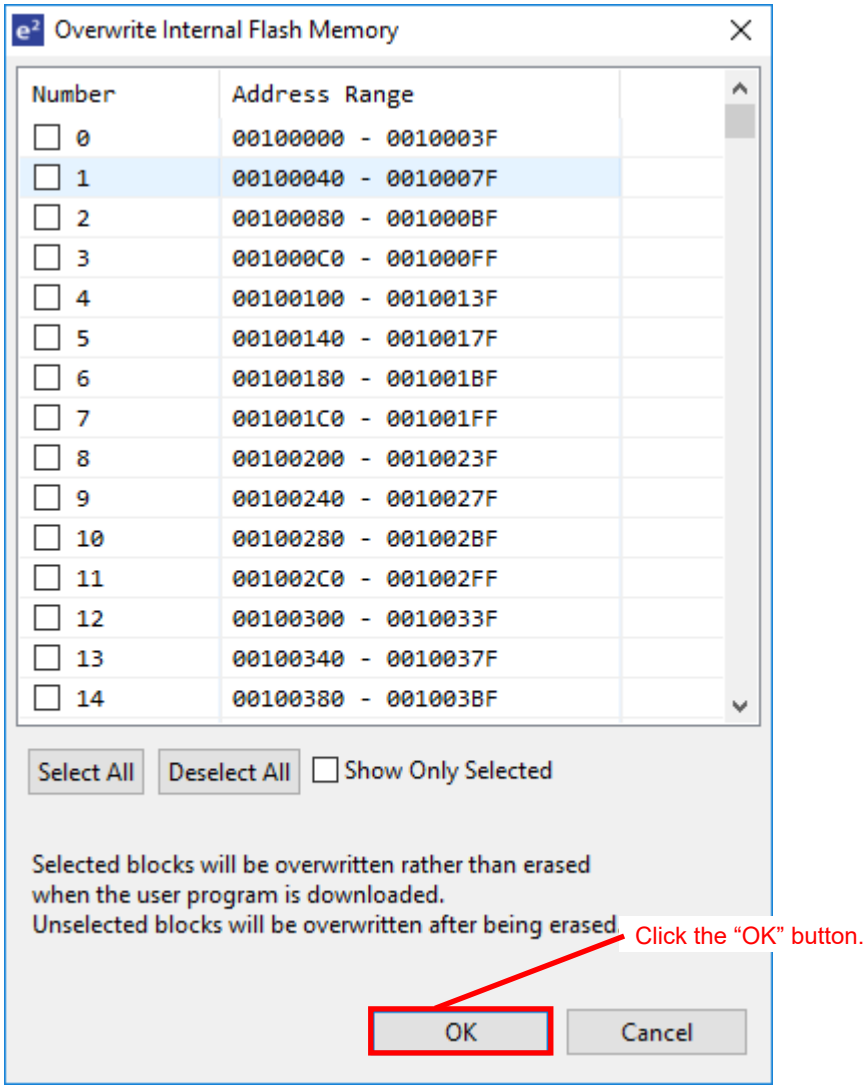
2. Select the debug configuration you want to confirm the changes under “Renesas GDB Hardware Debugging”.
3. Click the “Debugger” tab.
4. Click the “Connection Settings” tab.
5. The “Power Target From The Emulator” setting is changed to “No”.
6. The “Change startup bank” setting is changed to “Yes”.



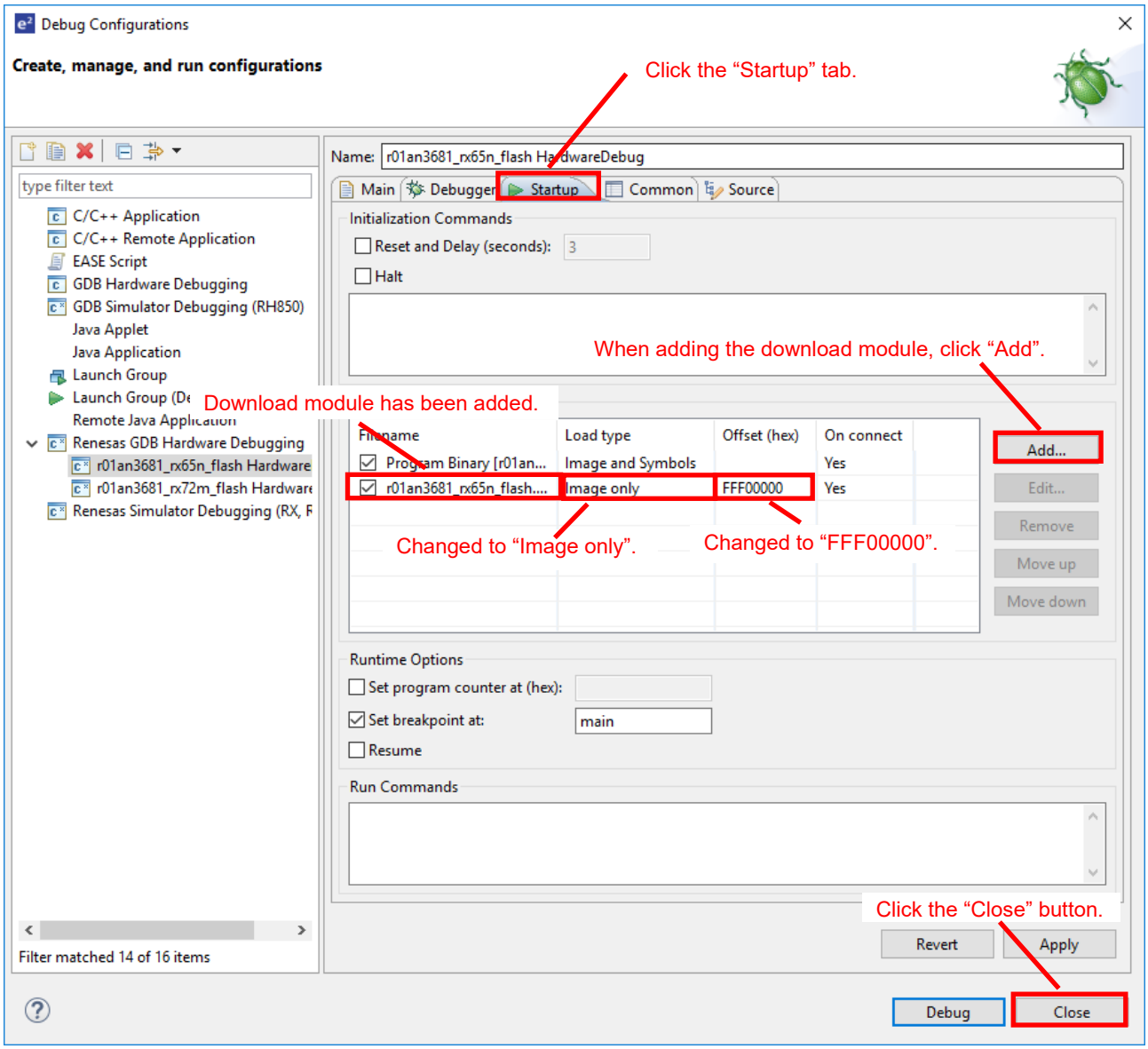
7. Click the “Debug Tool Settings” tab.
8. See the setting for “Debug the program to re-write the on-chip ROM” has been changed to “Yes”.
9. On the RX72M, “Performance Timer” is set to “240”.
10. See the setting for “Internal Flash Memory Overwrite” has been changed to “0”. To change this setting, click “…” button on the right.



11. See the overwrite setting has been specified to execute overwrite operation after erasing all blocks in the on-chip flash memory. Clicking the “Deselect All” button deselects all address ranges (removes all check marks).
12. Click the “OK” button.



13. Click the “Startup” tab to see the image to load into the other bank of the startup bank has been added.
14. To add the download module, click the “Add” button in the “Startup” tab. The “Add download module” dialog opens. To add the file in the project, click the “Search Project” button.

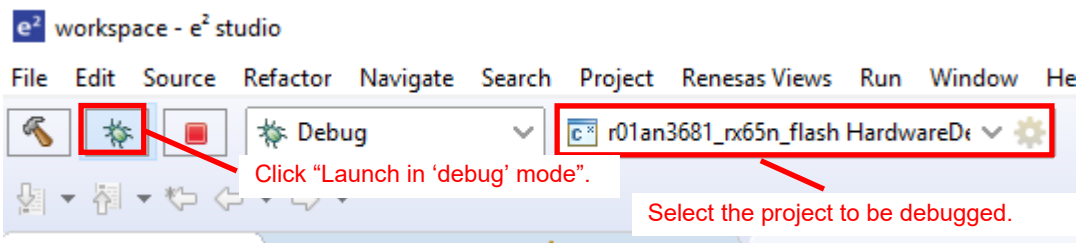


15. A file with the HardwareDebug extension x has been added.
16. See the “Load type” setting has been changed to “Image only”.
17. On the RX65N, the “Offset” setting is changed to “FFF00000”. On the RX72M, the “Offset” setting is changed to “FFE00000”.*[1]
18. Click the “Close” button.

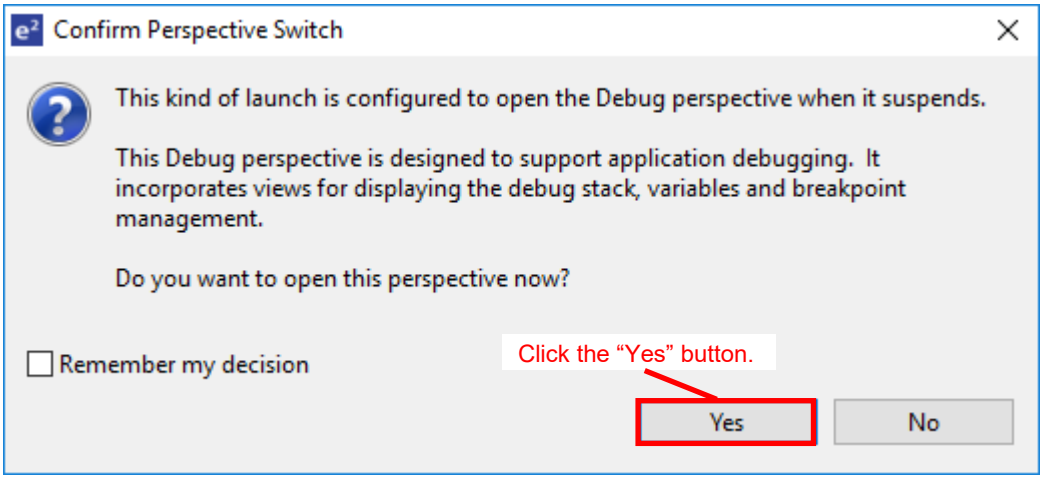
Note: 1. The value specified here is not an address value but an offset value. On the RX65N, the significance is “address value − 100000H”. It is not possible to enter a negative value, so the 2’s complement is entered as FFF00000H. Setting an offset of FFF00000H specifies an address value in the bank other than the current startup bank address value. On the RX72M the offset is “address value − 200000H”, so FFE00000H is entered.

4.4 Debugging

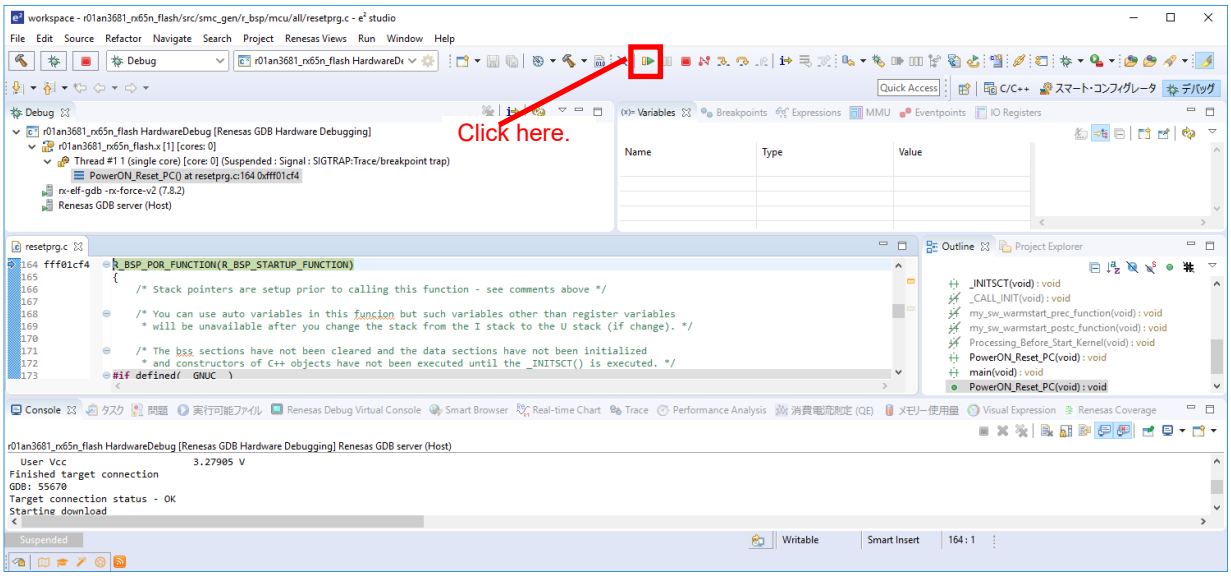
1. Select the project to be debugged.
2. Click the “Launch in ‘debug’ mode” icon.



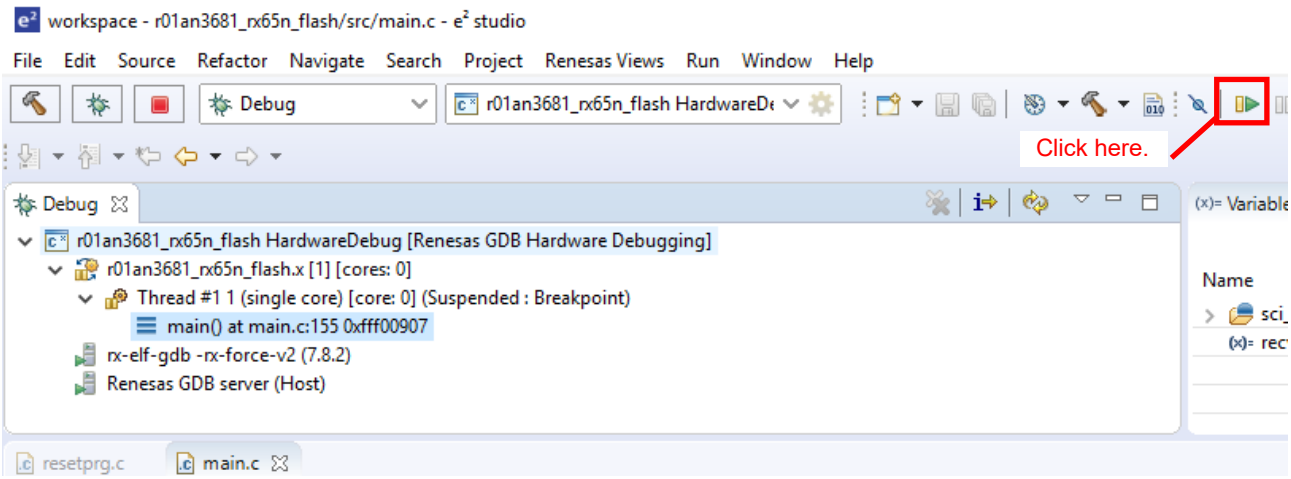
3. When the following message appears, click the “Yes” button.



4. When the load module has been downloaded, the “Debug” perspective opens.
5. Click the “Resume” icon on the toolbar. The program is executed and halted at the start of the main function where the break point is inserted.



6. Click the “Resume” icon on the toolbar again after the halt in step 5.



7. The following message is displayed in the serial communication software screen.

```
Dualbank firmware update menu ver1.20A
1...Update
2...Changing the Startup Bank and Reset
```

4.5 mot File for Update Confirmation

The sample program uses a mot file for update confirmation. When an update to the mot file is completed, the character string appearing in the menu display changes to “ver1.20B”.

```
Dualbank firmware update menu ver1.20B
1...Update
2...Changing the Startup Bank and Reset
```

The mot file is located in the “download” folder in the zip file containing this application note. There are a total of four versions of the mot file to cover the various device and endian setting combinations.

5. Overview of the Sample Program

5.1 Operation Overview

The sample program transfers the firmware (.mot file) through serial communication using the XMODEM/SUM protocol to program the firmware into the code flash memory. The firmware operates on the startup bank of the dual bank. The mot file in addresses FFF0 0000h to FFFF FFFFh is programmed in addresses FFE0 0000h to FFEF FFFFh in the other bank of the startup bank. After confirmed that the firmware has been programmed successfully, the firmware can be updated safely by changing the startup bank and performing a software reset. Programming is not performed if the specified address range is other than the range between FFF0 0000h and FFFF FFFFh.

Note that the addresses FFF0 0000h, FFE0 0000h, and FFEF FFFFh apply to the RX65N with 2 MB of ROM. When the device is the RX72M with 4 MB of ROM, the above addresses are replaced by FFE0 0000h, FFC0 0000h, and FFDF FFFFh, respectively. This also applies to Figure 5.1 to Figure 5.6.

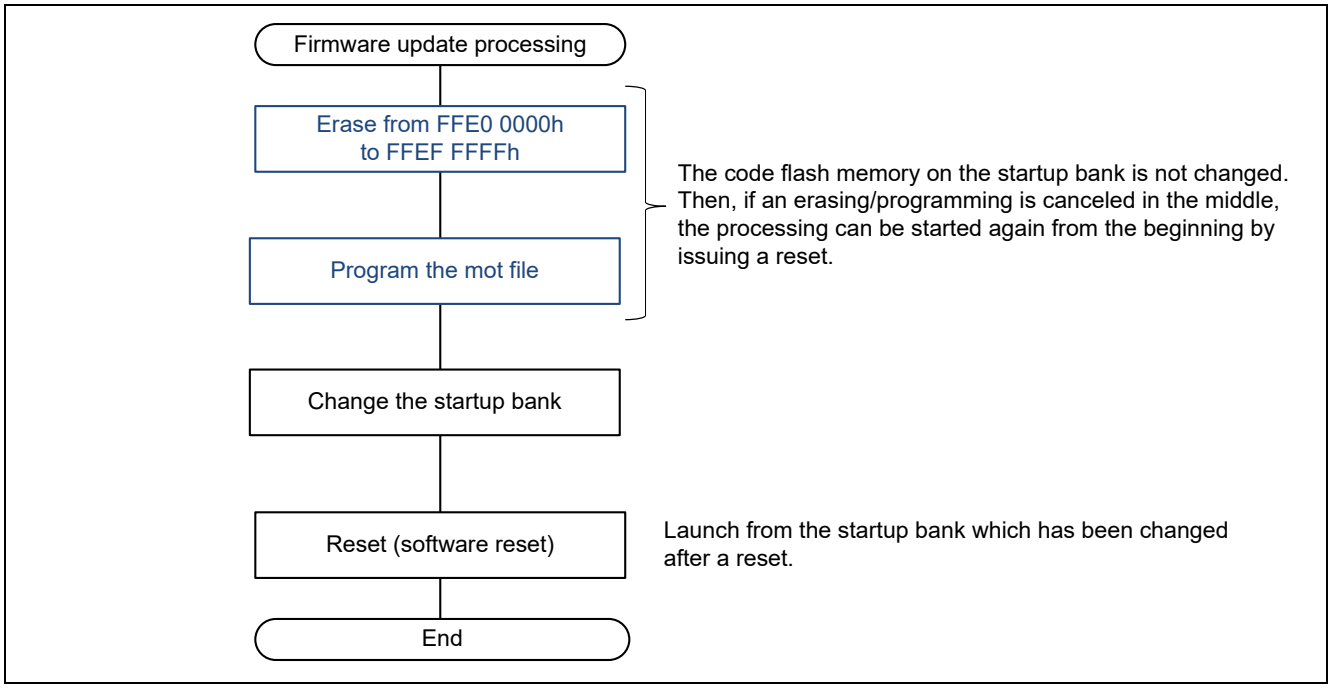


Figure 5.1 Firmware Update

5.1.1 Updating the Firmware

This section describes updating the firmware using the sample program.

1. Start the sample program. The sample program displays the menu in the serial communication software on the host PC.

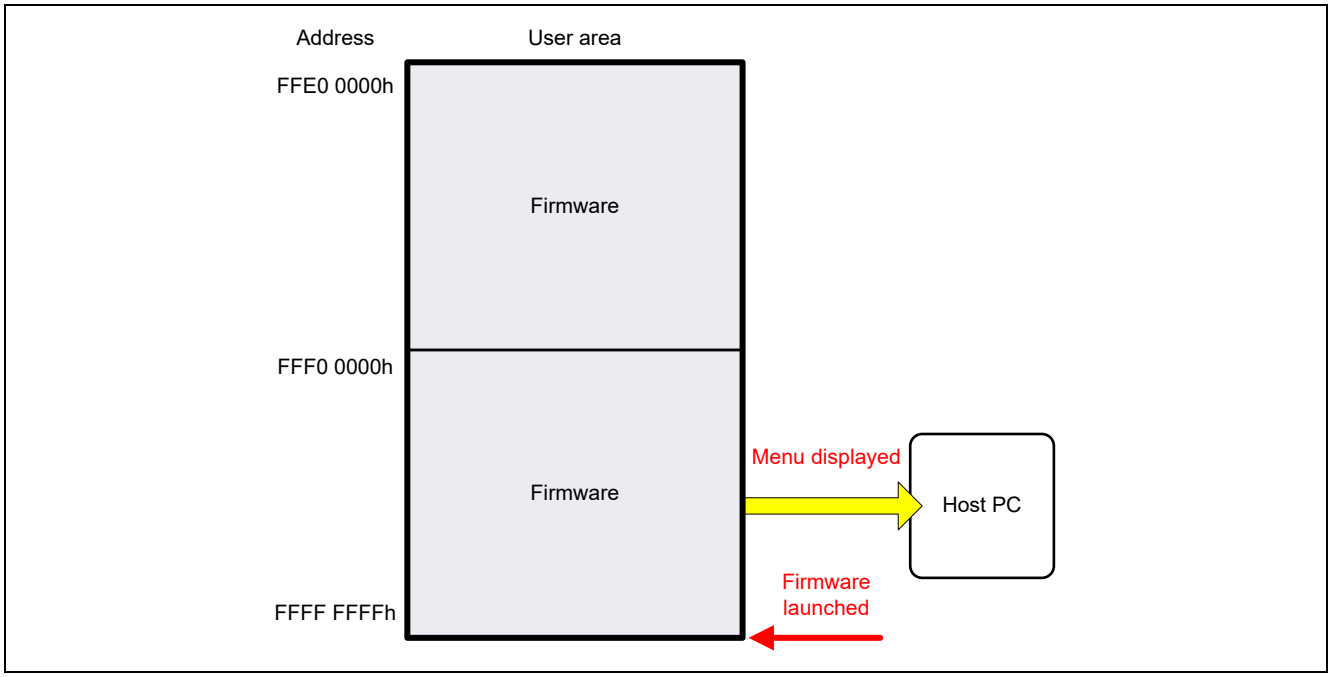


Figure 5.2 Launching the Sample Program

2. When the Update command is executed, the sample program erases the code flash memory on the other bank of the startup bank.

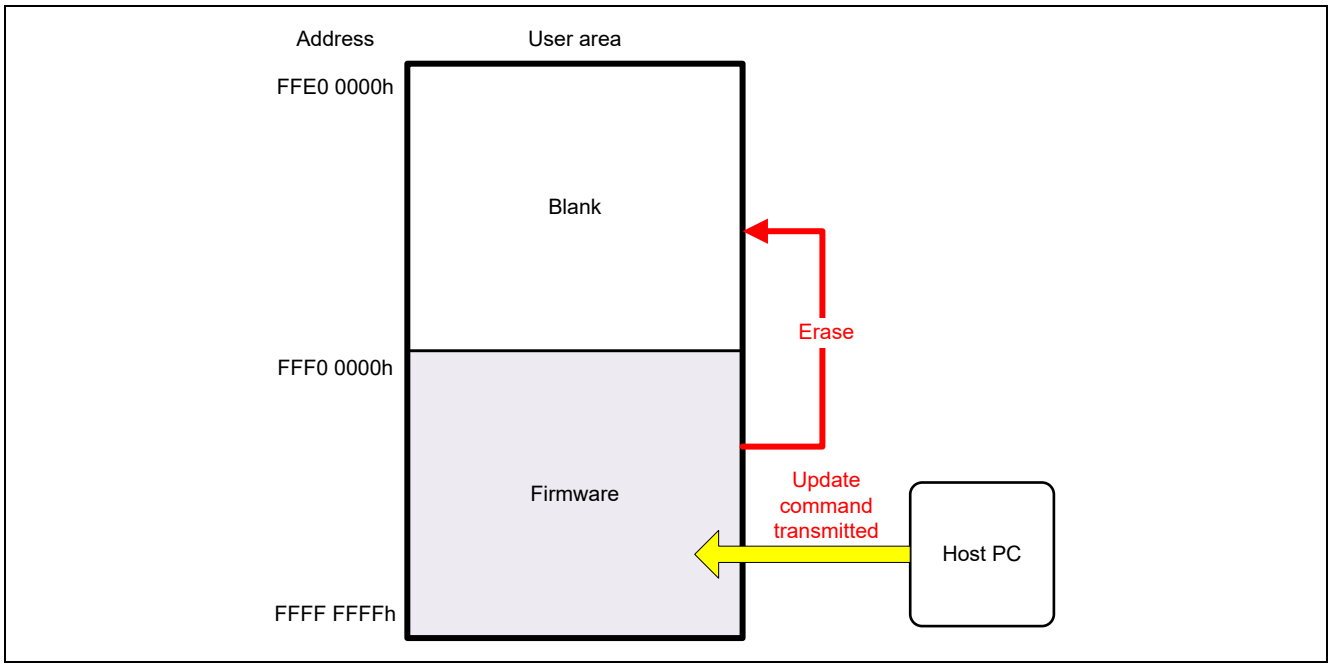


Figure 5.3 Erasing the Code Flash Memory

3. The firmware is transmitted using the serial communication software. The sample program analyzes the received data and program it into the code flash memory.

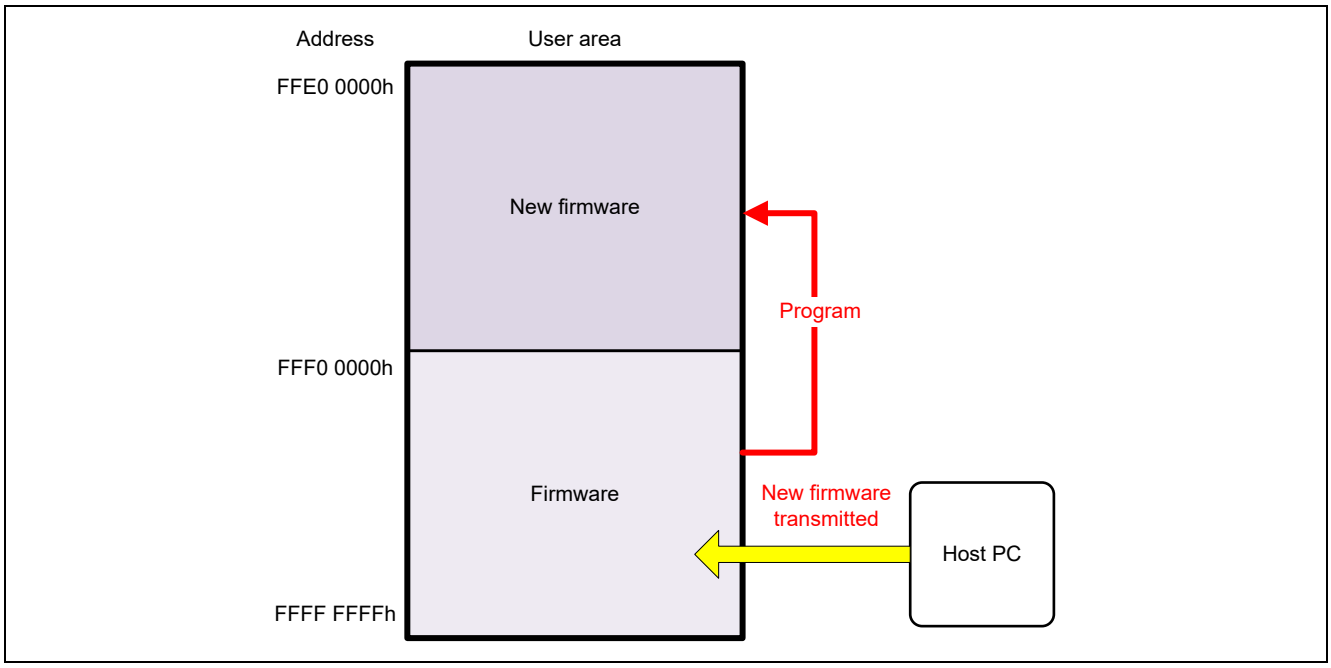


Figure 5.4 Programming New Firmware

4. When the sample program has completed to program the new firmware into the code flash memory, it displays the menu in the serial communication software on the host PC.

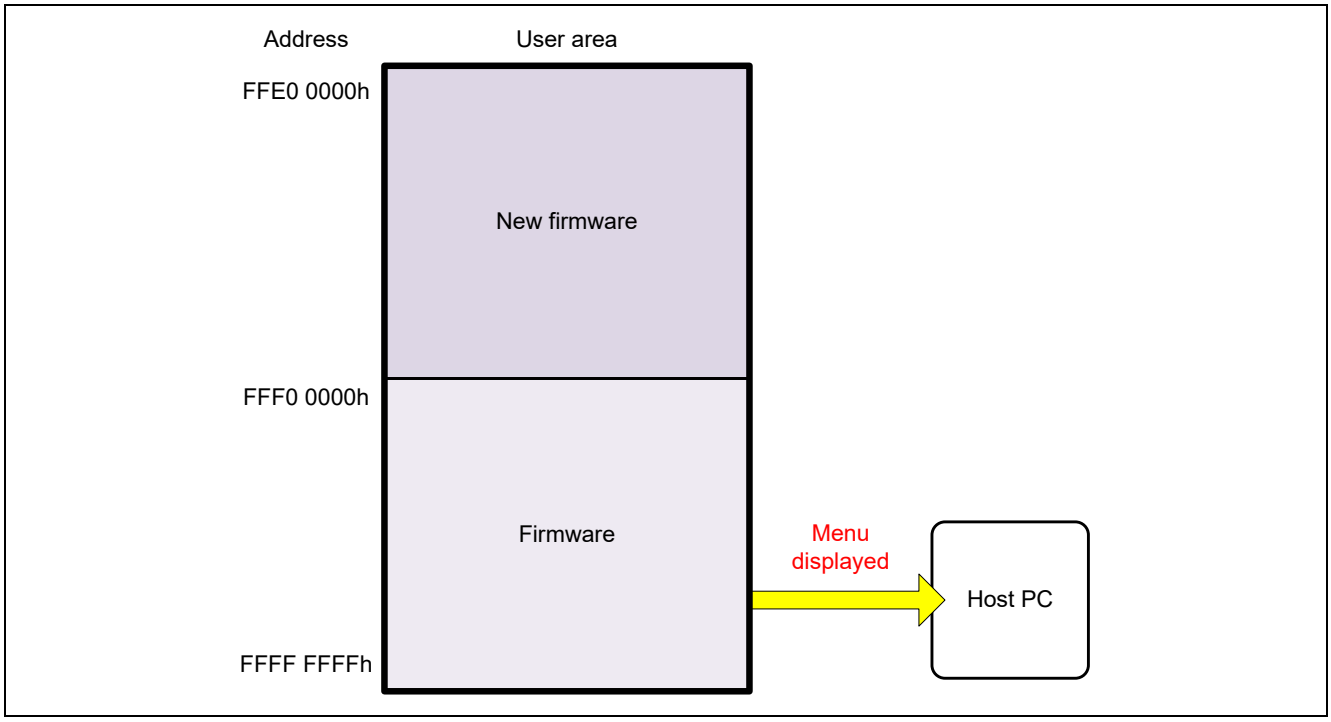


Figure 5.5 Completion of Programming New Firmware

5. When the Changing the Startup Bank and Reset command is executed, the sample program executes a software reset after reprogramming the bank select register of the option-setting memory. The startup bank is switched after a reset and the new firmware is launched.

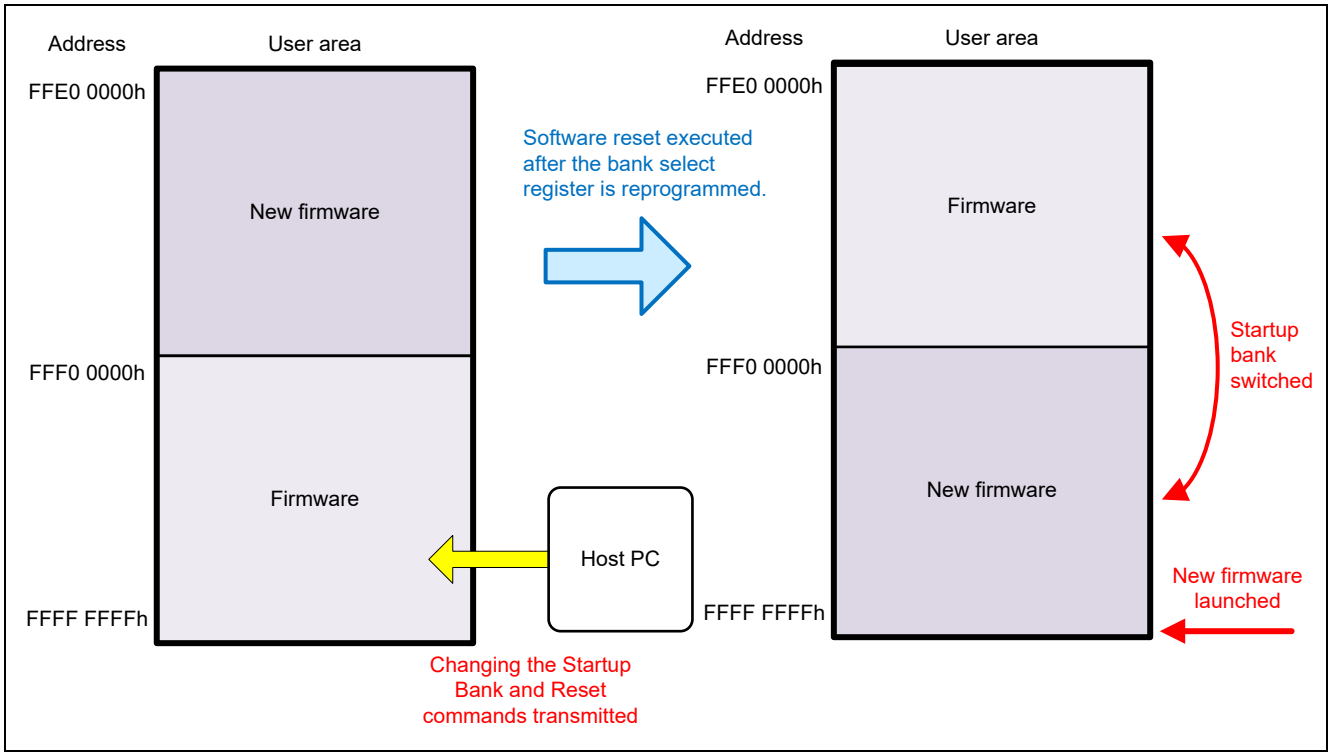


Figure 5.6 Software Reset and Launching New Firmware

5.2 Process Flowchart and Screen Output

The sample program outputs messages to the serial communication software on the host PC using the serial communication and branches to an appropriate processing according to the command input from the serial communication software.

5.2.1 Main Processing

The main processing initializes the SCI FIT module and the Flash FIT module, and uses the SCI to display the menu in the serial communication software on the host PC. Then the main processing waits for a key input from the serial communication software and branches to an appropriate processing according to the key input.

(1) Process Flowchart

Figure 5.7 shows the flowchart of main processing.

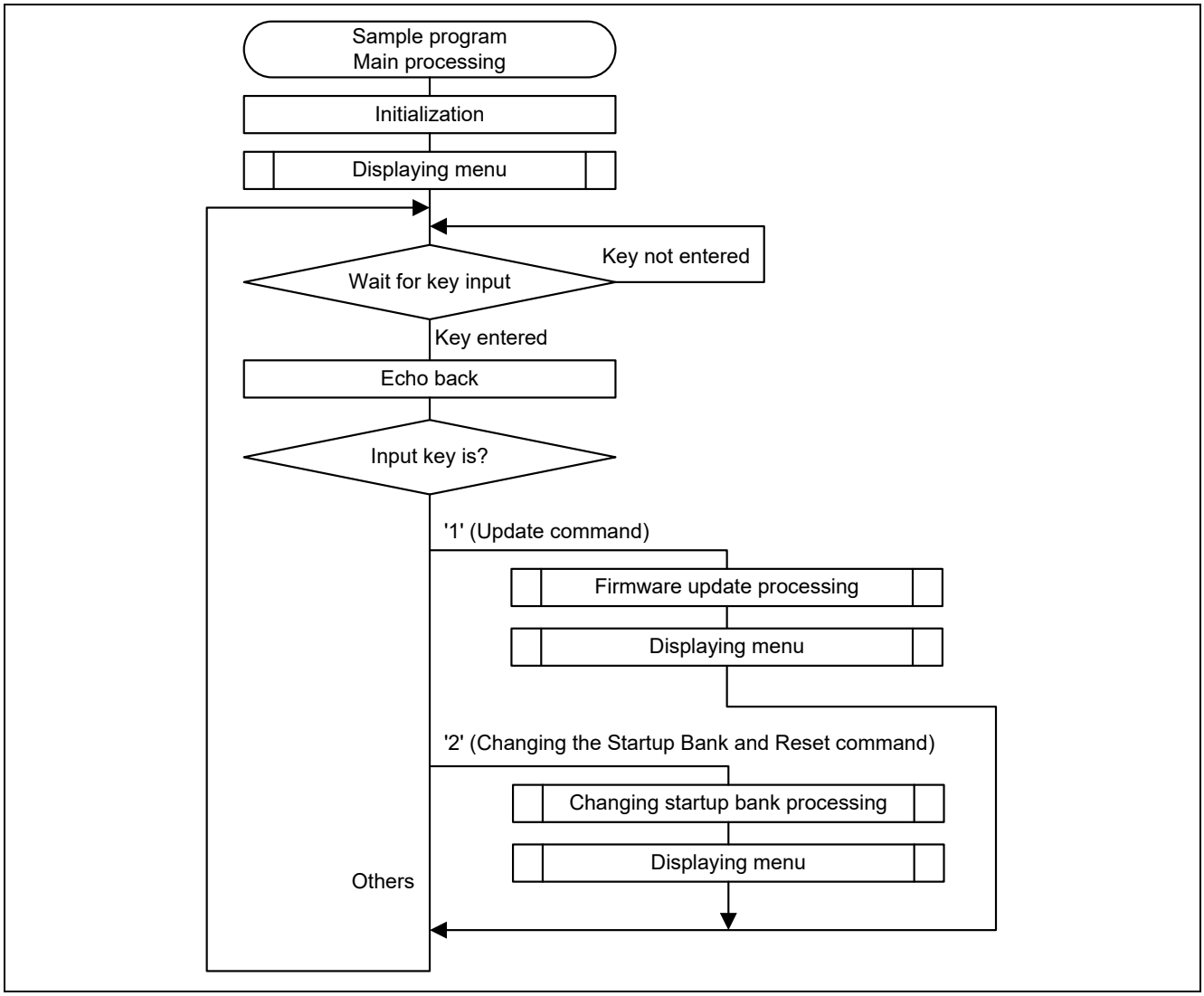


Figure 5.7 Flowchart of Main Processing

(2) Screen Output in the Serial Communication Software

When the sample program is launched, the menu is displayed in the serial communication software.

```
Dualbank firmware update menu ver1.20A
1...Update
2...Changing the Startup Bank and Reset
```

When “1” is entered in the serial communication software, the firmware update processing is executed.
When “2” is entered, the changing startup bank processing is executed.

5.2.2 Firmware Update Processing

When “1” is entered, the main processing branches to the firmware update processing. The firmware is received through the serial communication using the XMODEM/SUM protocol and then programmed into the code flash memory.

(1) Process Flowchart

Figure 5.8 shows the flowchart of firmware update processing.

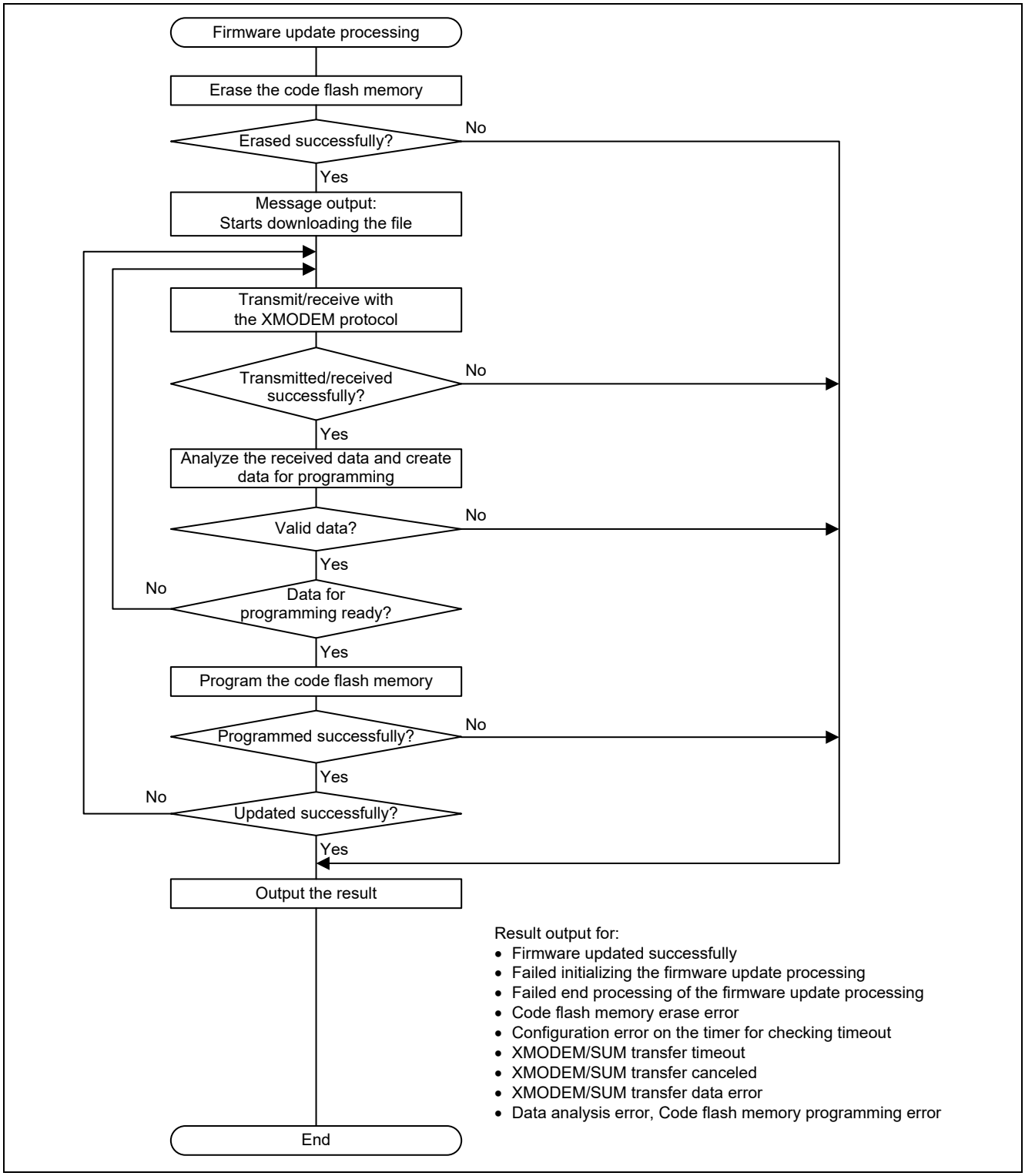


Figure 5.8 Flowchart of Firmware Update Processing

(2) Screen Output in the Serial Communication Software

1. Starting file download
   When addresses FFE0 0000h to FFEF FFFFh in the code flash memory have been successfully erased, the message to start downloading is output. Then a wait processing for receiving the XMODEM/SUM transferred data is started.

Ready

   Transmit the firmware with the XMODEM/SUM protocol from the serial communication software.

2. Completing the firmware update
   When the firmware update has been completed, the following message is output and the firmware update processing is completed.

Finish

3. 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 the firmware update processing
finalize update error:	Failed end processing of the firmware update processing
Erase error:	Code flash memory erase error
CMT error:	Configuration error on the timer for checking timeout
Timeout:	XMODEM/SUM transfer timeout
Received CAN:	XMODEM/SUM transfer canceled
Data error:	XMODEM/SUM transfer data error
Block processing error:	Data analysis error, Code flash memory programming error

5.2.3 Changing the Startup Bank Processing

When “2” is entered, the main processing branches to the changing startup bank processing. A new firmware is launched by executing a software reset after switching the startup bank.

(1) Process Flowchart

Figure 5.9 shows the flowchart of changing startup bank processing.

The option-setting memory is not an area that can be overwritten using background operation (BGO). This means that it is not possible to read data from the code flash memory while the option-setting memory is being overwritten. The interrupt vector table is located in the code flash memory, so interrupts are disabled while switching startup banks. If you do not wish to disable interrupts during this period, allocate the interrupt vector table to the RAM.

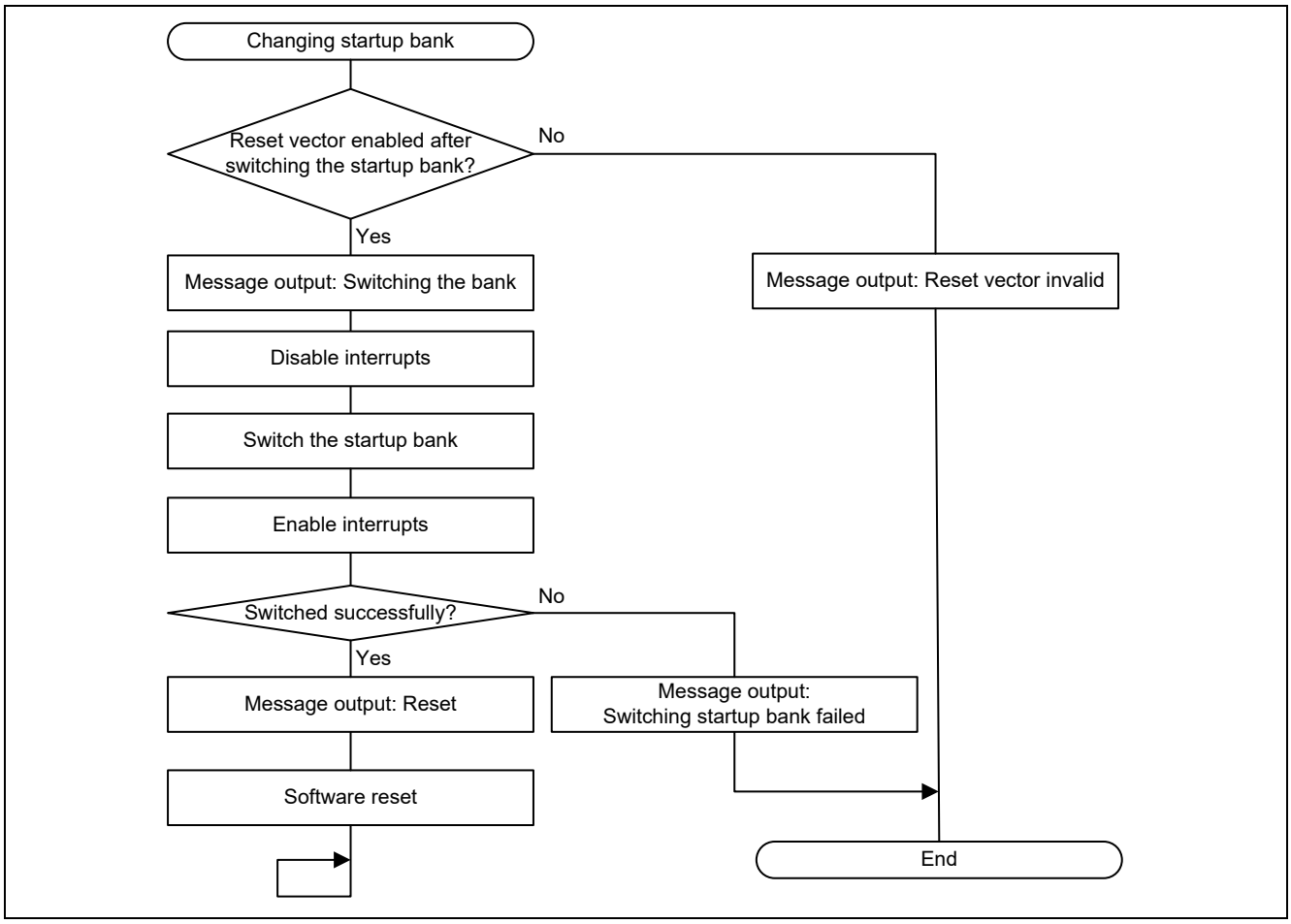


Figure 5.9 Flowchart of Changing Startup Bank Processing

(2) Screen Output in the Serial Communication Software

1. Changing the startup bank
   The following message is output and the startup bank is switched.

```
Changing the Startup Bank
```

2. Software reset
   When the startup bank is switched successfully, the following message is output and a software reset is executed.

```
Reset
```

When an update is successfully performed with the update file for your device included in the ZIP file, the following menu is displayed.
The version number changes from 1.20A to 1.20B.

```
Dualbank firmware update menu ver1.20B
1...Update
2...Changing the Startup Bank and Reset
```

3. Error output
   If an error occurs during changing the startup bank, any of the following message is output according to the error.

```
Reset vector is invalid:  Reset vector invalid
Flash error:              Switching startup bank failed
```

5.3 Sample Program

5.3.1 File Composition

Table 5.1 lists the files used in the sample program and Table 5.2 lists the standard header files used in the sample program. Files generated by the FIT module and files generated by the integrated development environment are not included in these tables.

Table 5.1 Files Used in the Sample 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 5.2 Standard Header Files Used in the Sample 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.

5.3.2 Constants

Table 5.3 to Table 5.6 list constants used in the sample program.

Table 5.3 Constants Used in the Sample Program (main.c)

Constant	Setting Value	Description
SCI_CH	((uint8_t)SCI_CH8)	SCI channel used on RX65N
	((uint8_t)SCI_CH6)	SCI channel used on RX72M
TX_MAX_SIZE	SCI_CFG_CH8_TX_BUFSIZ	Transmit buffer size when using SCI8
	SCI_CFG_CH6_TX_BUFSIZ	Transmit buffer size when using SCI6
RECV_BYTE_SIZE	((uint16_t)1)	1 byte size for receiving
SEND_BYTE_SIZE	((uint16_t)1)	1 byte size for transmitting
CHARACTER_RETURN	((uint8_t)'r')	Linefeed code
COMMAND_UPDATE	((uint8_t)'1')	Character code for the Update command
COMMAND_CHANGE_BANK	((uint8_t)'2')	Character code for the Changing the Startup Bank and Reset command
CMT_FREQUENCY_Hz	(2u)	Compare match timer interrupt period (2 Hz)
STRING_MAX_SIZE	((uint16_t) TX_MAX_SIZE)	Maximum size for an output string
STRING_SIZE	((uint16_t)((sizeof((s))) – 1u))	Macro which indicates the string array size in uint16_t

Table 5.4 Constants Used in the Sample Program (r_xmodem.c)

Constant	Setting Value	Description
XM_SOH	((uint8_t)0x01)	XMODEM control code (SOH)
XM_EOT	((uint8_t)0x04)	XMODEM control code (EOT)
XM_ACK	((uint8_t)0x06)	XMODEM control code (ACK)
XM_NAK	((uint8_t)0x15)	XMODEM control code (NAK)
XM_CAN	((uint8_t)0x18)	XMODEM control code (CAN)
XM_HEADER_SIZE	((uint8_t)(1+1+1))	Header size of the XMODEM data block (the number of bytes)
XM_DATA_SIZE	((uint8_t)128)	Data size of the XMODEM data block (the number of bytes)
XM_SUM_SIZE	((uint8_t)1)	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	((uint8_t)10)	The number of retries upon XMODEM data transfer timeout
UINT8T_0	((uint8_t)0)	0 in uint8_t
UINT8T_1	((uint8_t)1)	1 in uint8_t

Table 5.5 Constants Used in the Sample Program (r_fw_up_rx_private.h)

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_BLANK_VALUE	(0xFFFFFFFFu)	Read value when the code flash memory is blank.

Table 5.6 Constants Used in the Sample Program (r_fw_up_buf.h)

Constant	Setting Value	Description
MOT_S_CHECK_SUM_FIELD	(0x02)	The number of characters for the checksum field in the Motorola S-record format
ADDRESS_LENGTH_S1	(0x04)	The number of characters for the address field in the Motorola S-record format (S1 type)
ADDRESS_LENGTH_S2	(0x06)	The number of characters for the address field in the Motorola S-record format (S2 type)
ADDRESS_LENGTH_S3	(0x08)	The number of characters for the address field in the Motorola S-record format (S3 type)
BUF_LOCK	(1)	The specified buffer of Motorola S-record format is locked.
BUF_UNLOCK	(0)	The specified buffer of Motorola S-record format is open.

5.3.3 Type Definitions

This section describes type definitions used in the sample program.

Type definitions in r_xmodem.c

```
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 *precv_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;
```

Type definitions in r_xmodem_if.h

```
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);
```

Type definitions in r_fw_up_rx_if.h

```
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_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;
```

Type definitions in r_fw_up_buf.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 *paddress;
    uint8_t *pdata;
    uint8_t type;
    uint8_t act;
    struct MotSBufS *pnext;
} 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 *pnext;
    struct WriteDataS *pprev;
} fw_up_write_data_t;
```

5.3.4 Variables

Table 5.7 to Table 5.10 list static variables and Table 5.11 lists const variables.

Table 5.7 static Variables (main.c)

Type	Variable	Description	Used by Functions
static sci_hdl	sci_handle	SCI module control handle	main send_byte_xm recv_byte_xm send_string_sci
static volatile bool	sci_send_end_flag	SCI transmit complete determination flag	sci_callback send_string_sci
static uint32_t	cmt_ch	Channel number of compare match timer	update_dualbank
static volatile int32_t	timeout_count	Timeout determination counter	cmt_callback recv_byte_xm
static volatile bool	timeout_flag	Timeout determination flag	cmt_callback recv_byte_xm
static volatile bool	start_timer_flag	Timeout determination enable flag	cmt_callback recv_byte_xm

Table 5.8 static Variable (r_xmodem.c)

Type	Variable	Description	Used by Functions
static uint8_t	recv_buf[XM_BLOCK_SIZE]	XMODEM receive data buffer	exec_xmodem

Table 5.9 static Variable (r_fw_up_rx.c)

Type	Variable	Description	Used by Functions
static bool	is_opened	Firmware update initialization complete flag	fw_up_open fw_up_close write_firmware fw_up_put_data fw_up_get_data

Table 5.10 static Variables (r_fw_up_buf.c)

Type	Variable	Description	Used by Functions
static fw_up_mot_s_buf_t	*papp_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	*papp_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	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	*papp_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	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	mot_s_data_state	Analysis state of the Motorola S-record data	fw_up_buf_init fw_up_put_mot_s
static uint32_t	write_current_address	Current address to program in the code flash memory	fw_up_buf_init fw_up_get_binary
static bool	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 5.11 const Variables (main.c)

Type	Variable	Description	Used by Functions
static const uint8_t	string_menu0[]	“RX65N-2MB Dualbank firmware update menu ver1.00A\r\n”	show_menu_dualbank
static const uint8_t	string_menu1[]	“1...Update\r\n”	show_menu_dualbank
static const uint8_t	string_menu2[]	“2...Changing the Startup Bank and Reset\r\n”	show_menu_dualbank
static const uint8_t	string_change_bank[]	“Changing the Startup Bank\r\n”	change_startup_bank
static const uint8_t	string_reset[]	“Reset\r\n”	change_startup_bank
static const uint8_t	string_crlf[]	“\r\n”	main
static const uint8_t	string_start_xmodem[]	“Ready\r\n”	update_dualbank
static const uint8_t	string_finish_xmodem[]	“Finish\r\n”	update_dualbank
static const uint8_t	string_timeout[]	“Timeout\r\n”	update_dualbank
static const uint8_t	string_cmt_err[]	“CMT error\r\n”	update_dualbank
static const uint8_t	string_erase_err[]	“Erase error\r\n”	update_dualbank
static const uint8_t	string_flash_err[]	“Flash error\r\n”	change_startup_bank
static const uint8_t	string_block_err[]	“Block processing error\r\n”	update_dualbank
static const uint8_t	string_data_err[]	“Data error\r\n”	update_dualbank
static const uint8_t	string_recv_can[]	“Received CAN\r\n”	update_dualbank
static const uint8_t	string_initialize_update_err[]	“initialize update error\r\n”	update_dualbank
static const uint8_t	string_finalize_update_err[]	“finalize update error\r\n”	update_dualbank
static const uint8_t	string_reset_vector_invalid[]	“Reset vector is invalid\r\n”	change_startup_bank

5.3.5 Functions

Table 5.12 lists the functions used in the sample program, Table 5.13 lists the FIT module functions used in the sample program, and Table 5.14 lists the e^2 studio smart configurator generated function used in the sample program.

Table 5.12 Functions Used in the Sample Program

Function	Description	Defined File
main	Main processing	main.c
sci_callback	Callback function for the SCI FIT module to check completion of an SCI transmission	main.c
cmt_callback	Callback function for the CMT FIT module to check timeout with the CMT	main.c
send_string_sci	Transmitting strings	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
show_menu_dualbank	Displaying menu	main.c
update_dualbank	Firmware update processing	main.c
change_startup_bank	Changing startup bank processing	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
erase_another_bank	Erasing code flash memory	r_fw_up_rx.c
analyze_and_write_data	Analyzing receive data and programming code flash memory	r_fw_up_rx.c
bank_toggle	Switching startup bank	r_fw_up_rx.c
fw_up_soft_reset	Executing software reset	r_fw_up_rx.c
fw_up_check_reset_vector	Checking reset vector	r_fw_up_rx.c
write_firmware	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
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 5.13 FIT Module Functions Used in the Sample Program

Function	FIT Module	Application	Used by Functions
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_another_bank
R_FLASH_Write	Flash FIT module	Programming the code flash memory	write_firmware
R_FLASH_Control	Flash FIT module	Switching the startup bank	bank_toggle
R_BSP_RegisterProtect Disable	BSP module	Disabling the write protection for the SWRR register	fw_up_soft_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
R_CMT_CreatePeriodic	CMT FIT module	Generating the timer for checking timeout	update_dualbank
R_CMT_Stop	CMT FIT module	Stopping the timer for checking timeout	update_dualbank

Table 5.14 e^2 studio Smart Configurator Generated Function Used in the Sample Program

Function	FIT Module	Application	Used by Functions
R_SCI_PinSet_SCI8	SCI FIT module	SCI pin setting when using SCI8	main
R_SCI_PinSet_SCI6	SCI FIT module	SCI pin setting when using SCI6	main

6. Appendices

6.1 Troubleshooting

6.1.1 Operation Terminated Abnormally when Programming a mot File with the Renesas Flash Programmer

An operation may be abnormally terminated with the error: Error (E3000107): This device does not match the connection parameters. In this case, create a new project from “File” >> “Create New Project”.

This error occurs when the connection information in the project does not match the device. The Renesas Flash Programmer distinguishes between linear mode and dual mode in the connection information. A project for linear mode cannot be connected to the MCU operating in dual mode. Also a project for dual mode cannot be connected to the MCU operating in linear mode.

6.1.2 File Transfer Canceled During XMODEM Transfer

The following two are possibly the causes of this issue.

1. A file other than mot file is transferred
   In this case, terminate XMODEM processing in the serial communication software. Alternatively, wait until the XMODEM processing in the serial communication software becomes timeout. Then, perform the Update processing again. For terminating XMODEM processing, refer to the document for the serial communication software.

2. It takes 10 seconds or more to select a mot file:
   In this case, terminate XMODEM processing in the serial communication software and execute the sample program again. Then, execute the Update processing again. For terminating XMODEM processing, refer to the document for the serial communication software.
   When a XMODEM transmit file is selected, the serial communication software cannot process NAK properly. Then communication timings do not match and a data block transmission will be repeated. This is the cause of this issue.

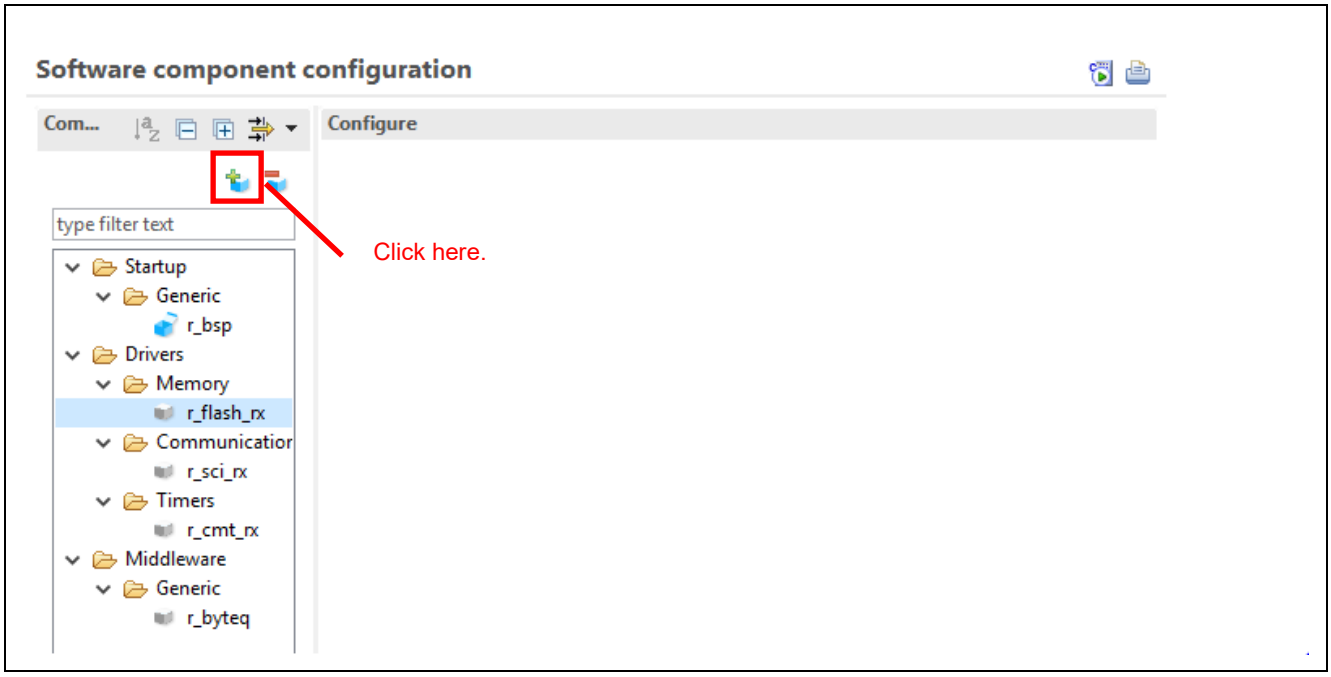
6.1.3 Nothing Displayed Even Though XMODEM Transfer Completed

Enter a character other than “1” or “2” in the serial communication software. If the echo back is displayed, it means the sample program has completed the XMODEM processing.

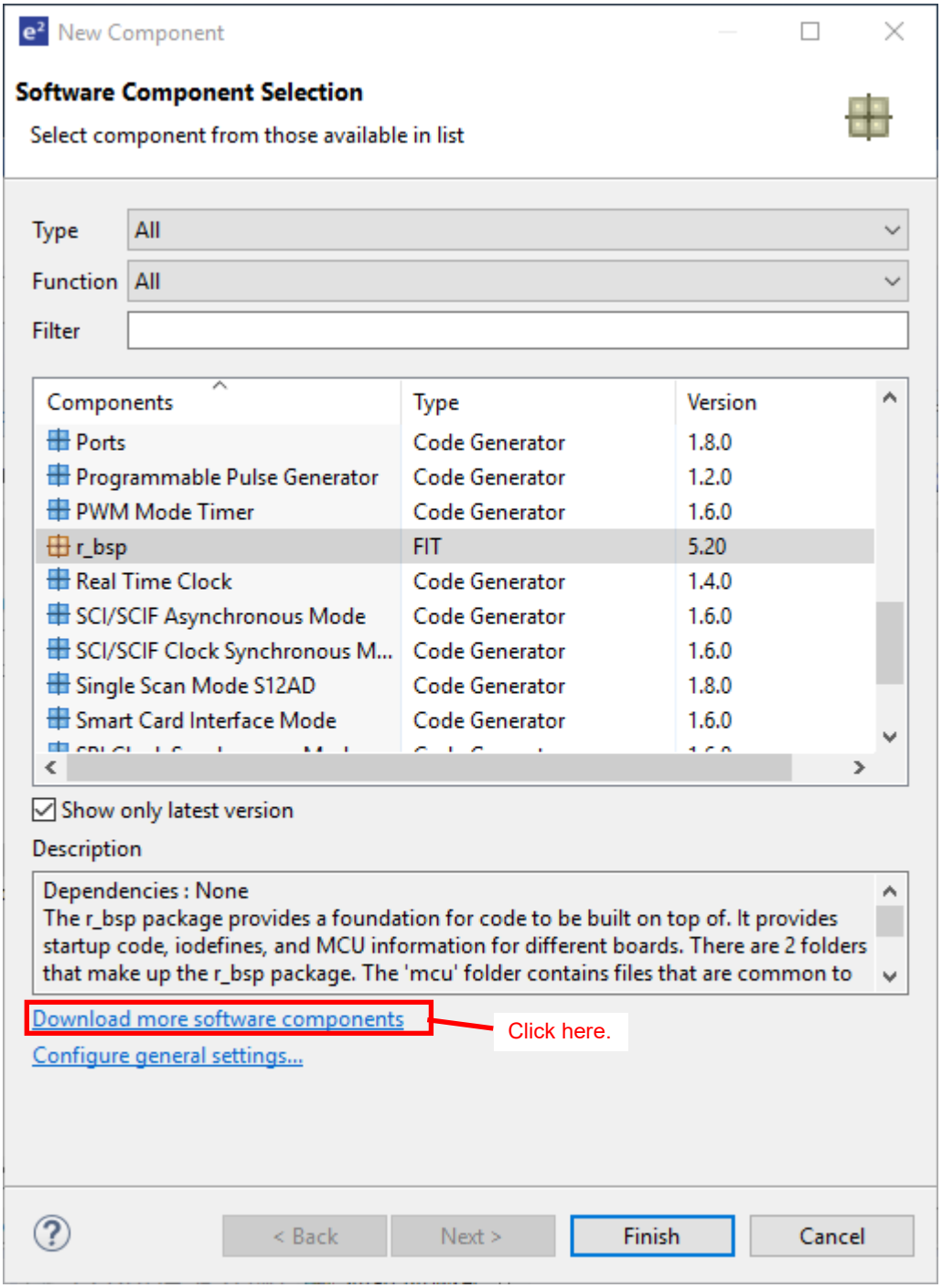
If it takes time to switch between XMODEM processing and normal communication processing, the serial communication software cannot receive the message sent by the sample program. This is the cause of this issue.

6.1.4 FIT Module Setting Items Not Displayed in Software Component Configuration

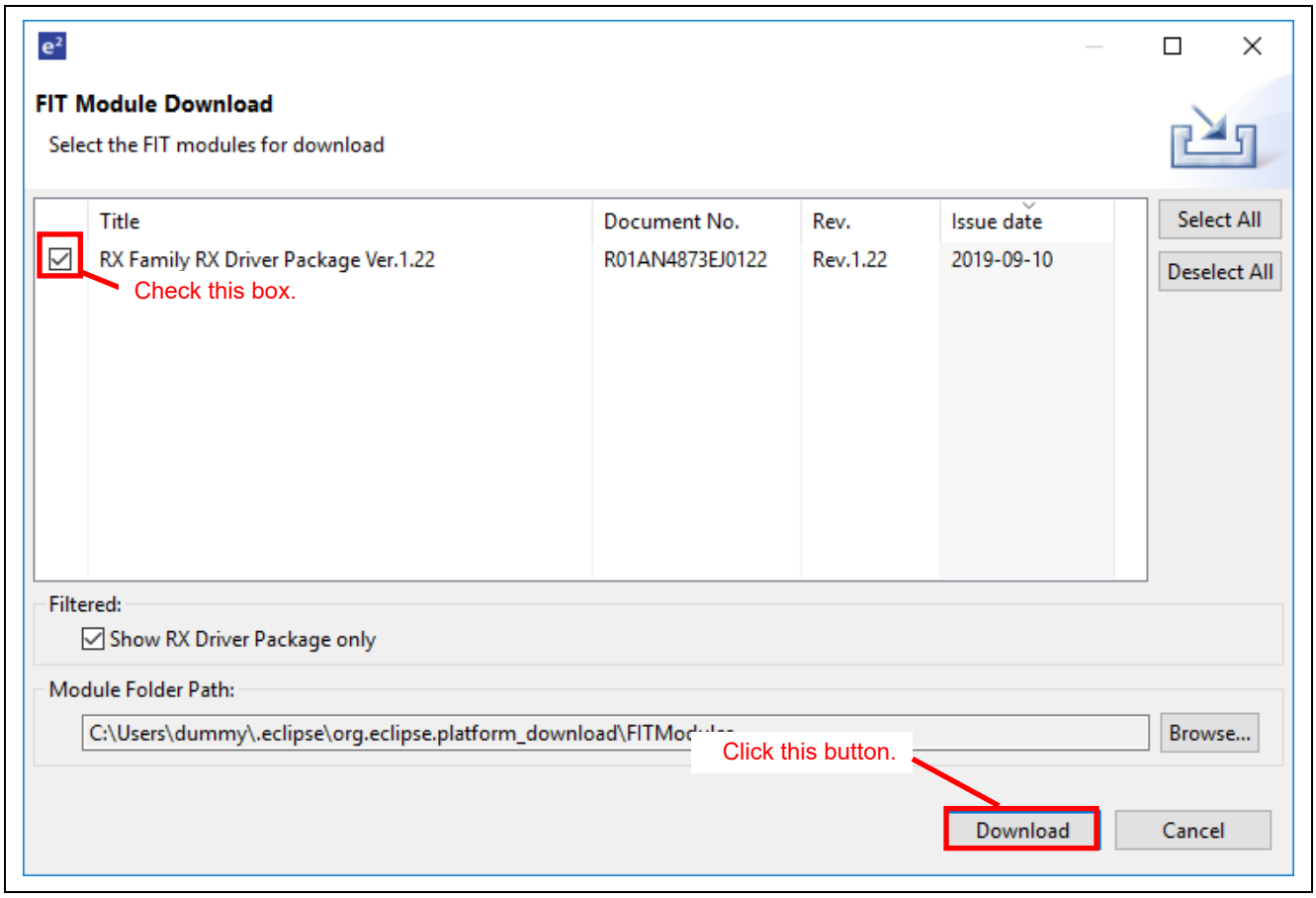
If the FIT modules are grayed out and no setting items are displayed in “Software component condifuration”, the version of the FIT modules used by the project may not have been downloaded to the folder recognized by Smart Configurator. To correct this, download the latest FIT modules. Click the “Add component” button.



Click “Download more software components”.

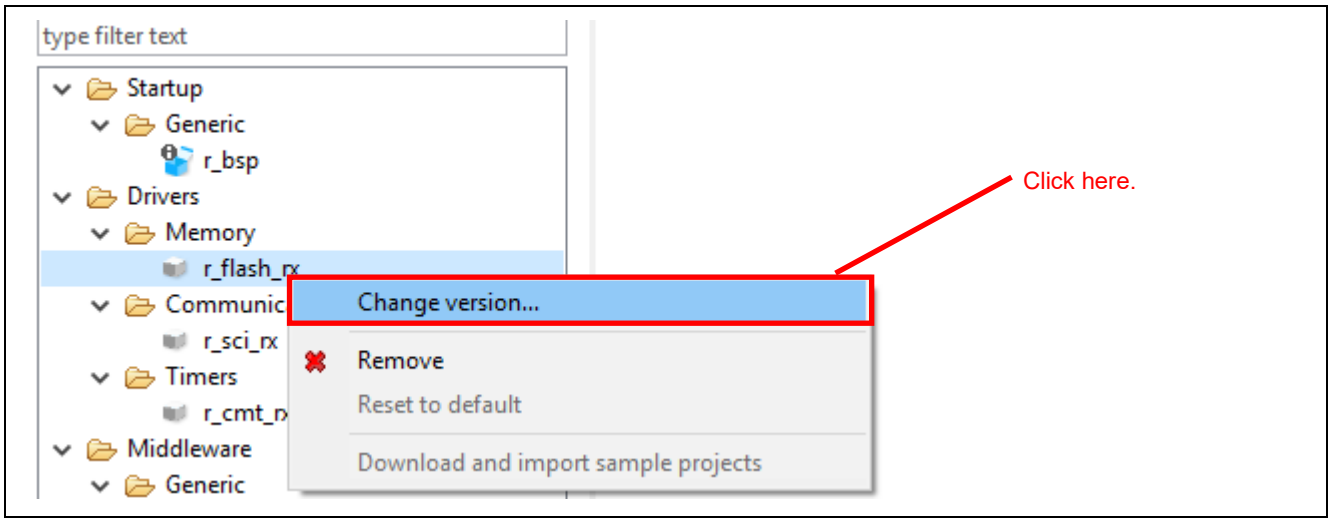


The latest RX Driver Package is displayed. Check the box and download it.

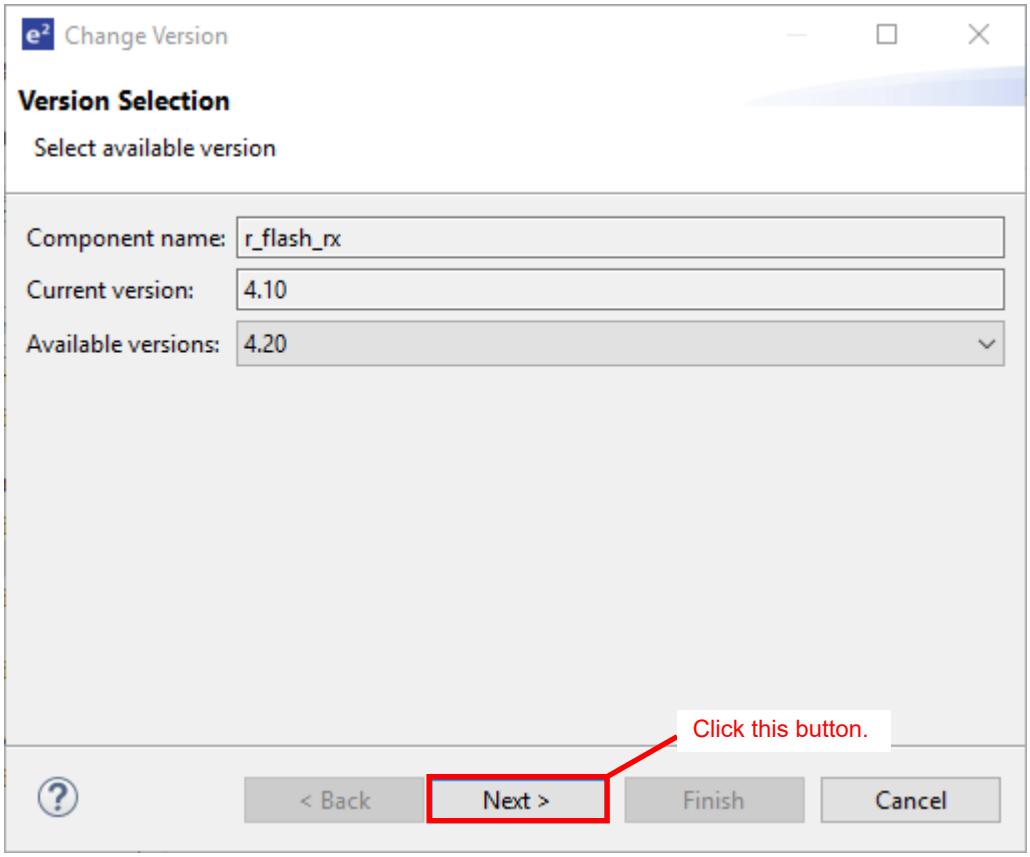


If the FIT modules are still grayed out after the download completes, upgrade the FIT modules used by the project to the latest version.

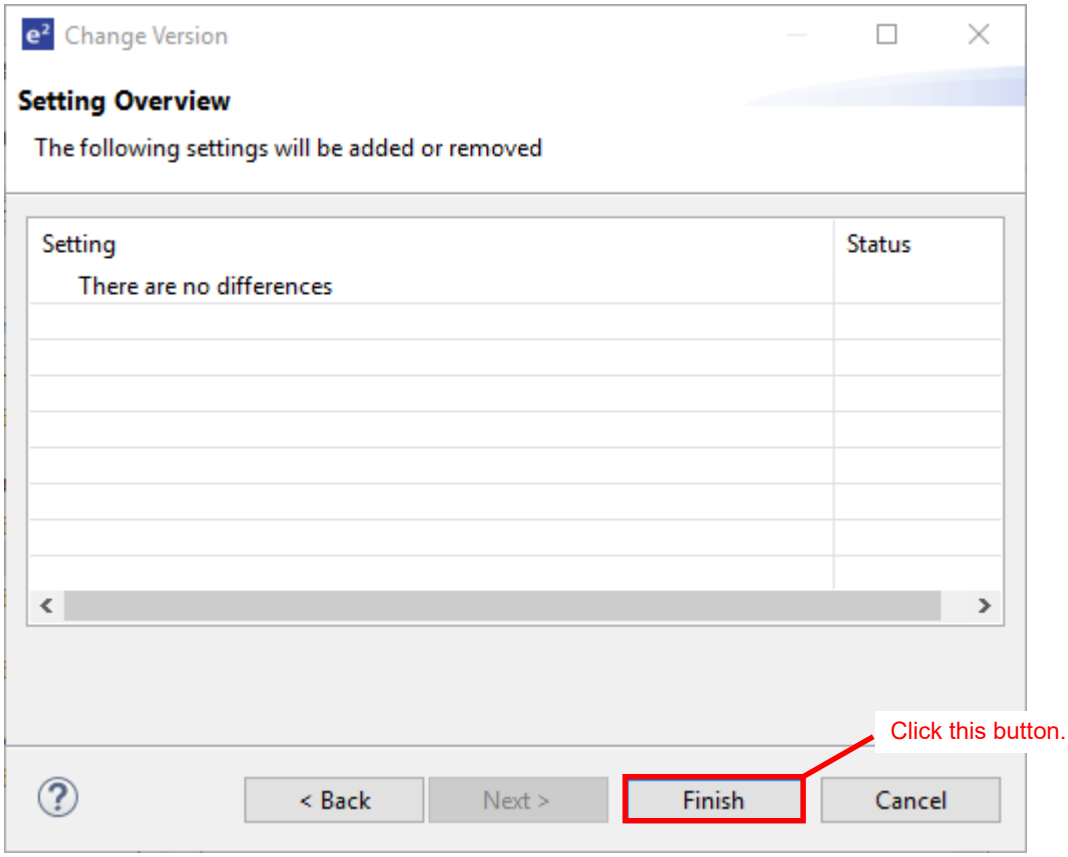
On the context menu, click “Change version”.



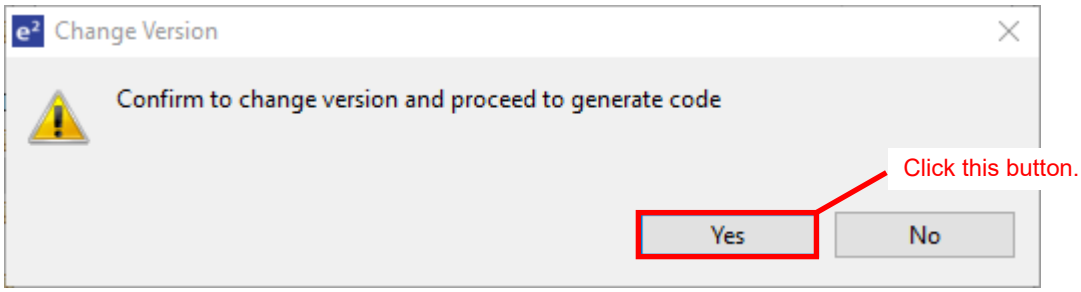
Click the “Next” button.



Click the “Finish” button.

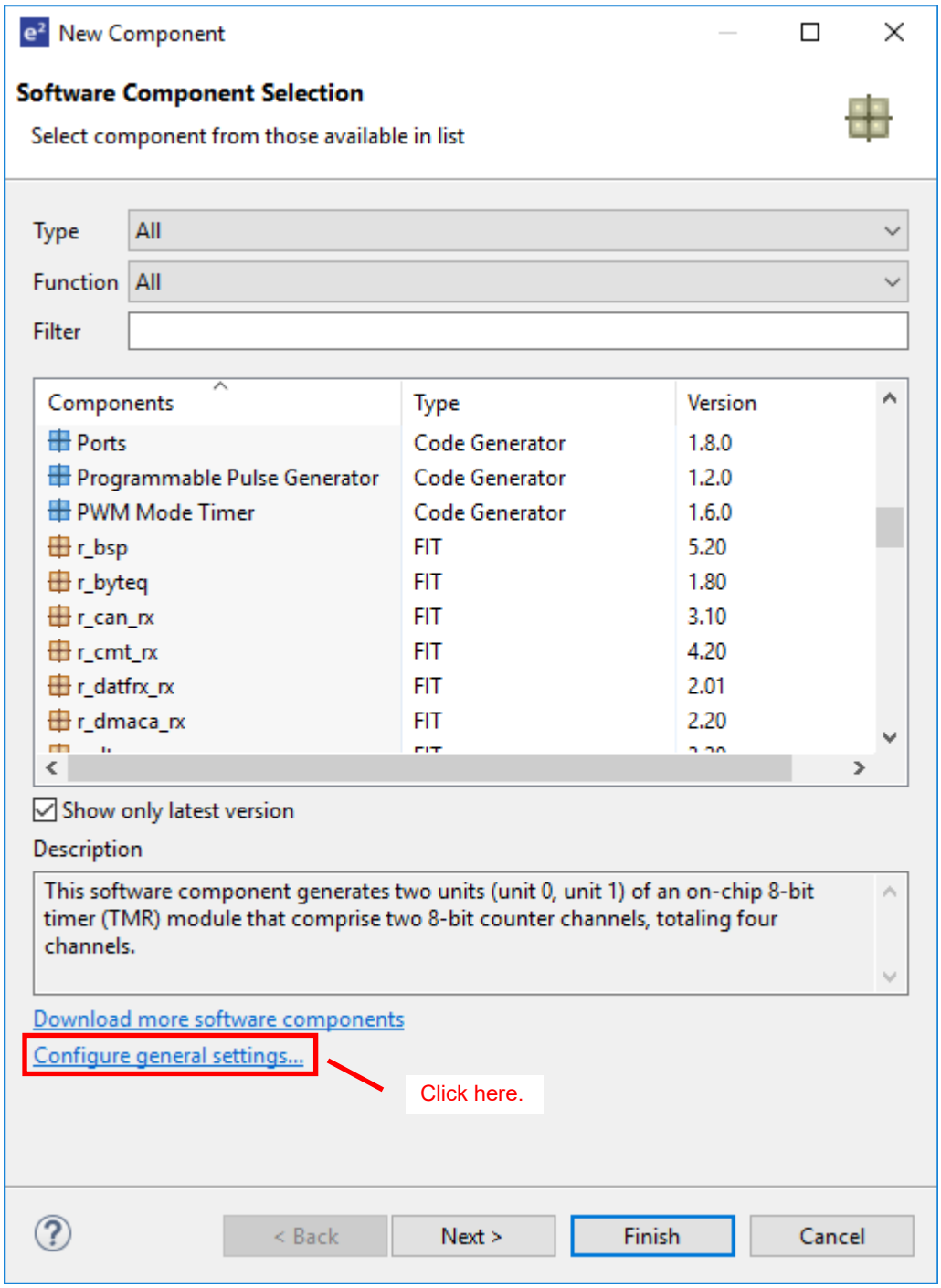


Click the “Yes” button.



6.1.5 SCI FIT Module Not Displayed in “New Component” Dialog Box

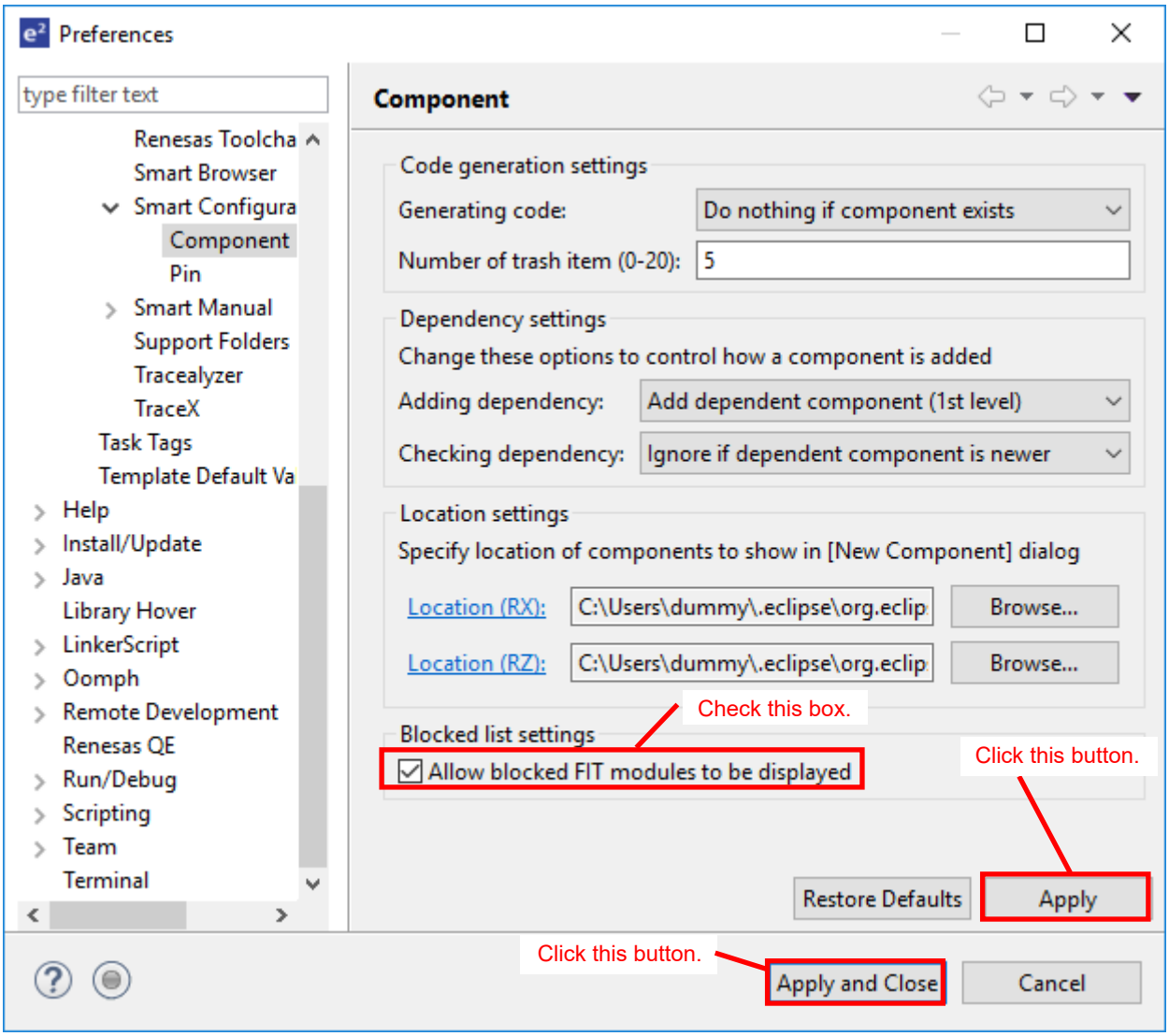
In the “New Component” dialog box, click “Configure general settings…”.



Check the box next to “Allow blocked FIT Modules to be displayed”.

Click the “Apply” button.

Click the “Apply and Close” button.



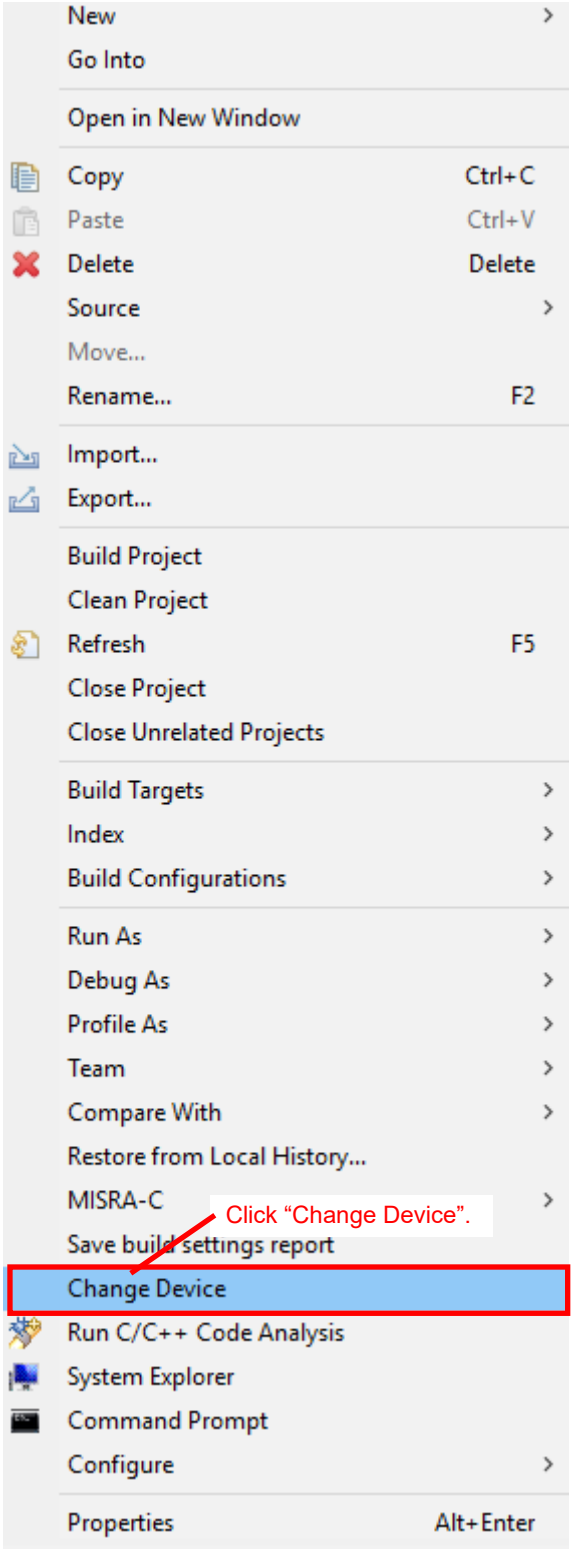
6.2 FAQ

6.2.1 Q: How do I change the device?

The example below shows how to modify the project for an RX65N MCU with 2 MB of ROM to work on an RX65N MCU with 1.5 MB of ROM.

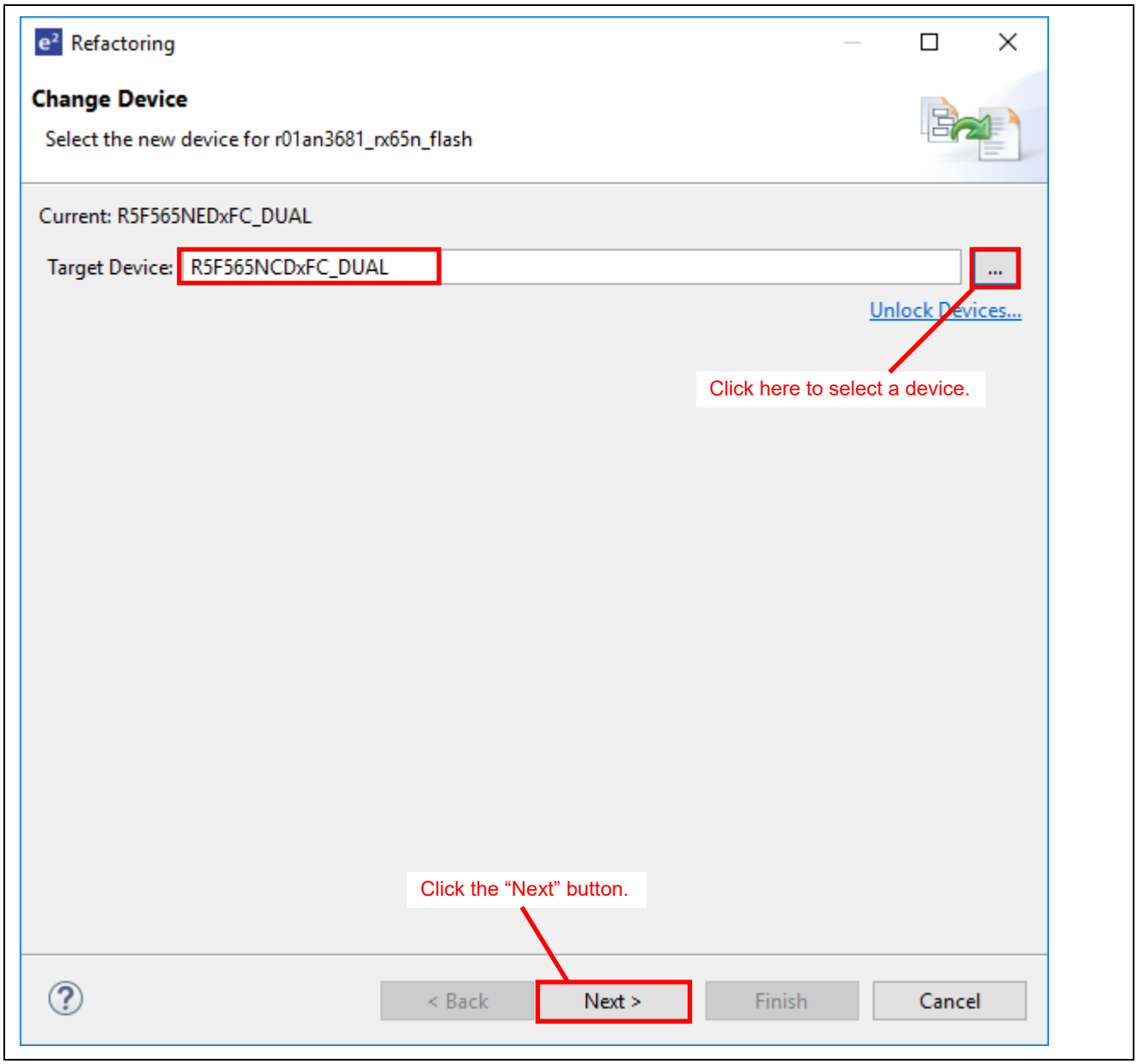
In Project Explorer, display the context menu of the target project.

Click “Change Device”.

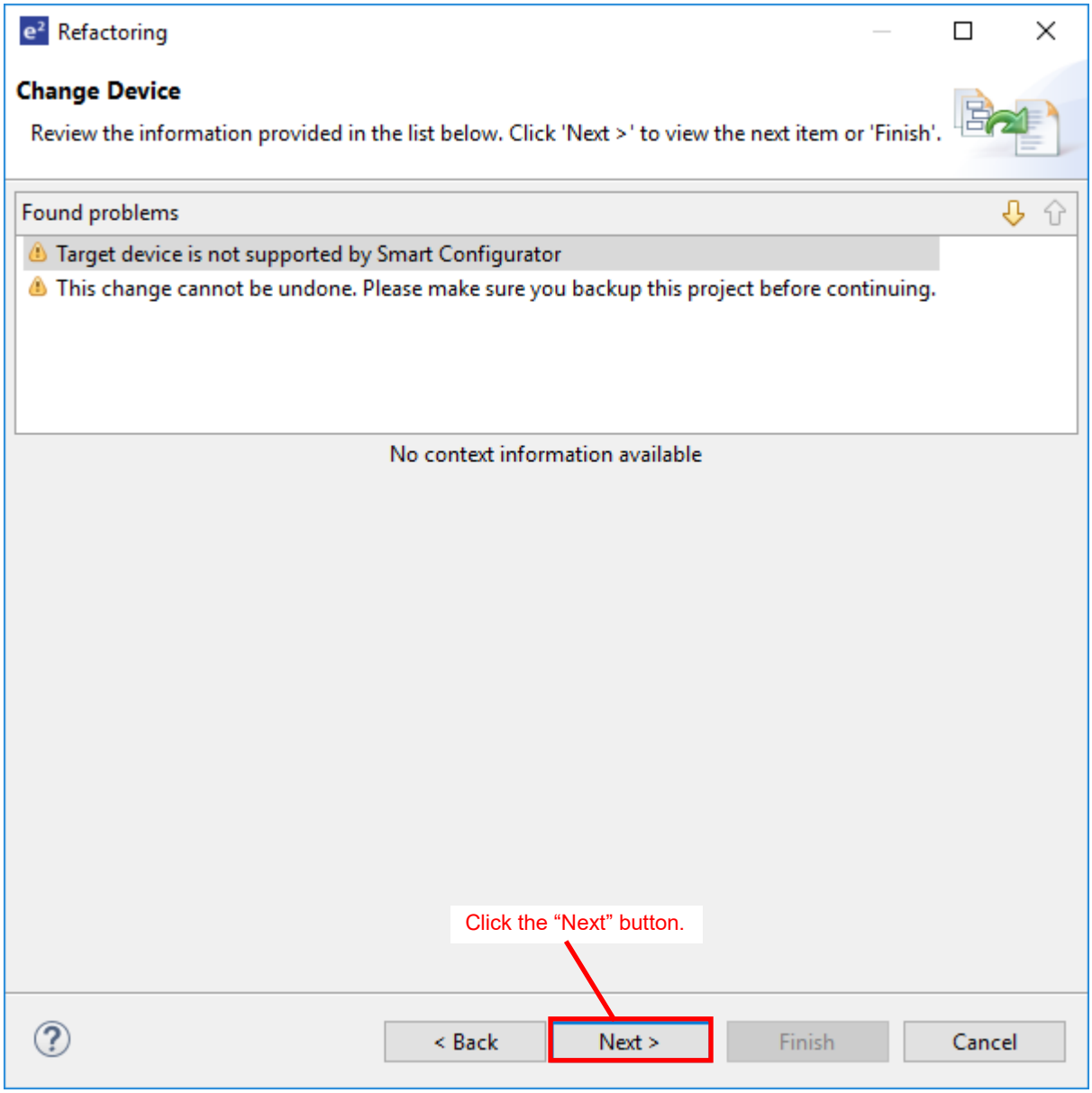

New
Go Into
Open in New Window
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects
Build Targets
Index
Build Configurations
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
MISRA-C
Save build settings report
Change Device
Run C/C++ Code Analysis
System Explorer
Command Prompt
Configure
Properties Alt+Enter
Click "Change Device".

Select the device to change to.

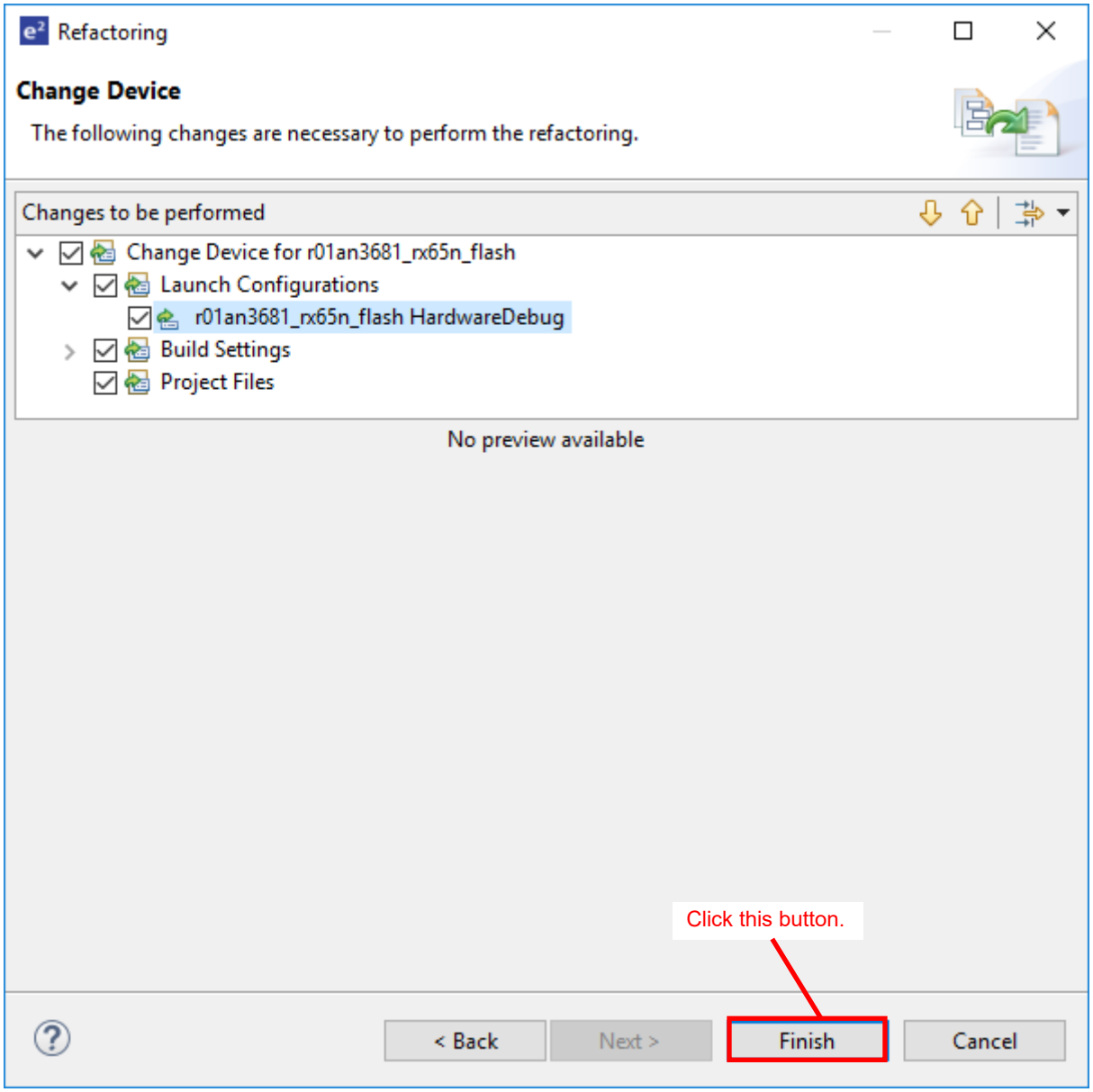
After making your selection, click the “Next” button.



A warning is displayed. Click the “Next” button to continue.



Click the “Finish” button.

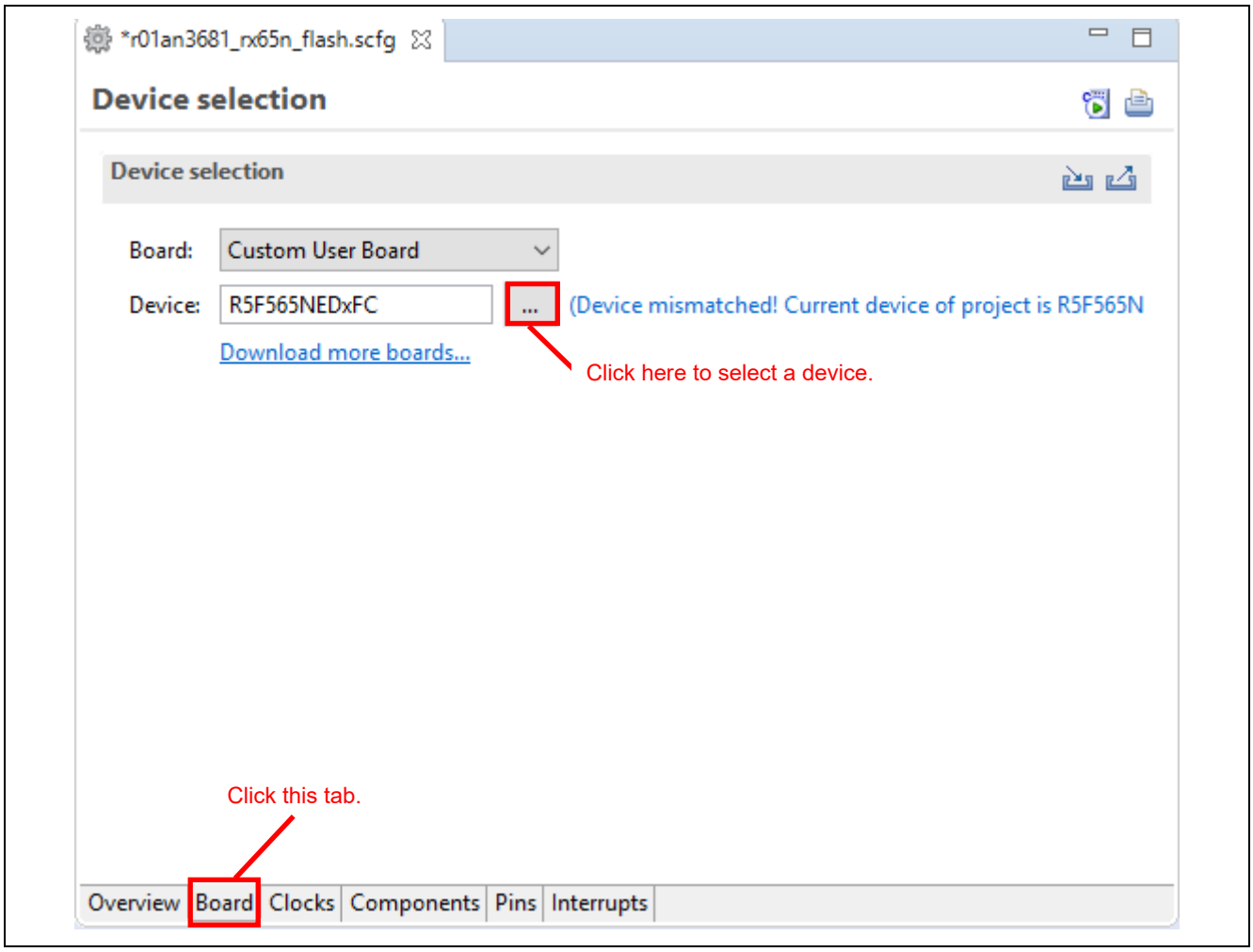


Change the selected device in Smart Configurator.

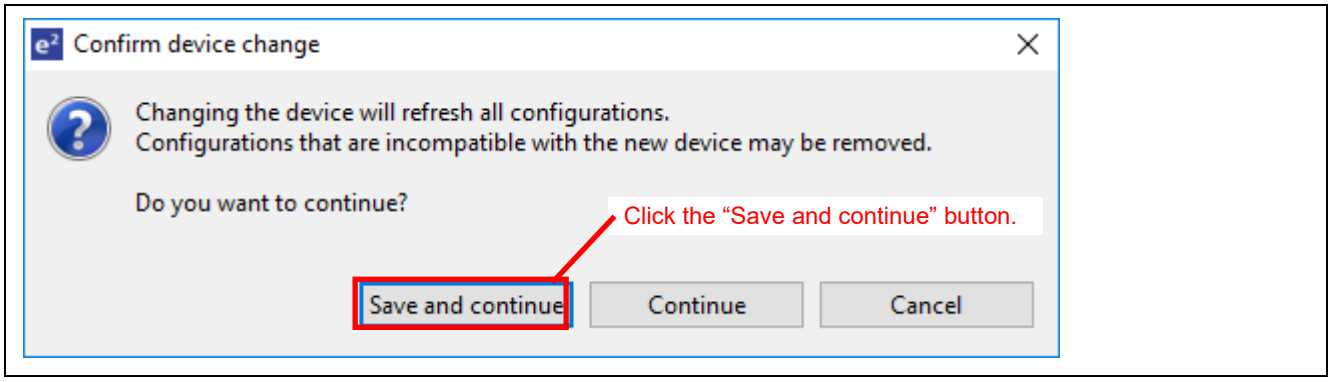
In Project Explorer, double-click r01an3681_rx65n_flash.scfg.

Click the “Board” tab to switch to it.

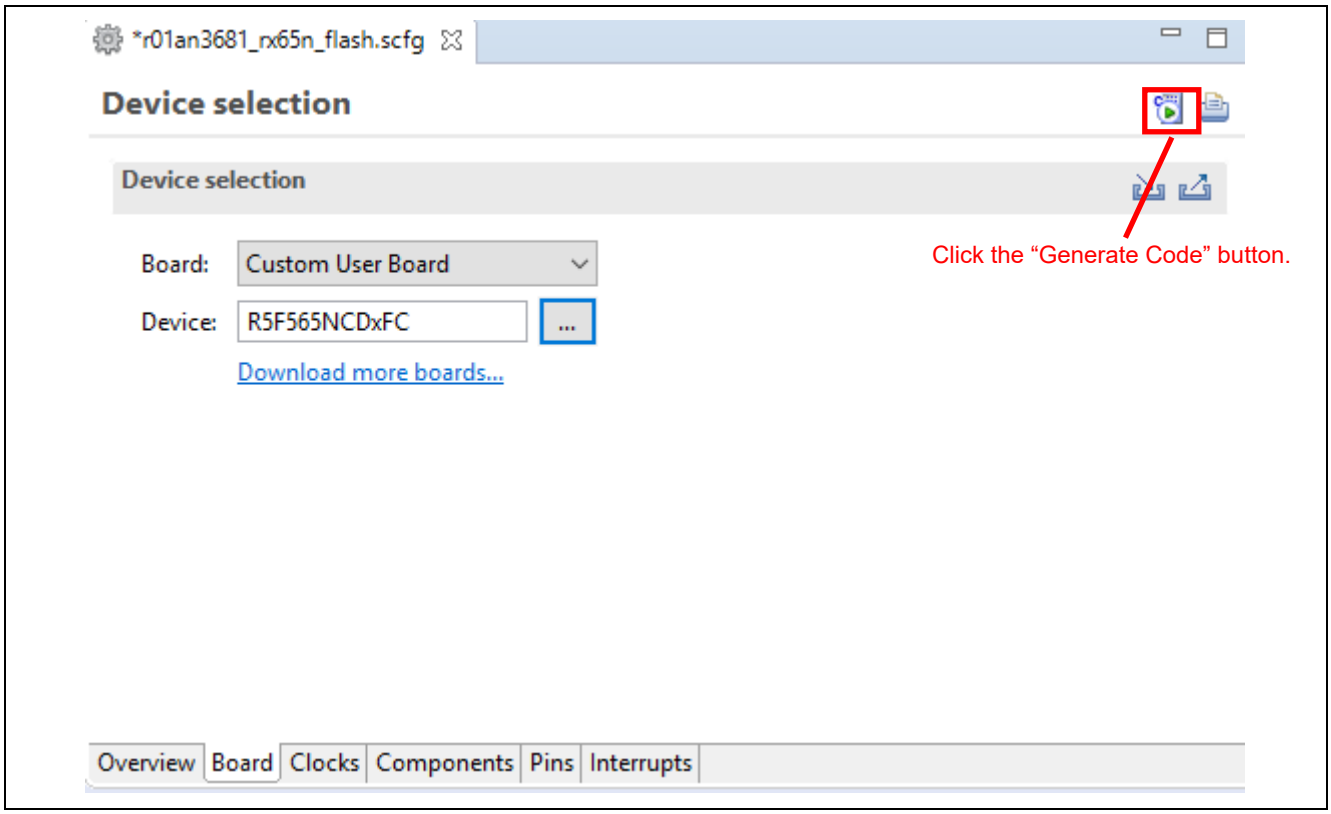
Select the same device selected previously here as well.



A dialog box to confirm the device change appears. Click the “Save and continue” button.



Click the “Generate Code” button.



The settings for the RPFRAM2 section added to the RAM area have been deleted. Refer to 3.4.3, Changes to “C/C++ Build” Settings, and add the RPFRAM2 section.

When changing the SCI channel used, refer to 3.4.1(3), Changes to SCI API settings, 3.4.2, Pin Settings, 5.3.2, Constants, and 5.3.5, Functions.

6.2.2 Q: How can I change the endianness to big endian?

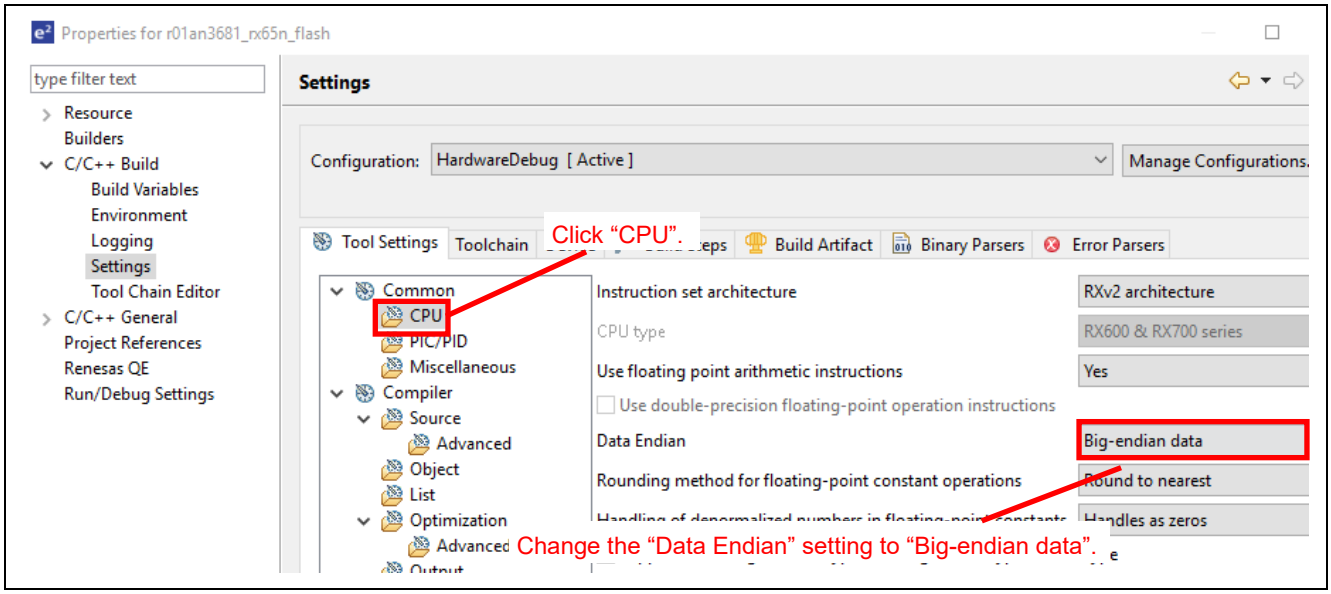
For this, settings for “C/C++ Build” and “Debug Configurations” need to be changed.

Changing the C/C++ configurations:

1. Open the “Properties” window of the project.
2. Select “C/C++ Build” >> “Settings” in the “Properties” window.
3. Select “Common” >> “CPU” in the “Tool Settings” tab.
4. Change the “Data Endian” setting to “Big-endian data”.

The HardwareDebug configuration and the Release configuration need to be changed.

Selecting “All configurations” changes both configurations at a time. Refer to 3.4.3, Changes to “C/C++ Build” Settings for the detailed procedure.

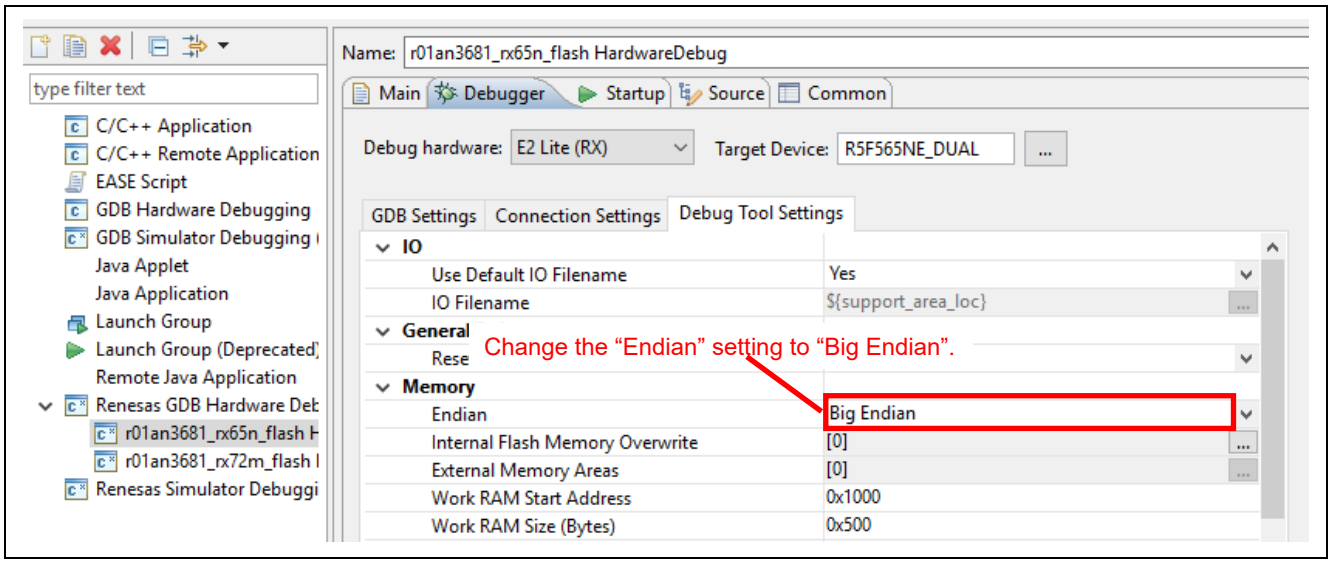


Clean the project with the “Clean” menu and then build the project with the “Build” menu. They can be done in the HardwareDebug configurations and the Release configurations, respectively.

Changing the debug configurations:

1. Select “Run” >> “Debug Configurations”.
2. Click the debug configuration to change under “Renesas GDB Hardware Debugging”.
3. Click the “Debug Tool Settings” tab in the “Debugger” tab.
4. Change the “Endian” setting under “Memory” to “Big endian”.

Refer to 4.3, Debug Configuration for the detailed procedure.



6.2.3 Q: How can I change the interface to an interface other than the SCI?

When changing the interface, functions in main.c and the pin setting function generated by the Smart Configurator need to be changed.

Table 6.1 lists the functions to be changed the settings. Change values of these functions to values appropriate to the interface used. sci_callback is the callback function to detect completion of a transmission. Change the method to detect completion of a transmission according to the interface. If there is no method to detect completion of a transmission, add necessary processing such as inserting a wait processing in the send_string_sci function or in the point where the send_string_sci function is called.

Use the smart configurator to perform the pin setting for the FIT module. The setting can be done in the “Properties” in the “Component” tab.

Table 6.1 Functions to be Changed the Settings

Function Name	Application	File
main	Initialization of the interface and reception other than XMODEM protocol.	main.c
sci_callback	Detection for completion of a string transmission	main.c
send_string_sci	String transmission	main.c
send_byte_xm	1-byte transmission in XMODEM protocol. The transmission result (success or error) is returned as the return value.	main.c
recv_byte_xm	1-byte reception in XMODEM protocol. Timeout occurs after 10 seconds elapses. The reception result (success, reception error, or timeout) is returned as the return value.	main.c
R_SCI_PinSet_SCI8	Performing the pin setting	r_sci_rx_pinset.c

6.2.4 Q: How can I erase the option-setting memory?

Perform the following procedure with the Renesas Flash Programmer.

1. Click the “Operation Setting” tab.
2. Select “Command” >> “Erase”.
3. Specify “Chip Erase” for the “Erase Option”.
4. Uncheck the following items: “Program”, “Checksum”, and “Verify”.
5. Click the “Operation” tab.
6. Click the “Start” button.
7. Now the option-setting memory is erased.

Note that the data flash memory and the code flash memory are also erased at the same time as the option-setting memory. If “Chip Erase” is not selected as the erase option, the Renesas Flash Programmer does not erase the option-setting memory.

Refer to the RX65N Group, RX651 Group User’s Manual: Hardware and RX72M Group User’s Manual: Hardware for details on the option-setting memory.

6.2.5 Q: How can I specify to read the other bank of the startup bank?

In the debug perspective, select “Window” >> “Show View” >> “Memory”. In the Memory Monitors pane, click the “Add Memory Monitor” button to open the “Memory Monitor” dialog. Enter the address for the other bank of the startup bank in the “Memory Monitor” dialog.

Also, confirm that in the debug configuration “Debugger” >> “Debug Tool Settings” >> “Debug program that overwrites on-chip program ROM” is set to “Yes”. If it is set to “No”, the overwritten content will not be updated in the debugger.

6.2.6 Q: How can I check the bank currently specified as the startup bank?

The startup bank can be checked by reading the BANKSEL register immediately after the startup. The register value changed in the BANKSEL register indicates the startup bank after a reset.

Refer to the RX65N Group, RX651 Group User’s Manual: Hardware and RX72M Group User’s Manual: Hardware for details on the BANKSEL register.

6.2.7 Q: How can I import the sample program into CS+?

When using CS+, follow the steps below to import the sample program into CS+.

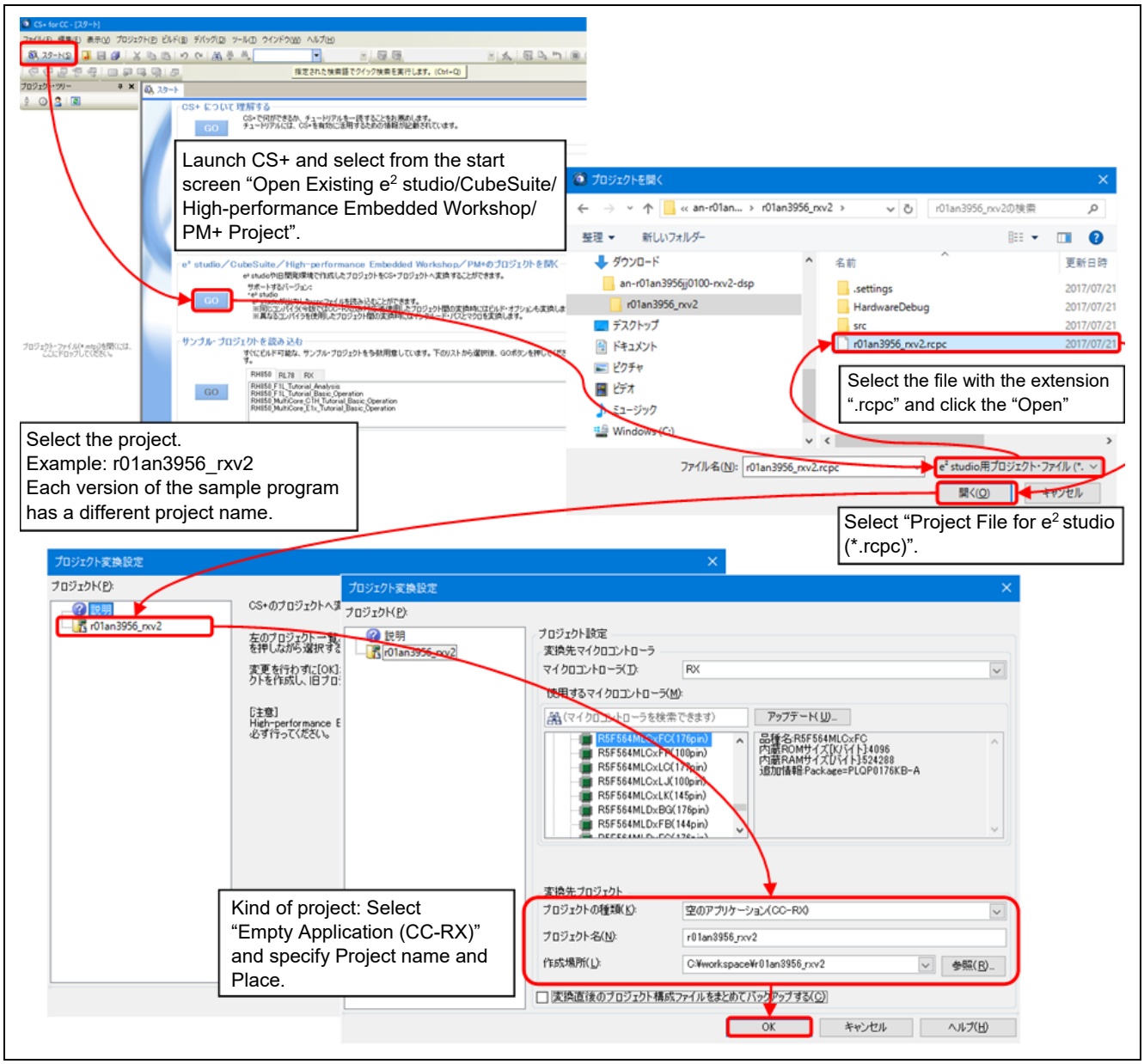


Figure 6.1 Importing a Project into CS+

7. Others

7.1 Notes on Using the Evaluation Version of C/C++ Compiler Package for RX Family

When using the evaluation version of C/C++ compiler package for RX Family, the evaluation period and usage limitations apply. When the evaluation period expires, the size of linkable object is reduced to 128 KB or less and this may cause the incorrect generation of the load module.

For details, refer to the following software tool page for evaluation versions on the Renesas website:

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct. 2, 2017	—	First edition issued
1.10	Dec. 18, 2017	—	Since the flash FIT module Rev 3.2 bundled with Rev 1.00 was the evaluation version, it was updated.
1.20	Sep. 6, 2019	—	Added support for RX72M Group. Changed integrated development environment to e^2 studio version 7.5.0. Changed description of FIT module configuration to procedure using Smart Configurator. Changed description of changing ROM capacity to procedure using functionality of e^2 studio. Changed debugger hardware to E2 Emulator Lite.

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. 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.4.0-1 November 2017)

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/.

© 2019 Renesas Electronics Corporation. All rights reserved.