RX113 Group

On-chip Flash Memory Programming Solution using USB Memory RX Driver Package Application

R01AN2537EJ0102
Rev.1.02
Mar 31, 2017

Introduction

This document is an application note of the on-chip flash memory programming solution using USB Memory.

This application note includes the main program that writes the program stored in the USB memory into the RX113 on-chip flash memory and execute it.

The main program of the application note is used in combination with FAT file system, USB driver, Flash memory module included in the RX110, RX111, RX113, RX231 Group RX Driver Package.

Target Device

RX113 Group

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

Related Documents

- Firmware Integration Technology User’s Manual (R01AN1833EU)
- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685EU)
- RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723EU)
- RX Family Adding Firmware Integration Technology Modules to CubeSuite+ Projects (R01AN1826EJ)

Contents

1. Overview

1.1 This Application Note

This application note describes the procedure for main program evaluation by combining the Board Support Package (referred to as “BSP”), Flash memory (referred to as “Flash API”), USB driver (Host Mass Storage Class Driver “USB HMSC” and “Basic Firmware”), M3S-TFAT-Tiny FAT file system (referred to as “TFAT”) of Firmware Integration Technology (referred to as “FIT”) modules included in the RX110, RX111, RX113, RX231 Group RX Driver Package.

This application note operates on the Renesas Starter Kit for RX113 (referred to as “RSK” in the remainder of this document)

The program (Sample program) executed after the programming is also available. The sample program is stored in the “demo” folder in each project.

1.2 Operating Environment

The table below lists the operating environment of the main program and the sample program.

Table 1-1 Operating Environment

Items	Contents
Microcontroller	RX113 Group
Evaluation board	Renesas Starter Kit for RX113 (Part No.: R0K505113S000BE)
Integrated development environment (IDE)	e2 studio, V4.1.0 or later
C Compiler	RX Family C/C++ Compiler Package V2.03.00 or later
Emulator	E1
Endian	Little endian
RX Driver Package	RX110, RX111, RX113, RX231 Group RX Driver Package Ver.1.01 (R01AN2670EJ)*

Note:* Operation of this application note has been verified when the modules in the RX Driver Package mentioned above are incorporated. If any of the modules used in this application note are replaced with a different module, the user must verify the operation.

1.3 Module Structure

Figure 1-1 shows Module structure of the main program, and Table 1-2 Modules lists the FIT modules to be included in the main program. Some modules are included in the sample program.

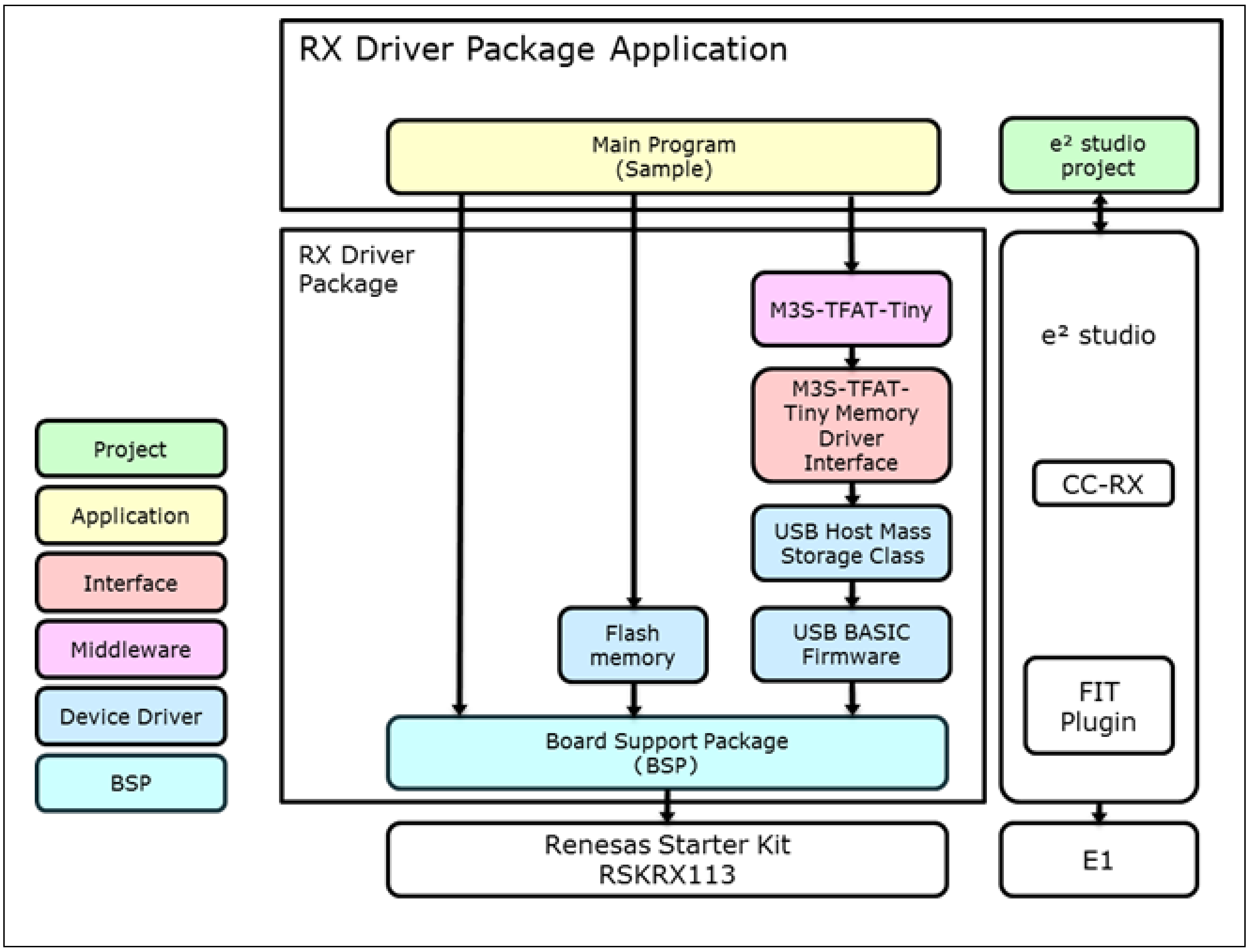


Figure 1-1 Module Structure

Table 1-2 Modules

Type	Module	FIT Module Name	Rev.
BSP	Board Support Package (BSP)	r_bsp	3.01
Middleware	M3S-TFAT-Tiny FAT file system (TFAT)	r_tfat_rx	3.02
	M3S-TFAT-Tiny Memory Driver Interface	r_tfat_driver_rx	1.02
Device Driver	USB Basic Firmware	r_usb_basic	1.01
	USB Host Mass Storage Class (USB HMSC)	r_usb_hmsc	1.01
	LCD controller/driver (LCDC)	r_lcdc_rx	1.00
	Flash memory (Flash API)	r_flash_rx	1.30
Application	Main program FIT module	r_flash_writer_rx113	1.00

1.4 File Structure

Figure 1-2 shows the file structure used in this application note.

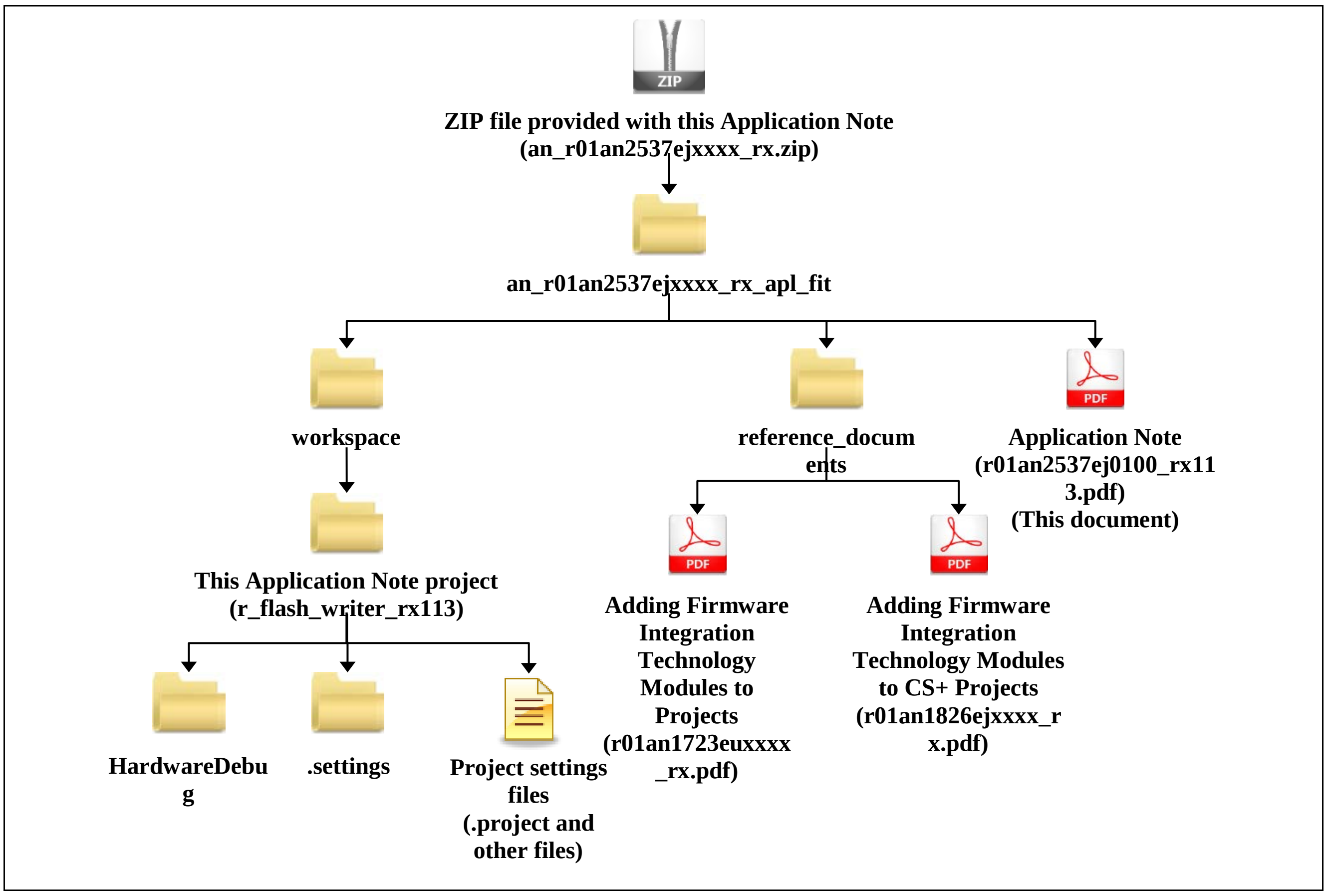

ZIP
ZIP file provided with this Application Note
(an_r01an2537ejxxxx_rx.zip)
an_r01an2537ejxxxx_rx_apl_fit
workspace
reference_documents
PDF
Application Note
(r01an2537ej0100_rx113.pdf)
(This document)
This Application Note project
(r_flash_writer_rx113)
Adding Firmware Integration Technology Modules to Projects
(r01an1723euxxxx_rx.pdf)
Adding Firmware Integration Technology Modules to CS+ Projects
(r01an1826ejxxxx_rx.pdf)
HardwareDebug
.settings
Project settings files
(.project and other files)

Figure 1-2 File Structure

When the ZIP file provided with this application note is decompressed, a folder with the same name is created, and the various folders and files are created within that folder

The “workspace” folder is the project to build “On-chip Flash Memory Programming application using the USB memory”. To use the e^2 studio, import the project into the workspace.

Documents that describe using the FIT modules in various development environments are included in the **reference_documents** folder. The document “Adding Firmware Integration Technology Modules to Projects” (R01AN1723EU) describes the method for including the FIT modules, as a FIT plugin, in an e^2 studio project. The document “Adding Firmware Integration Technology Modules to CS+ Projects” (R01AN1826EJ) describes the method for including the FIT modules in a CubeSuite+ project.

2. Acquiring a Development Environment

2.1 Acquire and install e^2 studio

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

http://japan.renesas.com/e2studio_download

This document requires you to use e^2 studio V4.1.0 or later. If the version older than V4.1.0 is used, some functions of the e^2 studio may not be available. For download, obtain the latest version of the e^2 studio on the website

2.2 Acquire a Compiler Package

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

http://japan.renesas.com/e2studio_download

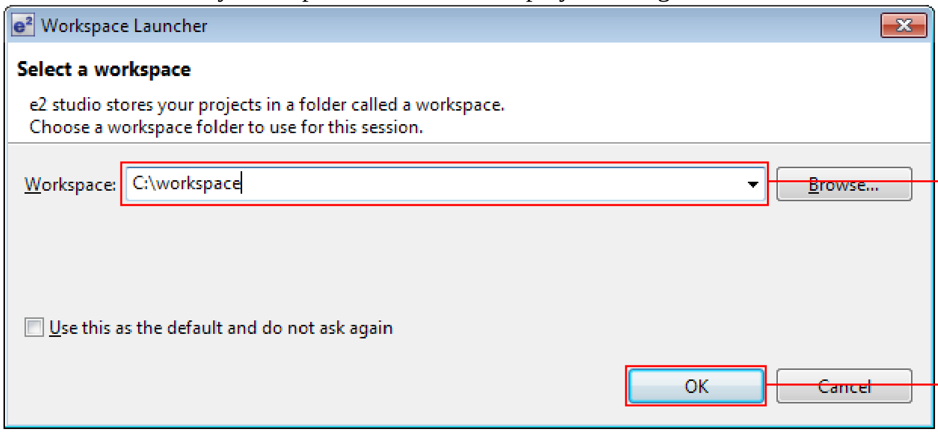
3. Building a Project

This application note includes environment-built project. The procedure to import a Project using e2 studio Smart browser is described below.

3.1 Create a Workspace

1. Start e^2 studio.

2. Enter an arbitrary workspace folder in the displayed dialog box and click [OK].



Workspace Launcher

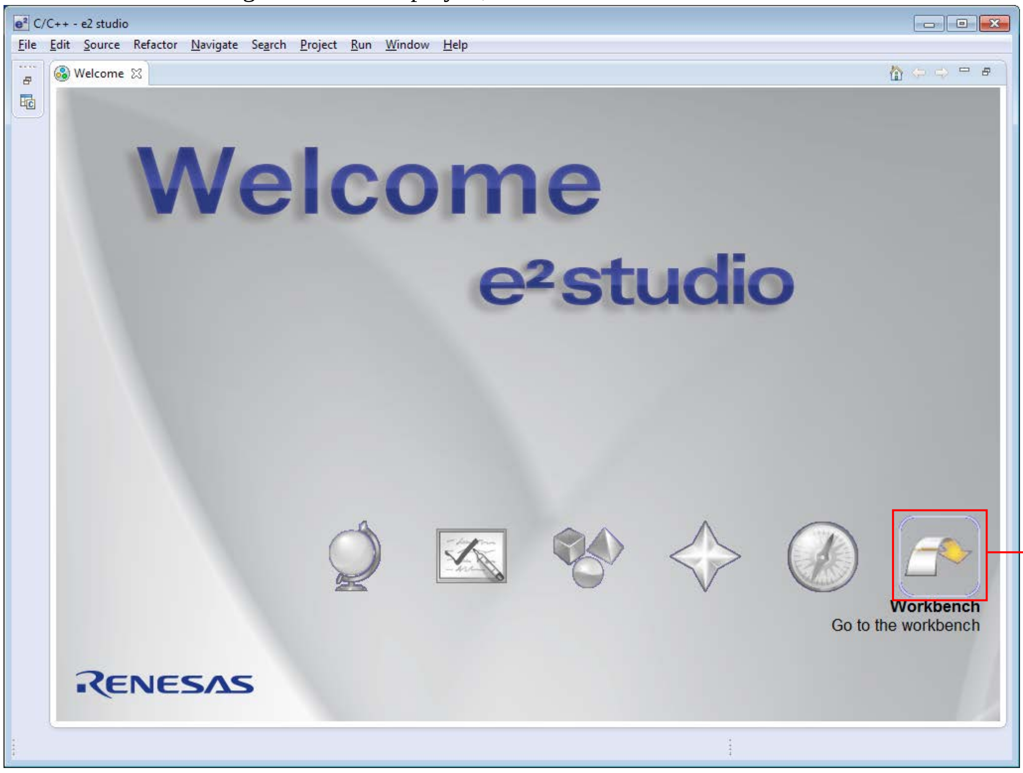
Select a workspace

e2 studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\workspace

Browse...
Use this as the default and do not ask again
OK
Cancel
Enter a workspace folder.
Click [**OK**].

3. When the following window is displayed, click [Workbench].

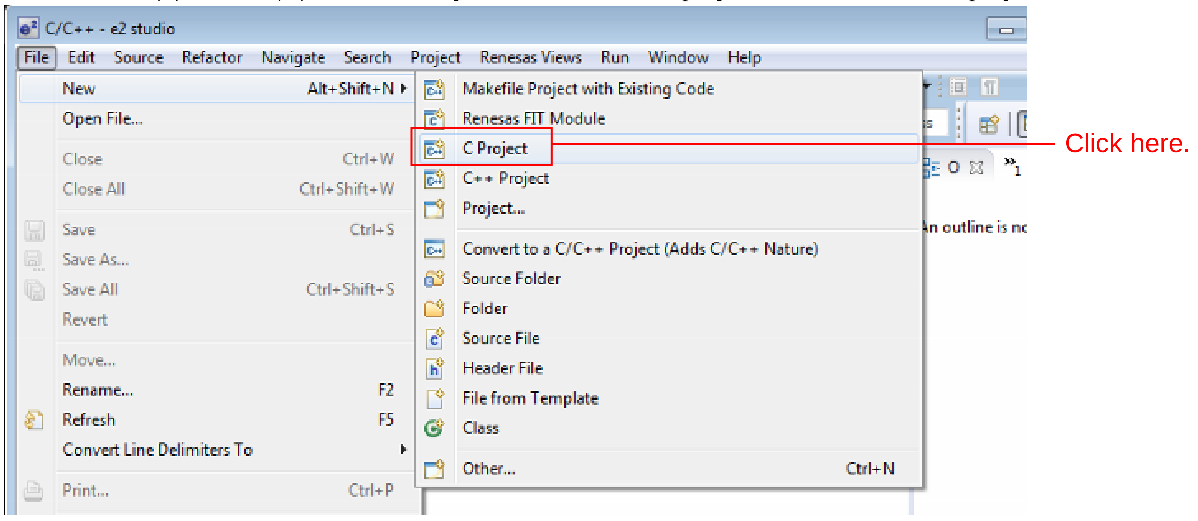


3.2 Create a Project

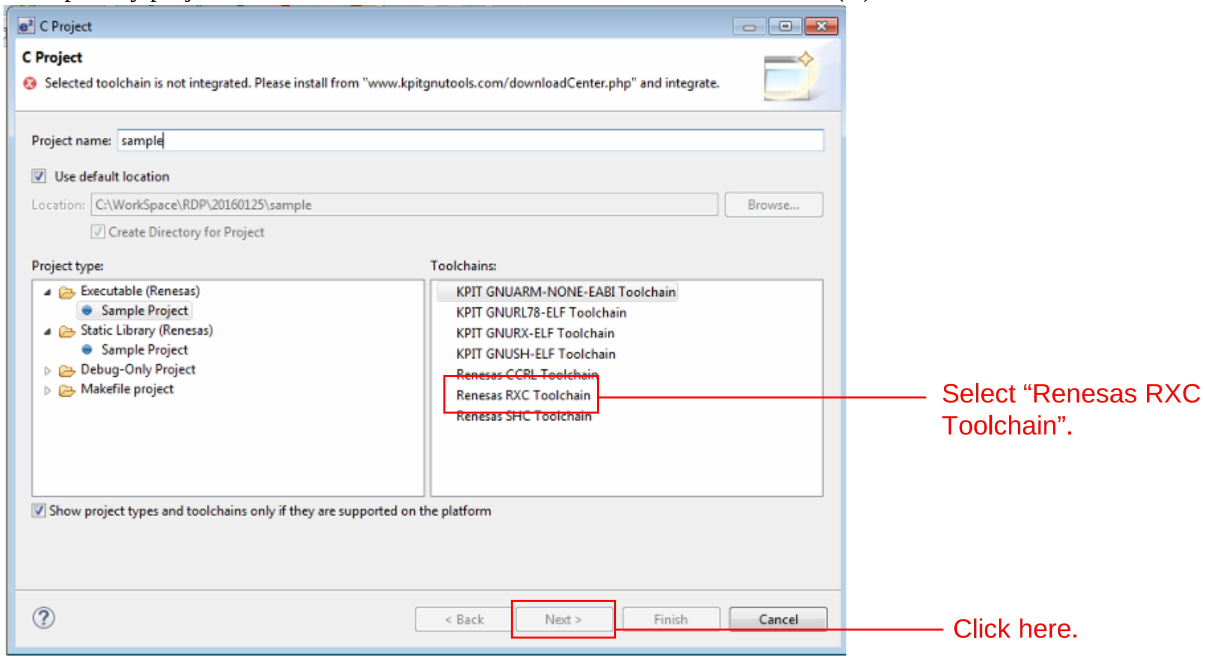
When using the Smart browser function, the target project or file needs to be selected. To use this function, create the project that specified the target device (Note 1).

Note 1: The project to be created here is a dummy to use the Smart browser.

1. Click [File (F)], [New (N)], then [C Project] to create new C project. Start [Create New project wizard].



2. Input any project name and select [Renesas RXC Toolchain]. Click [Next (N)].



C Project

Selected toolchain is not integrated. Please install from "www.kpitgnutools.com/downloadCenter.php" and integrate.

Project name: sample

☑ Use default location

Location: C:\WorkSpace\RDP\20160125\sample Browse...

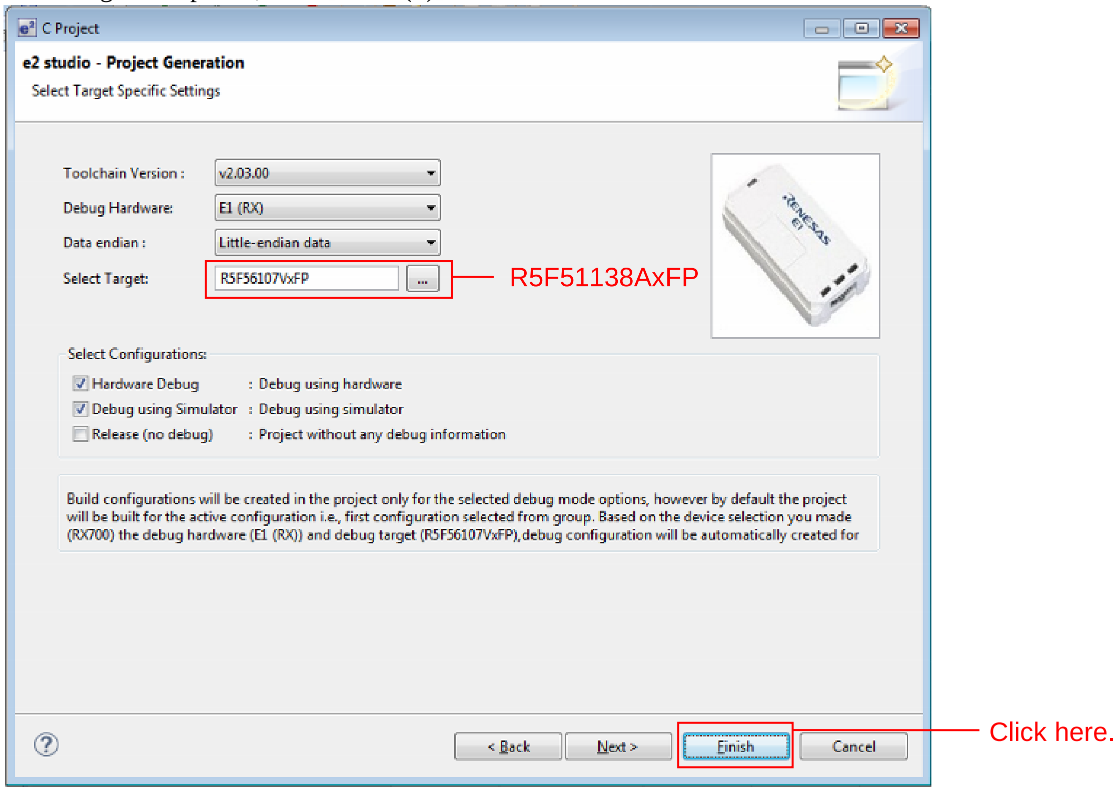
☑ Create Directory for Project

Project type:
- Executable (Renesas)
  - Sample Project
- Static Library (Renesas)
  - Sample Project
- Debug-Only Project
- Makefile project

Toolchains:
- KPIT GNUARM-NONE-EABI Toolchain
- KPIT GNURL78-ELF Toolchain
- KPIT GNURX-ELF Toolchain
- KPIT GNUSH-ELF Toolchain
- Renesas CCRL Toolchain
- Renesas RXC Toolchain
- Renesas SHC Toolchain

☑ Show project types and toolchains only if they are supported on the platform

3. Set Select Target to R5F51138AxFP for RX113 100 Pin device. For other items, any setting is OK. When the setting is completed, click [Finish(F)].


C Project

e2 studio - Project Generation

Select Target Specific Settings

Toolchain Version : v2.03.00

Debug Hardware: E1 (RX)

Data endian : Little-endian data

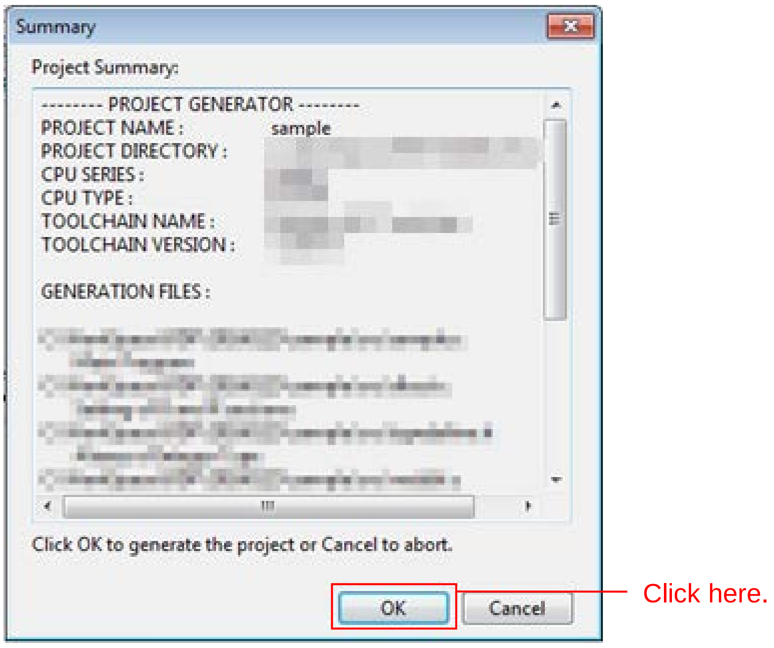
Select Target: R5F56107VxFP

Select Configurations:

- [✓] Hardware Debug : Debug using hardware
- [✓] Debug using Simulator : Debug using simulator
- [ ] Release (no debug) : Project without any debug information

Build configurations will be created in the project only for the selected debug mode options, however by default the project will be built for the active configuration i.e., first configuration selected from group. Based on the device selection you made (RX700) the debug hardware (E1 (RX)) and debug target (R5F56107VxFP),debug configuration will be automatically created for

4. Click [OK].


Summary

Project Summary:

```
-------- PROJECT GENERATOR --------
PROJECT NAME :          sample
PROJECT DIRECTORY :
CPU SERIES :
CPU TYPE :
TOOLCHAIN NAME :
TOOLCHAIN VERSION :

GENERATION FILES :
```

Click OK to generate the project or Cancel to abort.

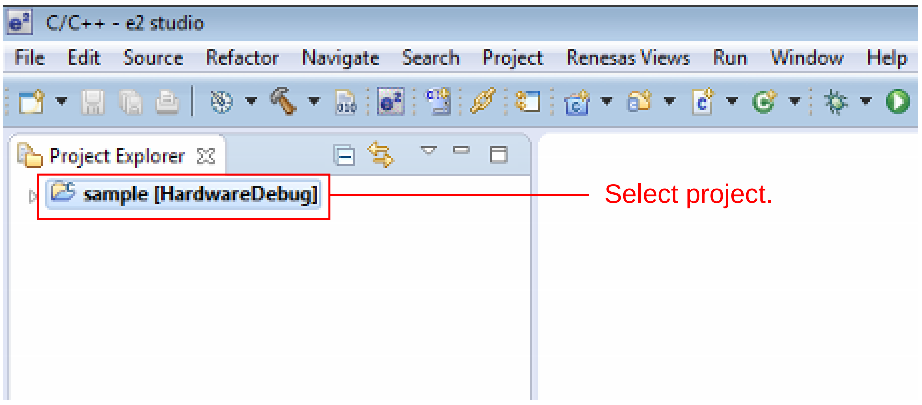
3.3 Import a Project

Import the project of Main program in the workspace created.

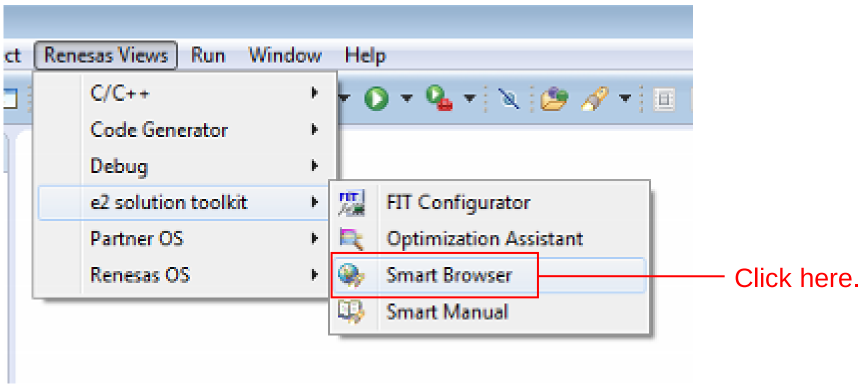
This application note includes the projects that select a file by;

- Project that selects a file by switch

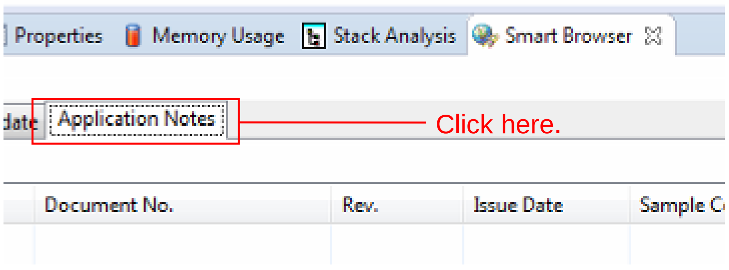
1. Select the project created in ”3.2 Create a Project” from Project explorer.


C/C++ - e2 studio
File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Project Explorer
sample [HardwareDebug]
Select project.

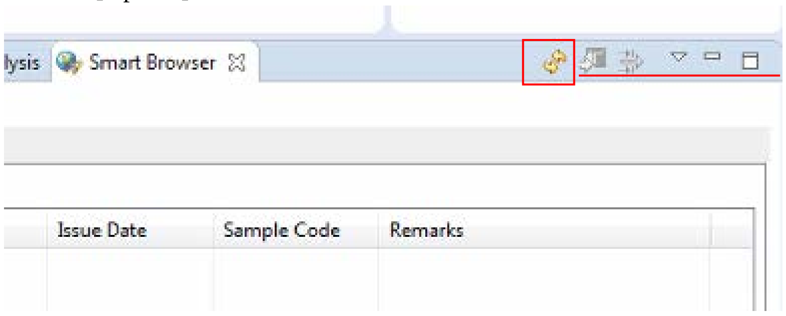
2. Click on [Renesas Views] → [e2 Solution Tool kit] → [Smart browser] to start the Smart browser.



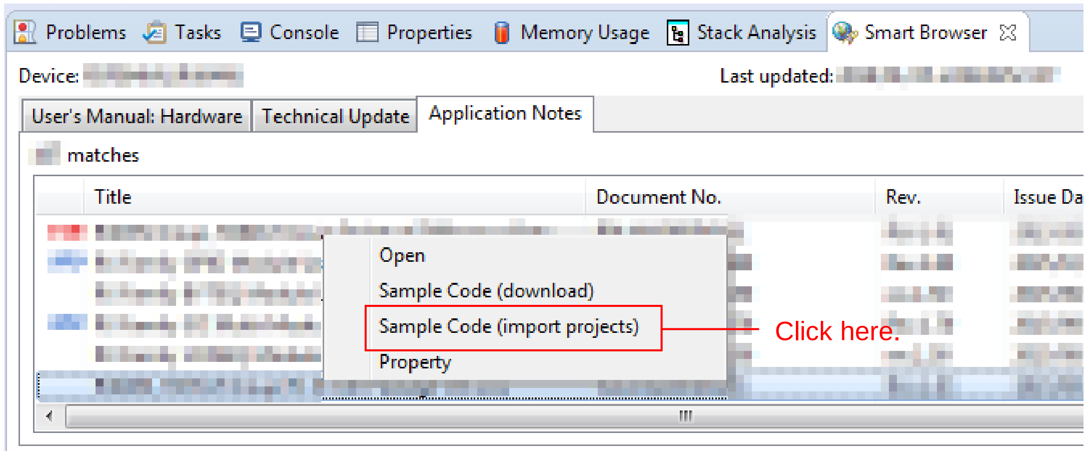
3. Click on the [Application Note] on the [Smart browser] tabbed page.



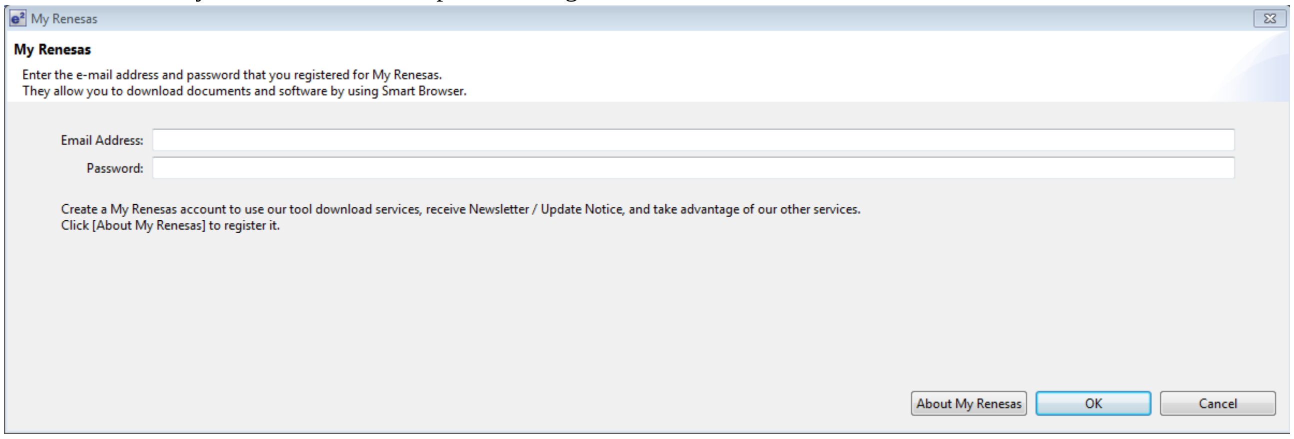
4. Click [Update].



5. Select the application note and right-click. Then, click on [Sample code(Project import)] in the context menu. (Note 1).



Note 1: If authentication by My Renesas has never been performed, ”My Renesas” dialog opens when downloading the file. Enter your mail address and password registered in the Renesas website.


My Renesas

My Renesas

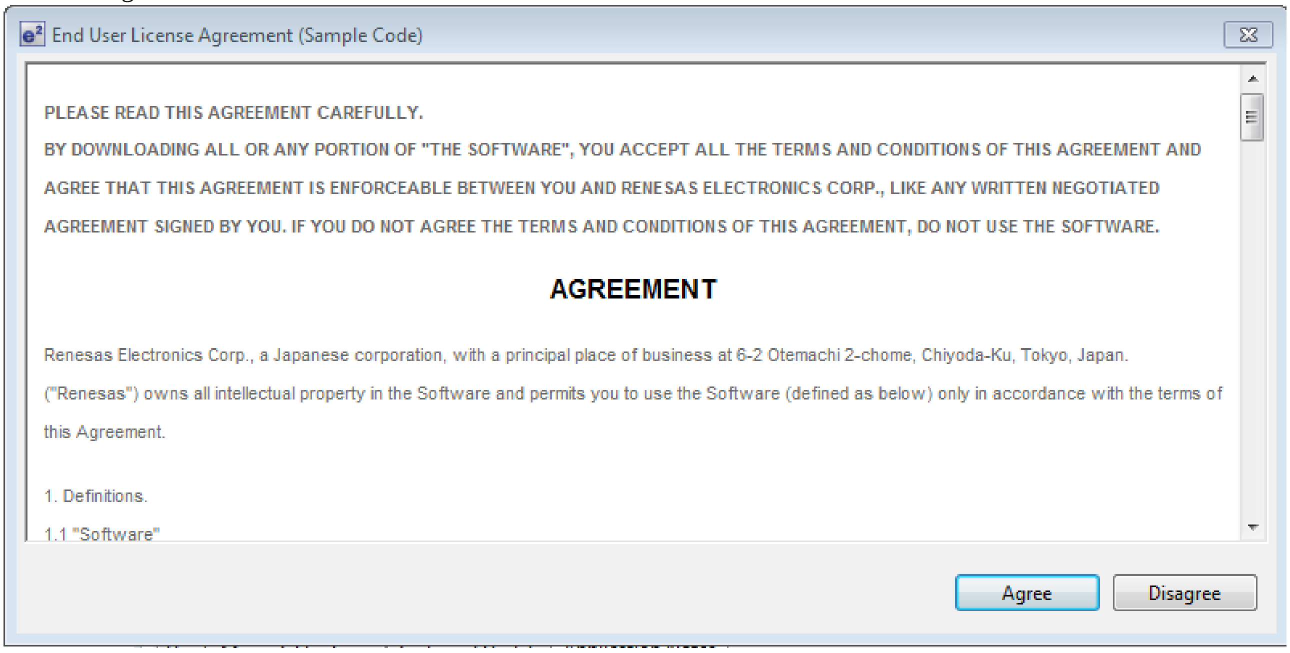
Enter the e-mail address and password that you registered for My Renesas.
They allow you to download documents and software by using Smart Browser.

Email Address:

Password:

Create a My Renesas account to use our tool download services, receive Newsletter / Update Notice, and take advantage of our other services.
Click [About My Renesas] to register it.

6. Click [Agree].



End User License Agreement (Sample Code)

PLEASE READ THIS AGREEMENT CAREFULLY.

BY DOWNLOADING ALL OR ANY PORTION OF "THE SOFTWARE", YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT AND AGREE THAT THIS AGREEMENT IS ENFORCEABLE BETWEEN YOU AND RENESAS ELECTRONICS CORP., LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. IF YOU DO NOT AGREE THE TERMS AND CONDITIONS OF THIS AGREEMENT, DO NOT USE THE SOFTWARE.

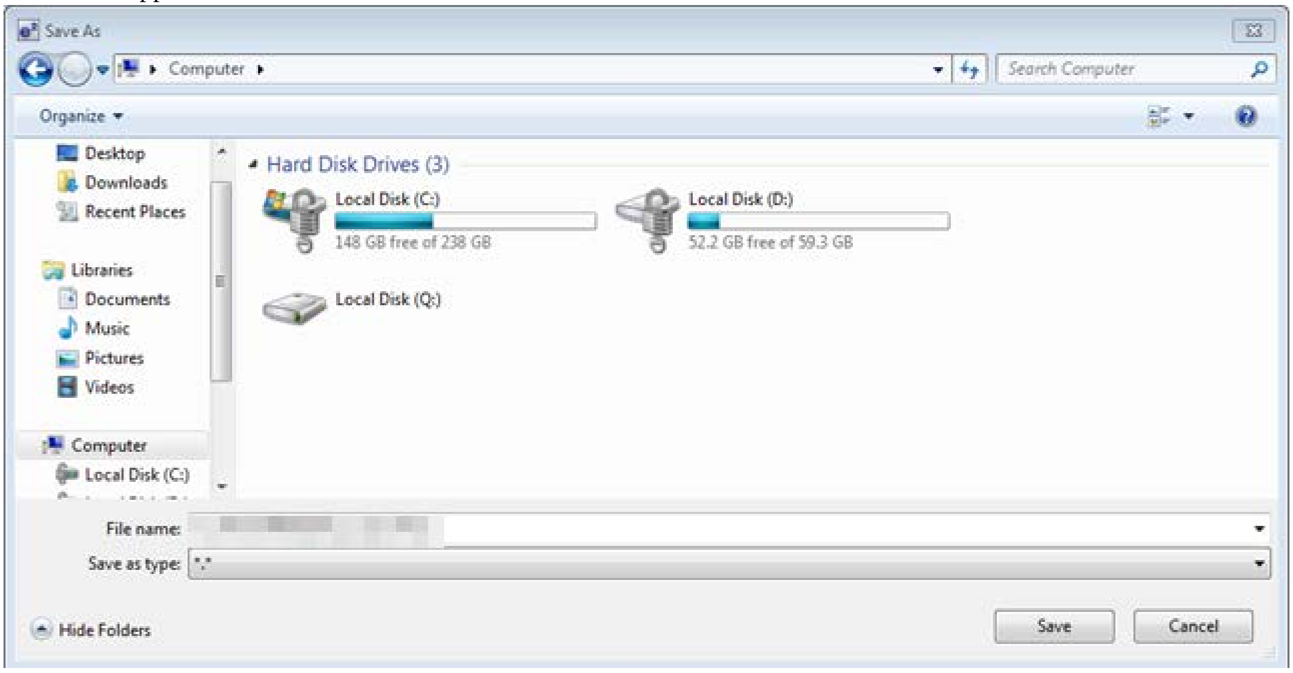
AGREEMENT

Renesas Electronics Corp., a Japanese corporation, with a principal place of business at 6-2 Otemachi 2-chome, Chiyoda-Ku, Tokyo, Japan. ("Renesas") owns all intellectual property in the Software and permits you to use the Software (defined as below) only in accordance with the terms of this Agreement.

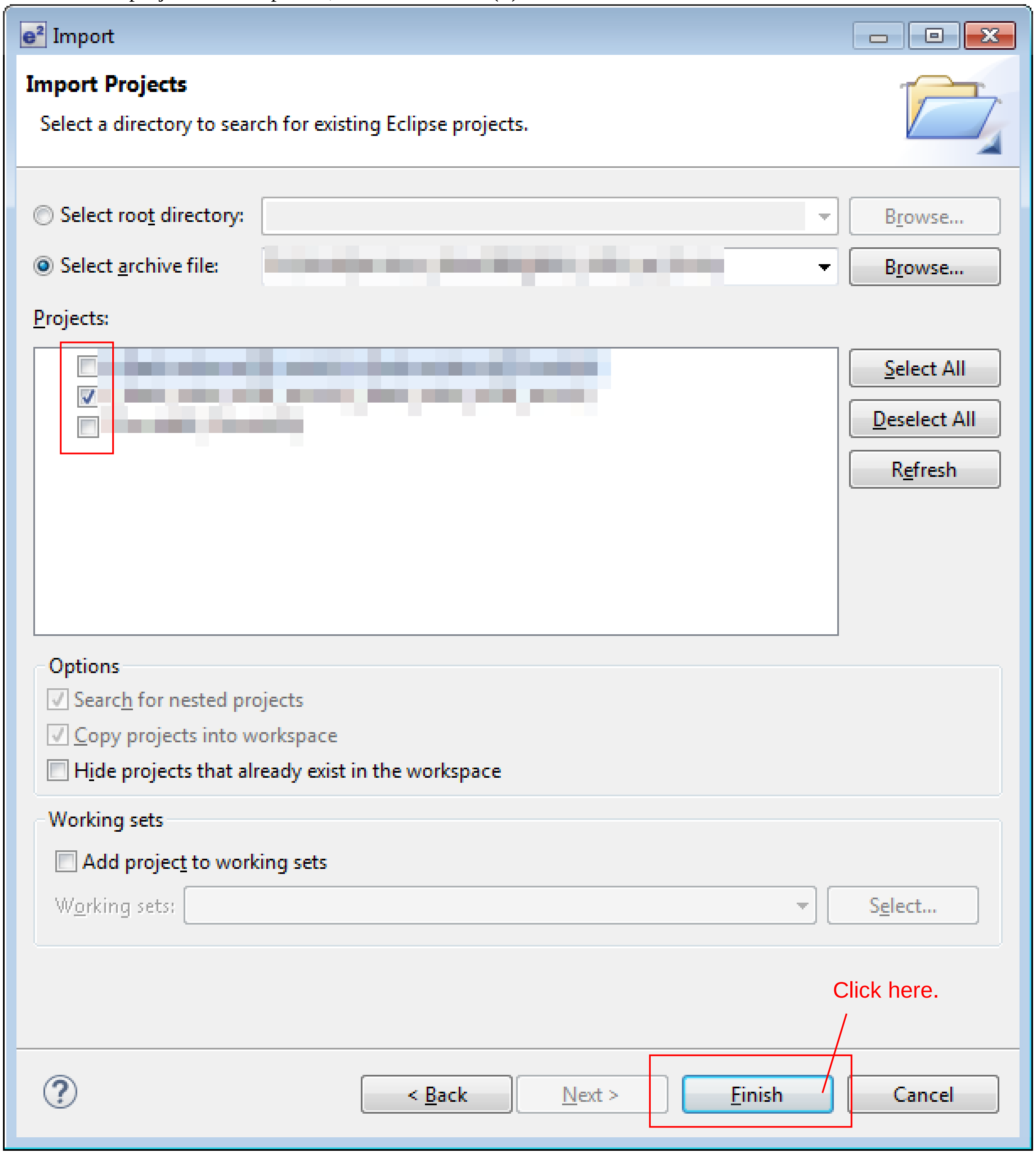
1. Definitions.

1.1 "Software"

7. Save the application note.



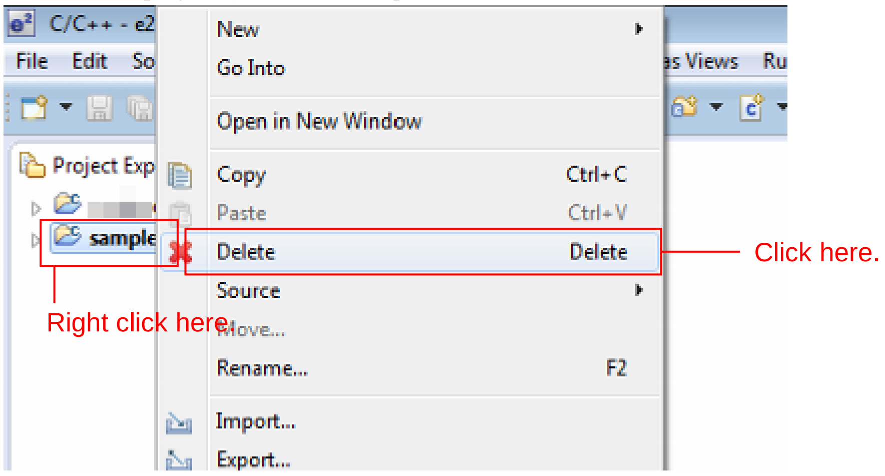
8. Select the project to be imported, and click [Finish(F)].



This application note includes the following projects.

Project name	Contents
r_flash_writer_rx113_switch	Project that selects a file by switch

9. Delete the project (shown as “sample” here) created to use the Smart browser as this is not required.



3.4 Modify Configuration

In this project, the configuration file setting and project setting for each FIT module are changed to configure the application. The detail is shown as follows.

Refer to this information when building new project. To use the project imported, go to “4. Verify Operation”.

3.4.1 Change Configuration

The configuration files for each FIT module configuring this application require modification.

Refer to the manuals and other files in the **doc** folder for each FIT module for details on the items and the settings in the configuration files.

The places to be changed in the configuration files are shown below.

(1) Change the number of drives for USB Mini

Change the number of drives for USB Mini defined by r_tfat_driver_rx configuration file as follows.

【r_config/r_tfat_driver_rx_config.h】

```
/* Number of logical drives to be used.
   Setting to 0    : unused memory
             other : number of logical drives
   (USB and SDHI can be used together.)
*/
#define TFAT_USB_DRIVE_NUM       (0)
#define TFAT_SDHI_DRIVE_NUM      (0)
#define TFAT_USB_MINI_DRIVE_NUM   (1)
```

(2) Change the device allocation

Allocate the device to the drive number. In this sample, drive 0 is allocated to USB Mini.

【r_config/r_tfat_driver_rx_config.h】

```
#define TFAT_DRIVE_ALLOC_NUM_0   TFAT_CTRL_USB_MINI
#define TFAT_DRIVE_ALLOC_NUM_1   NULL
#define TFAT_DRIVE_ALLOC_NUM_2   NULL
#define TFAT_DRIVE_ALLOC_NUM_3   NULL
#define TFAT_DRIVE_ALLOC_NUM_4   NULL
#define TFAT_DRIVE_ALLOC_NUM_5   NULL
#define TFAT_DRIVE_ALLOC_NUM_6   NULL
#define TFAT_DRIVE_ALLOC_NUM_7   NULL
```

(3) Change DTC transfer setting

The following DTC definition is described in r_usb_basic_mini_config.h.

Enable the ”USB_NOUSE” definition, as DTC transfer is not performed in the sample.

【r_config/r_usb_basic_mini_config.h】

```
 /* DTC DEFINE */
   #define DTC_USE_PIPE_NUM    USB_NOUSE
   //#define DTC_USE_PIPE_NUM    USB_PIPE1
   //#define DTC_USE_PIPE_NUM    USB_PIPE2
   //#define DTC_USE_PIPE_NUM    USB_PIPE3
   //#define DTC_USE_PIPE_NUM    USB_PIPE4
   //#define DTC_USE_PIPE_NUM    USB_PIPE5
```

(4) Change TFAT setting

TFAT definition is described in r_usb_hmsc_mini_config.h. To use TFAT, enable the following macro.

【r_config/r_usb_hmsc_mini_config.h】

```
     #define USB_TFAT_USE_PP
```

(5) Change Flash API setting

To program code flash, enable the following macro.

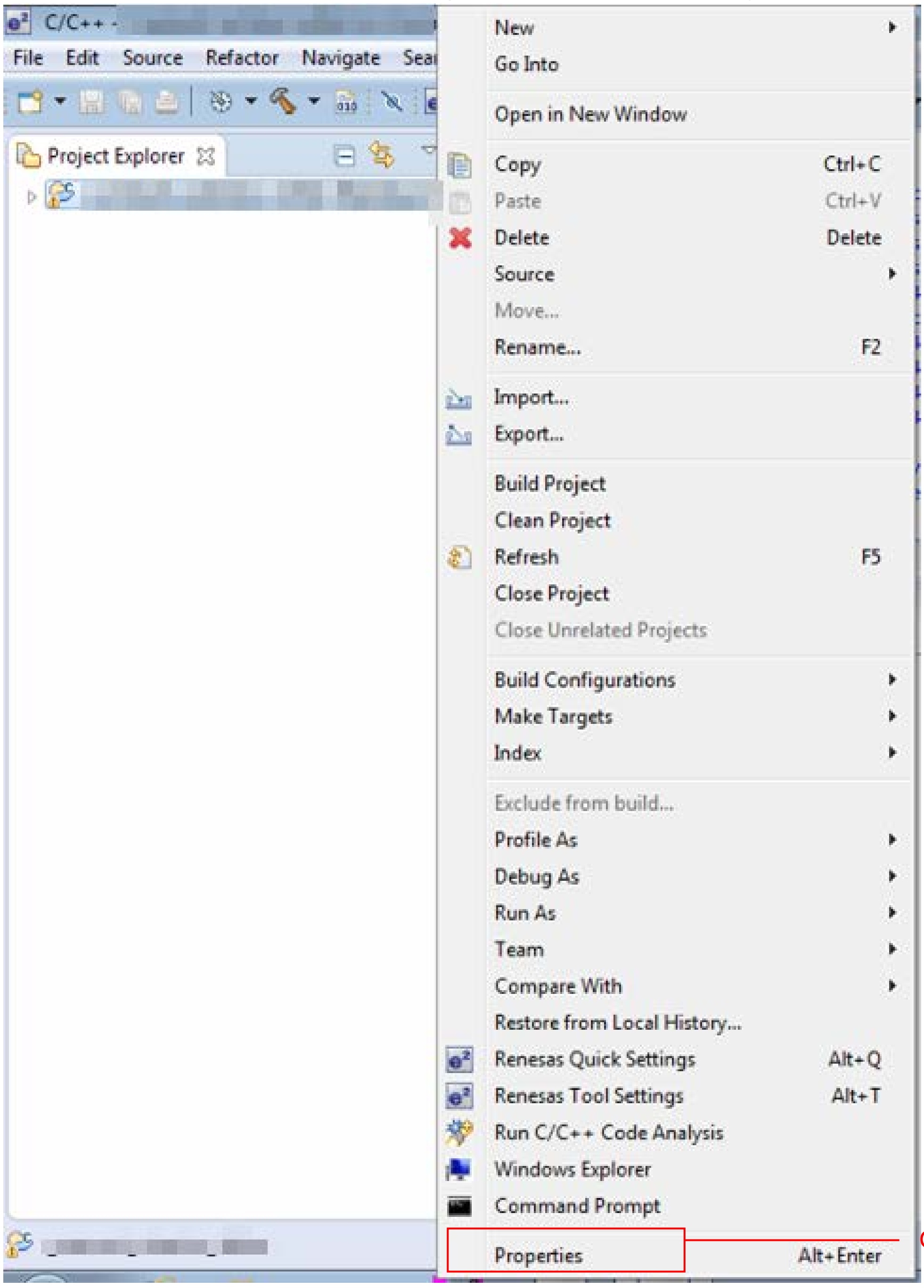
【r_config/r_flash_rx_config.h】

```
      #define FLASH_CFG_CODE_FLASH_ENABLE (1)
```

3.4.2 Change Project Setting

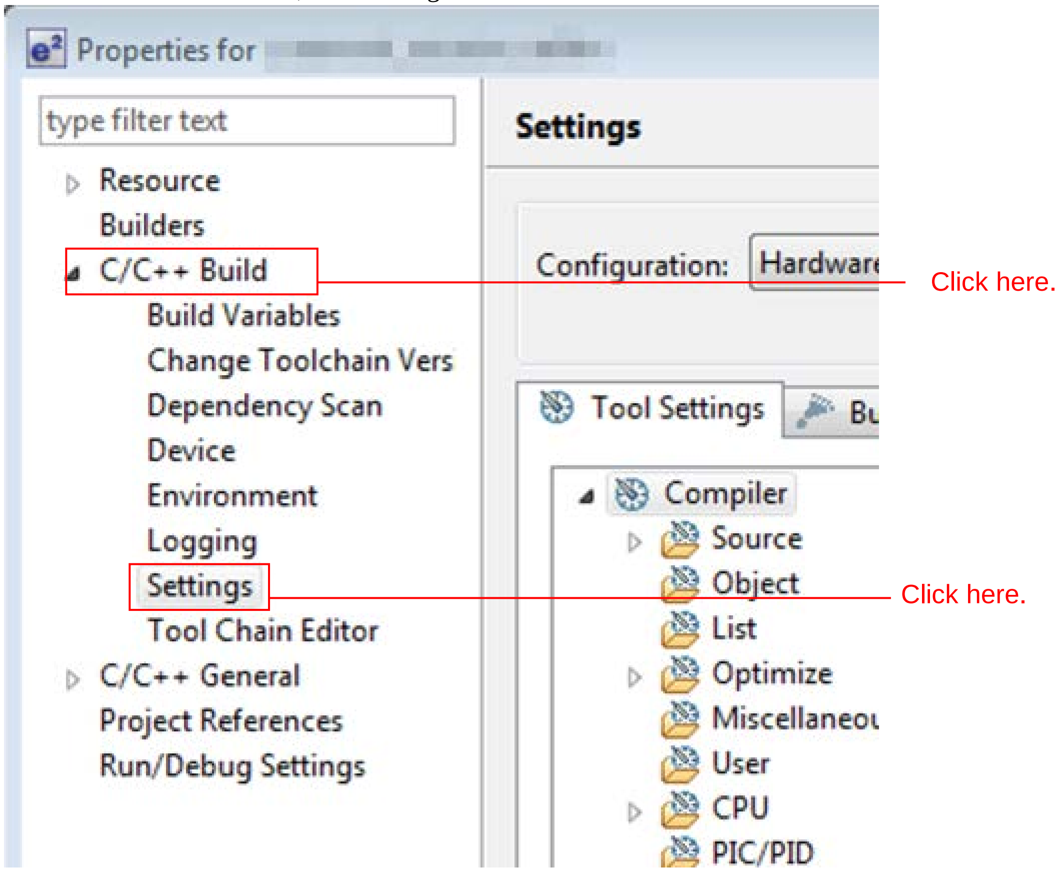
The contents changed from default setting of the project setting is shown. To check the project setting, use the following procedure.

1. Select the project for the e² studio and right-click. Then, click [Property (R)].



Click here.

2. Click on [C/C++ Build], and [Setting].



— Project setting of the main program

The main program setting is changed from default setting to the contents listed in Table 3-1 Changed build setting Modules for building, and in Table 3-2 Changed debug setting for debugging

Table 3-1 Changed build setting Modules

Items	Changed contents	Description
Compiler - Object	Check “Generate debug information”	Outputs the debug information required when debugging.
Assembler - Object	Check “Generate debug information”	Outputs the debug information to a relocatable file.
Linker - Input	Add "${workspace_loc:/${ProjName}/r_tfat_rx/lib/tfat_rx200_little.lib}" (Note)	Requires the setting when using TFAT. (required when using TFAT)
Linker - Section	Remove PResetPRG and PIntPRG from section definition (Note)	Requires the setting when using BSP (required when using FIT)
	Change P Section to P* Section (Note)	Requires the setting when using BSP (required when using FIT)
	Add RPFRAM Section after R Section (Note)	Requires the setting of the area Flash API uses (required when using Flash API)
Linker - Output	Map from ROM to RAM Add PFRAM=RPFRAM to the section (Note)	Requires the setting of the area Flash API uses (required when using Flash API)

Note The setting change is required when creating the project that includes each FIT module for BSP, TFAT, and Flash API. For the setting, refer to the manuals, etc. in the **doc** folder of each FIT module.

Table 3-2 Changed debug setting

Items	Changed contents	Description
Debugger - Debug tool setting	Change “Re-write the on-chip program ROM” to “Yes”	Required when debugging the program re-writing on-chip flash memory.

— Project setting of the sample program

The changed contents from default setting when building is listed in .

Table 3-3 Changed build setting (sample)

Items	Changed contents	Description
Linker - Section	Remove PResetPRG and PIntPRG from section definition (Note)	Requires the setting when using BSP (required when using FIT)
	Change P Section to P* Section (Note)	Requires the setting when using BSP (required when using FIT)
	Change the address of C_1 section to “0xFFFF 0000”	Define start address to program
	Change the address of FIXEDVECT section to “0xFFFF BF80”	Define start address of fixed vector table
Linker - Output	Change output file/type to “Binary via absolute”	Set the file type to write in the USB memory

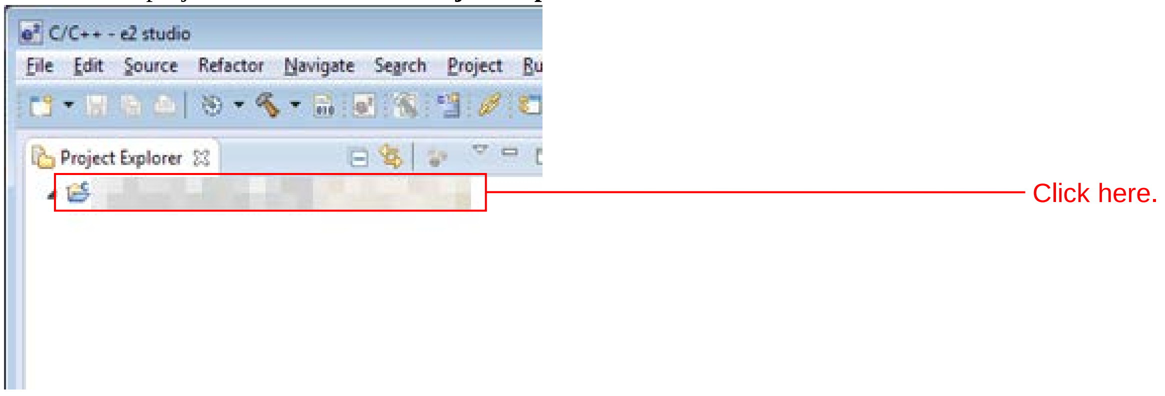
Note The setting change is required when creating the project that includes each BSP FIT module. For the setting, refer to the manuals, etc. in the **doc** folder of each BSP FIT module.

4. Verify Operation

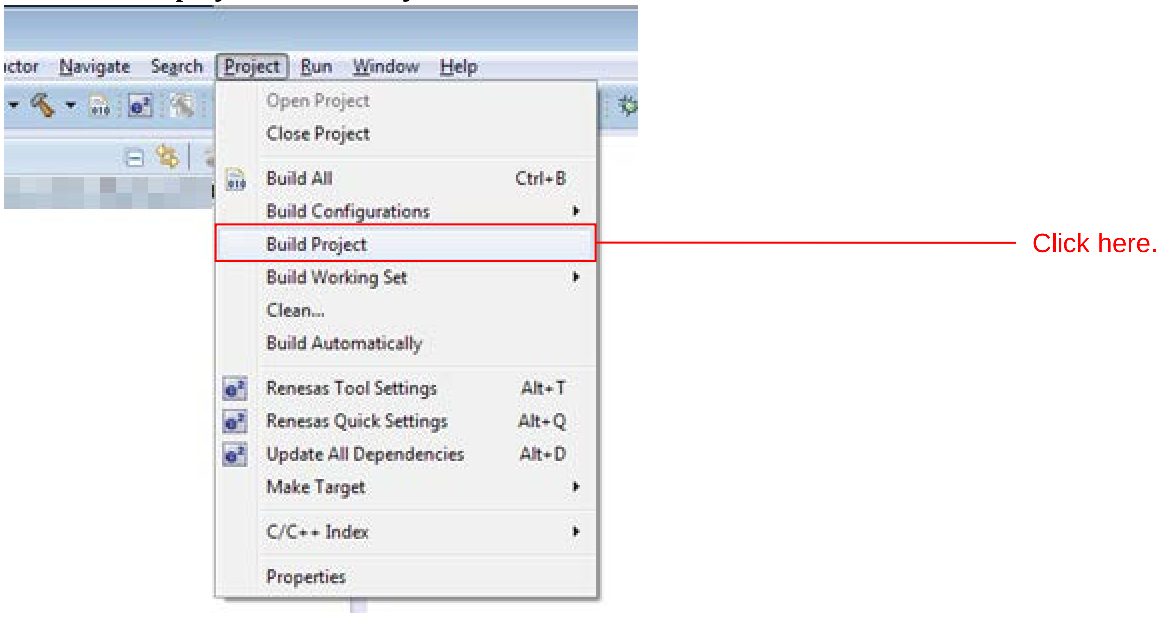
4.1 Build the Project

Use the following procedure to build the project and generate a load module.

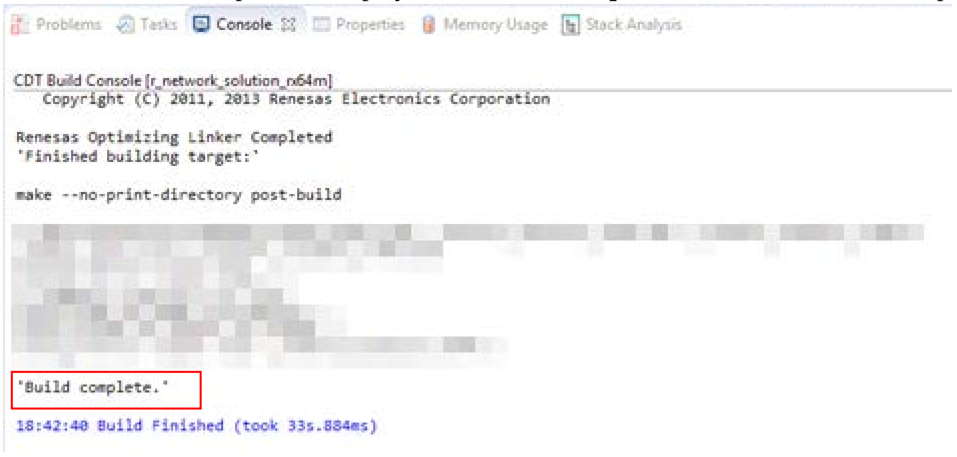
1. Click the project to build from the **Project Explorer**.



2. Click **Build project** from the **Project** menu.



3. When “Build complete” is displayed on the **Console panel**, the build will have completed.



```
CDT Build Console [r_network_solution_rx64m]
   Copyright (C) 2011, 2013 Renesas Electronics Corporation

Renesas Optimizing Linker Completed
'Finished building target:'

make --no-print-directory post-build

[redacted]

'Build complete.'

18:42:40 Build Finished (took 33s.884ms)
```

4.2 Prepare for Debugging

4.2.1 Configure Hardware

The evaluation board must be configured before starting debugging.

A table of the required equipment and its configuration are shown below.

Table 4-1 Hardware Configuration

Device	Supplementary Information
Development PC	
RSK	Evaluation board
E1 Emulator	Included in Renesas Starter Kit for RX113
USB memory	Memory that is formatted as either FAT or FAT32.

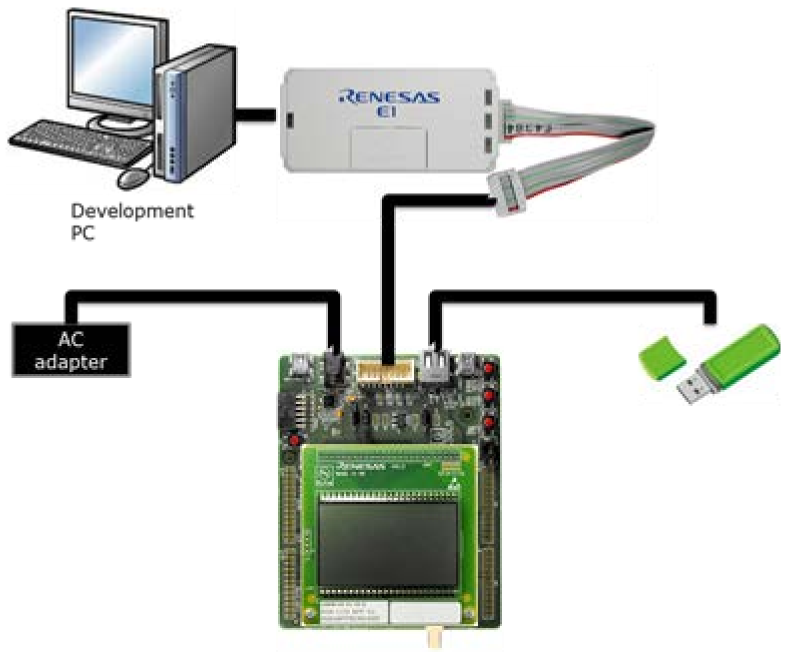


Figure 4-1 Operating environment example

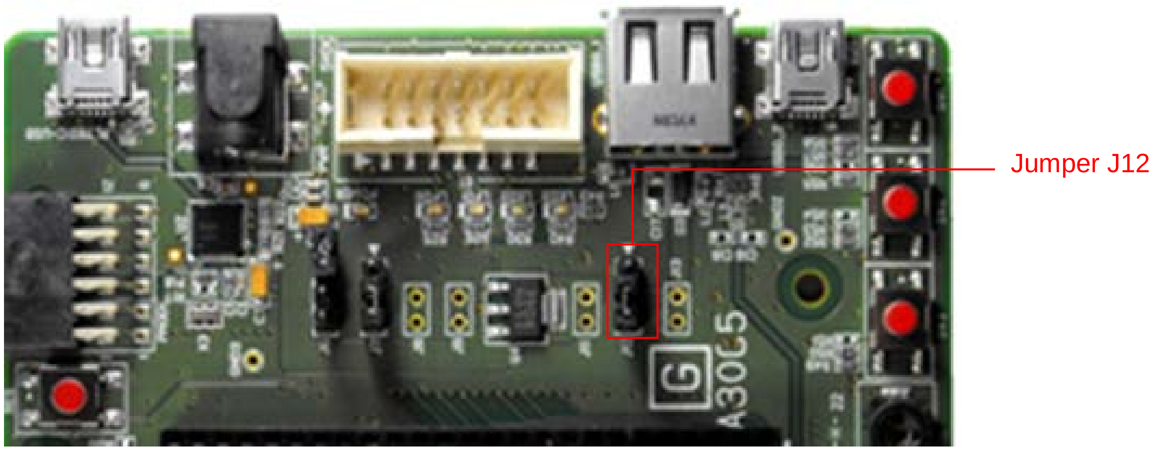
4.2.2 Set up the RSK

The RSK settings required to operate the main program are shown below.

Set the USB mode (Host/Peripheral). Set jumper J12 to match the setting of USB_FUNCSEL_PP in r_usb_basic_mini_config.h.

Table 4-2 Jumper Settings

Devices	Jumper	Setting contents
When use USB in host mode. (USB_FUNCSEL_PP = USB_HOST_PP)	J12	Short 1 to 2. (*selected this time)
When use USB in peripheral mode. (USB_FUNCSEL_PP = USB_PERI_PP)	J12	Short 2 to 3.



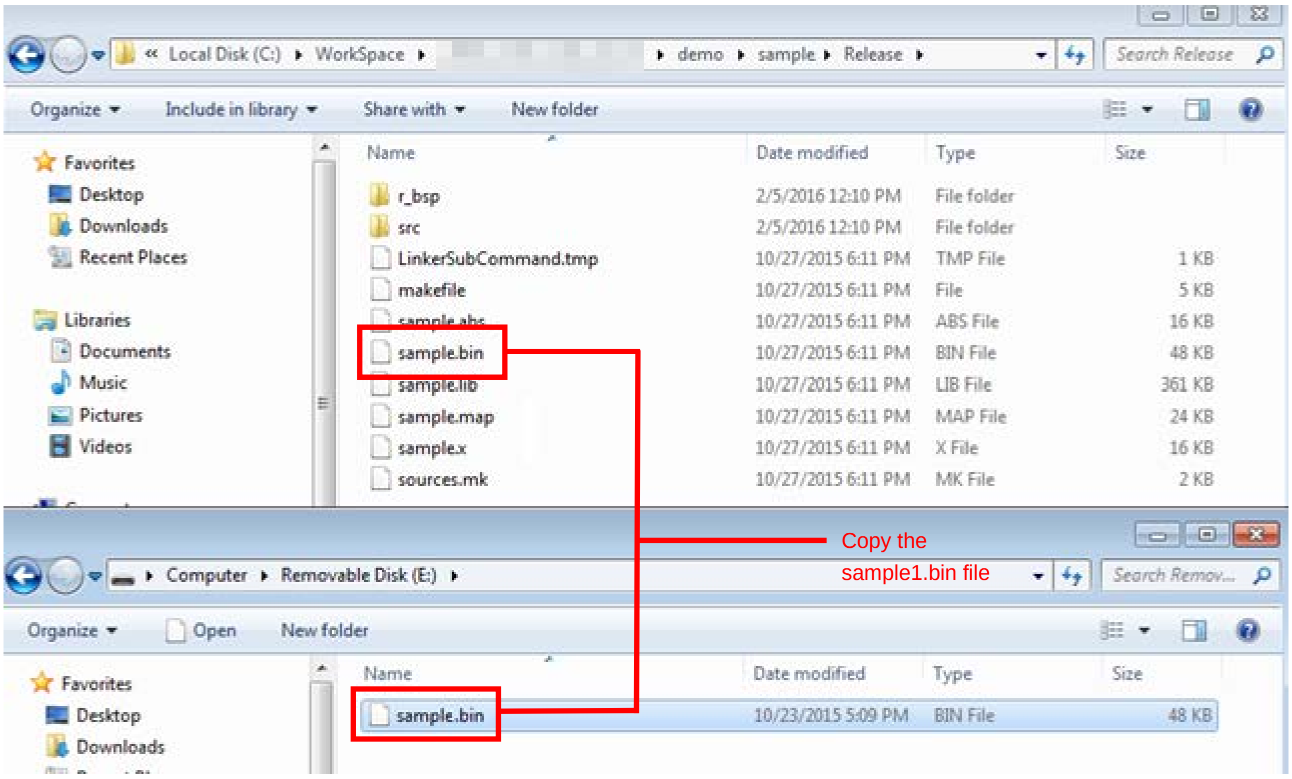
Note: The image and the actual configuration will be different.

Figure 4-2 RSK Jumper Locations

4.2.3 Prepare USB Memory

Store the binary file of the sample program on the USB memory.

Open the **demo** folder in the project of the main program, and decompress the **sample1.zip** file and save it into the desired location (folder). Copy the **sample.bin** file in the decompressed **sample/release** folder to the USB memory.



Name	Date modified	Type	Size
r_bsp	2/5/2016 12:10 PM	File folder	
src	2/5/2016 12:10 PM	File folder	
LinkerSubCommand.tmp	10/27/2015 6:11 PM	TMP File	1 KB
makefile	10/27/2015 6:11 PM	File	5 KB
sample.abs	10/27/2015 6:11 PM	ABS File	16 KB
sample.bin	10/27/2015 6:11 PM	BIN File	48 KB
sample.lib	10/27/2015 6:11 PM	LIB File	361 KB
sample.map	10/27/2015 6:11 PM	MAP File	24 KB
sample.x	10/27/2015 6:11 PM	X File	16 KB
sources.mk	10/27/2015 6:11 PM	MK File	2 KB

Name	Date modified	Type	Size
sample.bin	10/23/2015 5:09 PM	BIN File	48 KB

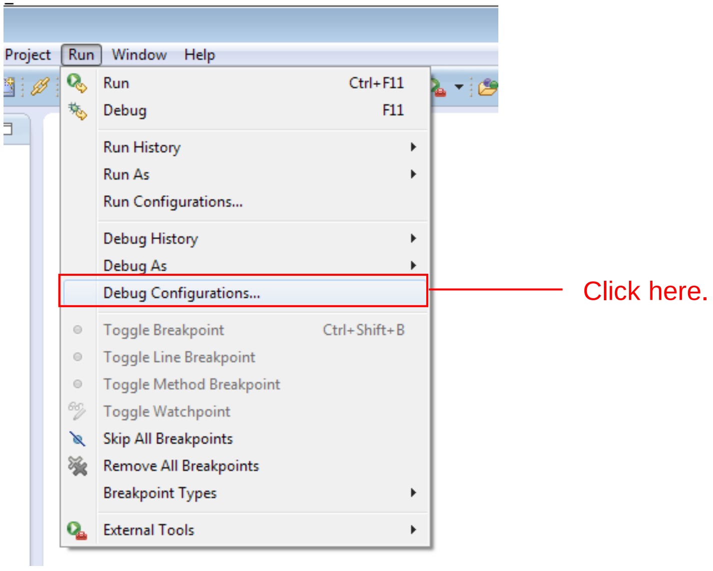
4.3 Debug the Project

Use the following procedure to start debugging the project.

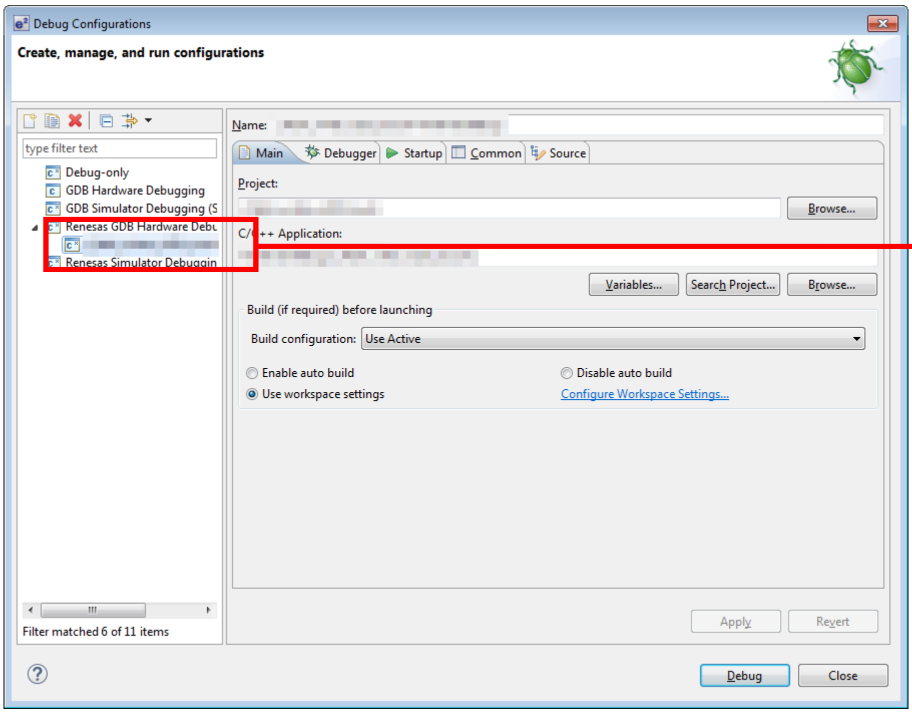
1. Connect the development PC to the E1 emulator with a USB cable, and connect E1 emulator to the RSK with user system interface cable.

2. Connect the RSK to the adapter and turn on the power.

3. Click on [Debug Configurations] in the e2 studio Run menu.



4. Click on [r_flash_writer_rx113.x] under [Renesas GDB Hardware Debugging].



Debug Configurations

Create, manage, and run configurations

type filter text

- Debug-only
- GDB Hardware Debugging
- GDB Simulator Debugging (S
- Renesas GDB Hardware Debu
  - [redacted]
- Renesas Simulator Debuggin

Filter matched 6 of 11 items

Name: [redacted]

Main | Debugger | Startup | Common | Source

Project: [redacted]

C/C++ Application: [redacted]

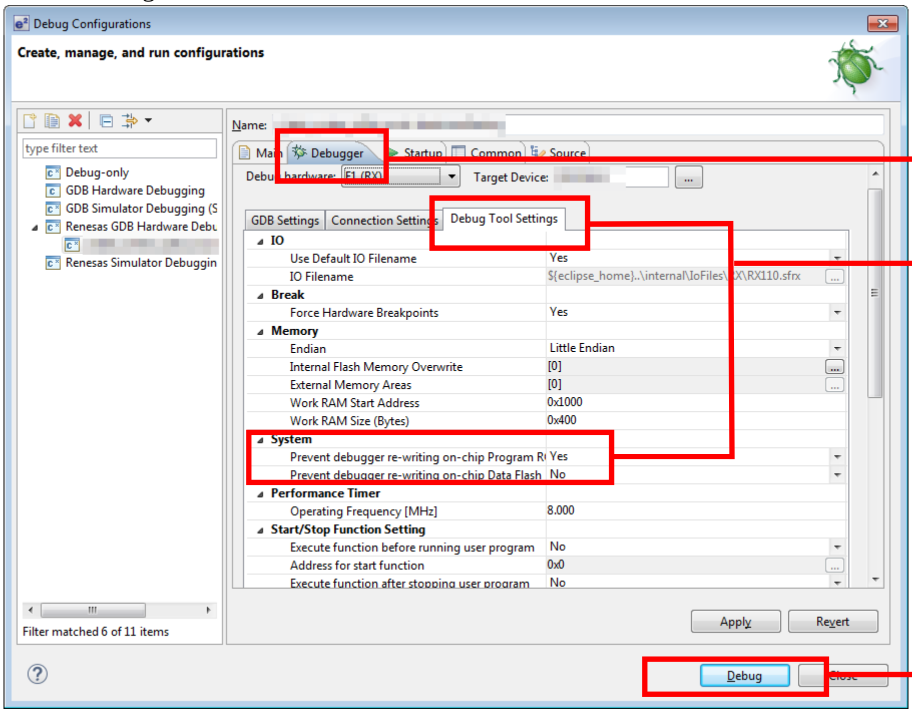
Build (if required) before launching

Build configuration: Use Active

- Enable auto build
- Disable auto build
- [✓ Use workspace settings]

Configure Workspace Settings...

5. Click on [Debug Tool setting] → [System] → [Re-write the on-chip program ROM] and select [Yes], then, click on [Debug].


Select [Debugger]
Select [Debug Tool setting]→ [System]→ [Re-write the on-chip program ROM] and select [Yes]
Click on [Debug]

Debug Configurations

Create, manage, and run configurations

Name: [redacted]

Main | Debugger | Startup | Common | Source

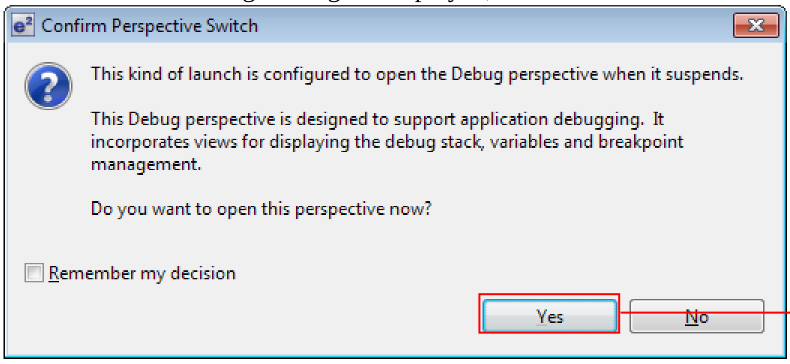
Debug hardware: E1 (RX) Target Device: [redacted]

GDB Settings | Connection Settings | Debug Tool Settings

IO	
Use Default IO Filename	Yes
IO Filename	${eclipse_home}..\internal\IoFiles\RX\RX110.sfrx
Break	
Force Hardware Breakpoints	Yes
Memory	
Endian	Little Endian
Internal Flash Memory Overwrite	[0]
External Memory Areas	[0]
Work RAM Start Address	0x1000
Work RAM Size (Bytes)	0x400
System	
Prevent debugger re-writing on-chip Program R	Yes
Prevent debugger re-writing on-chip Data Flash	No
Performance Timer	
Operating Frequency [MHz]	8.000
Start/Stop Function Setting	
Execute function before running user program	No
Address for start function	0x0
Execute function after stopping user program	No

Filter matched 6 of 11 items

6. When the following message is displayed, click [Yes].



Confirm Perspective Switch

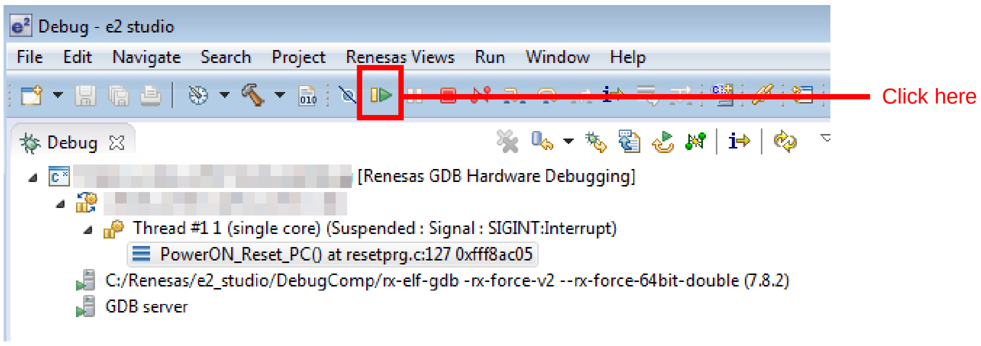
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

7. When the load module download completes, a Debug perspective opens.

8. Click [Resume] on the toolbar. The program will be executed and a break will occur at the start of the main function.



Debug - e2 studio

File Edit Navigate Search Project Renesas Views Run Window Help

Debug

[Renesas GDB Hardware Debugging]

Thread #1 1 (single core) (Suspended : Signal : SIGINT:Interrupt)

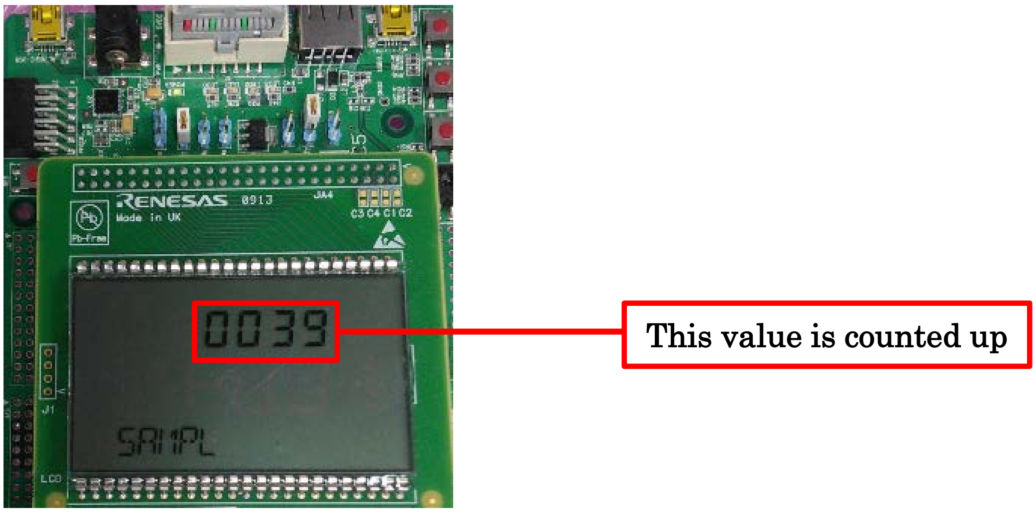
PowerON_Reset_PC() at resetprg.c:127 0xfff8ac05

C:/Renesas/e2_studio/DebugComp/rx-elf-gdb -rx-force-v2 --rx-force-64bit-double (7.8.2)

GDB server

9. After the break at the start of the main function, click [Resume] on the tool bar again.

   Connect the USB memory.
   Data will be read from the USB memory automatically and written to the flash.
   If the MCU is restarted and the value displayed on the LCD is counted up as shown in the image below, then it means that the program has completed successfully.



5. Application overview

The sample program counts up the value displayed on the 7 segment display of the RSK RX113 LCD every 1 second. It only requires the on-board memory of the MCU for its operation and no other memory is required.

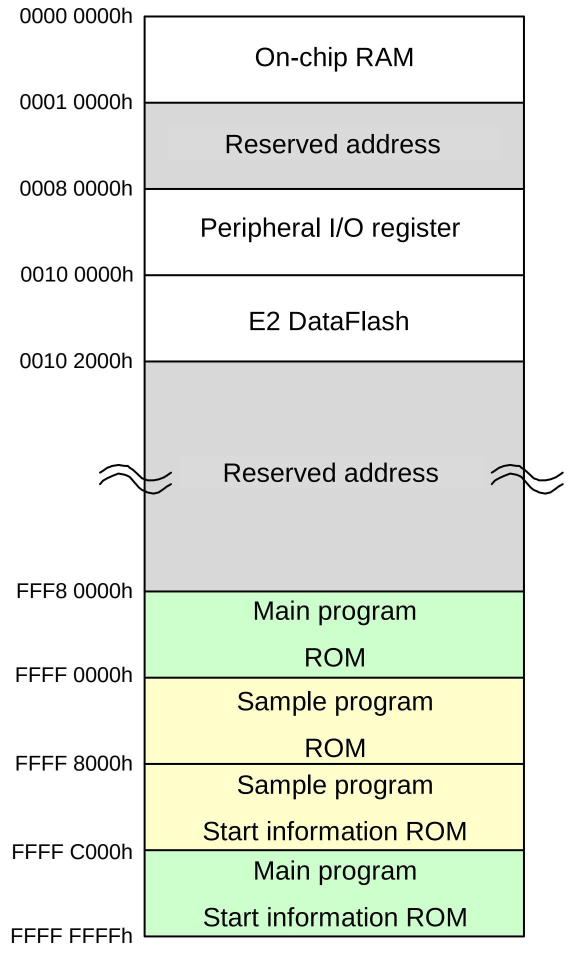
The memory settings are configured to suit the ROM and RAM capacities of RSK RX113 in the sample program.

1. Initializes the LCD driver.
2. Displays “SAMPL” on the display part of the LCD.
3. Performs 1-second wait processing.
4. Displays a counted-up value (0000 to 9999) on the 7 segment display of the LCD.
5. Repeats step 3 and 4.

Note: This sample program has been created for evaluating the main program operation. Therefore the program operates on memory areas that are not used by the main program.

5.1 Memory structure

It shows RX113 MCU memory map on the RSKRX113 used in this application.



Note 1 : The memory structure is for RX113 MCU in the RSKRX113.
Section address is different depending on the CPU model and main program size used.
Also, modification is required depending on the user system.

Figure 5-1 Memory map

6. Main Program Specifications

6.1 Files

The main program is included in the program. The source files of the main program is included in the **src** folder.

The main program FIT module name is "r_flash_writer_rx113". The source files are included in **src** folder.

The file structure of the FIT modules and the main program files are listed below.

The main program files are listed in Table 6-1 , and the FIT modules used are listed in Table 6-2 .

Table 6-1 Main Program Files

Folder	File	Description
src	main.c	Source file of main processing
	r_rsk_extention_lcd.c	Source file of LCD driver application processing
	r_rsk_flashdriver.c	Source file of flash processing
	r_rsk_keydriver.c	Source file of key input processing
	r_rsk_keydriver.h	Header file of key input processing
	r_rsk_lcddriver.c	Source file of LCD driver call processing
	r_rsk_lcddriver.h	Header file of LCD driver call processing
	r_rsk_leddriver.c	Source file of the LED initialization
	r_rsk_leddriver.h	Header file of the LED initialization
	r_usb_hmsc_apl.c	Source file of total control processing
	r_usb_hmsc_apl.h	Header file of USB driver call processing

Table 6-2 FIT modules used

Folder name	Contents
r_bsp	Board Support Package (BSP) file group
r_config	FIT module config file
r_flash_rx	Flash memory (Flash API) file group
r_lcdc_rx	LCD Controller/Driver (LCDC) file group
r_tfat_driver_rx	M3S-TFAT-Tiny Memory driver interface file group
r_tfat_rx	M3S-TFAT-Tiny FAT file system (TFAT) file group
r_usb_basic_mini	USB Basic Firmware file group
r_usb_hmsc_mini	USB Host Mass Storage Class (USB HMSC) file group

6.2 Modules

The following table lists the modules.

Table 6-3 Modules

File	Module	Description
main.c	main	Main processing in the main program Processing to open the flash driver and call for the usb_main function.
r_usb_hmsc_api.c	usb_main	Initializations and total control processing
	usb_hmsc_driver	HMSC driver task processing
	msc_detach_device	Processing for disconnecting the USB
	msc_connect_wait	Wait processing for connecting the USB device
	msc_drive	Processing for connecting the USB device and mounting the TFAT file system
	msc_data_ready	Processing for dummy state before read state
	msc_data_read	Processing for reading from the USB memory and writing to the flash
	usb_hsmpl_device_state	USB driver callback processing
	msc_configured	Processing for notifying of USB connection
	msc_drive_complete	Processing for notifying of completion of USB connection
	msc_detach	Processing for notifying of USB disconnection
	usb_mcu_init	USB port initialization
	usb_board_init	LED and LCD initializations and processing for enabling USB interrupts.
	usb_driver_init	USB driver initialization
	apl_init	Control table initialization
	msc_registration	Processing for registering the USB callback function
	msc_event_set	Event setting processing
	msc_event_get	Processing for detecting a switch input and obtaining an event
	msc_led_control	LED control processing
r_rsk_leddriver.c	usb_cpu_LedInitial	LED initialization
	usb_cpu_led_set_bit	LED individual control processing
	usb_cpu_led_set_data	LED whole control processing
r_rsk_extention_lcd.c	Init_Extension_LCD	LCD port initialization
	usb_lcd_print_14seg_string	Processing for turning on 14 segments
	usb_lcd_print_u7seg_digit	Processing for displaying on 7 segments (upper part)
	usb_lcd_print_c7seg_digit	Processing for displaying on 7 segments (middle part)
	dfw_lcd_ascii_calc	Processing for ASCII character conversion
r_rsk_lcddriver.c	usb_cpu_LcdInitial	LCD module initialization
	usb_cpu_LcdDisp	LCD display processing
	string_length_count	Processing for the number of strings information
r_rsk_flashdriver.c	SAMPLE_FLASH_Write	Flash erase/program processing
r_rsk_keydriver.c	usb_cpu_key_read	Chattering elimination processing
	usb_cpu_sw_data	Processing for detecting a switch input
	usb_cpu_sw1_data	Switch 1 input
	usb_cpu_sw2_data	Switch 2 input
	usb_cpu_sw3_data	Switch 3 input

6.3 Flowcharts

(1) Main processing

This is the main function which is firstly called from the startup routine of the board support package (BSP).

The function performs flash open processing and calls the usb_main function.

Although an infinite loop is performed in the usb_main function, an infinite loop is also performed in the end of this function.

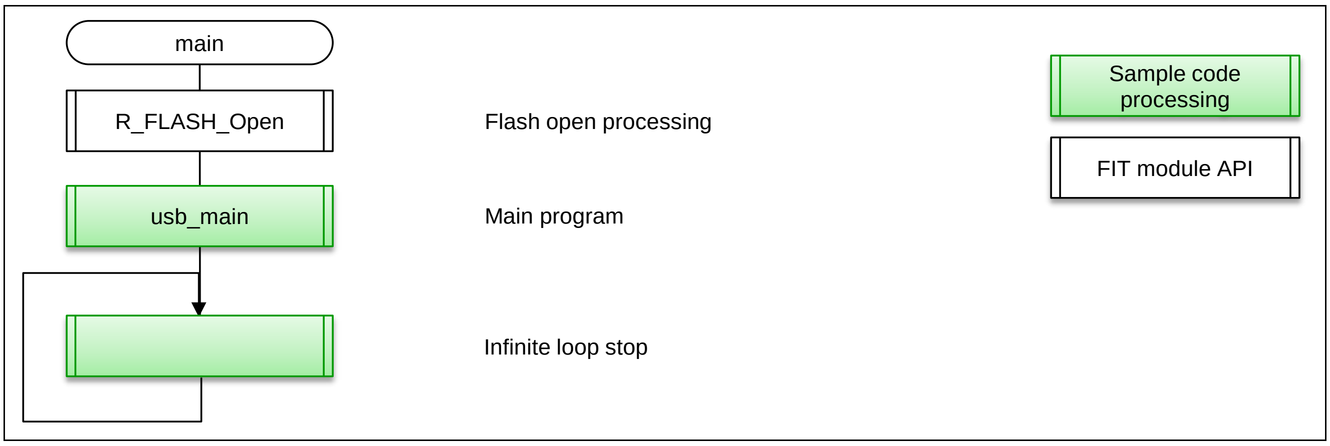


Figure 6-1 Main processing

(2) Main processing of USB

This is main processing which performs various initializations and controls states of processing order.

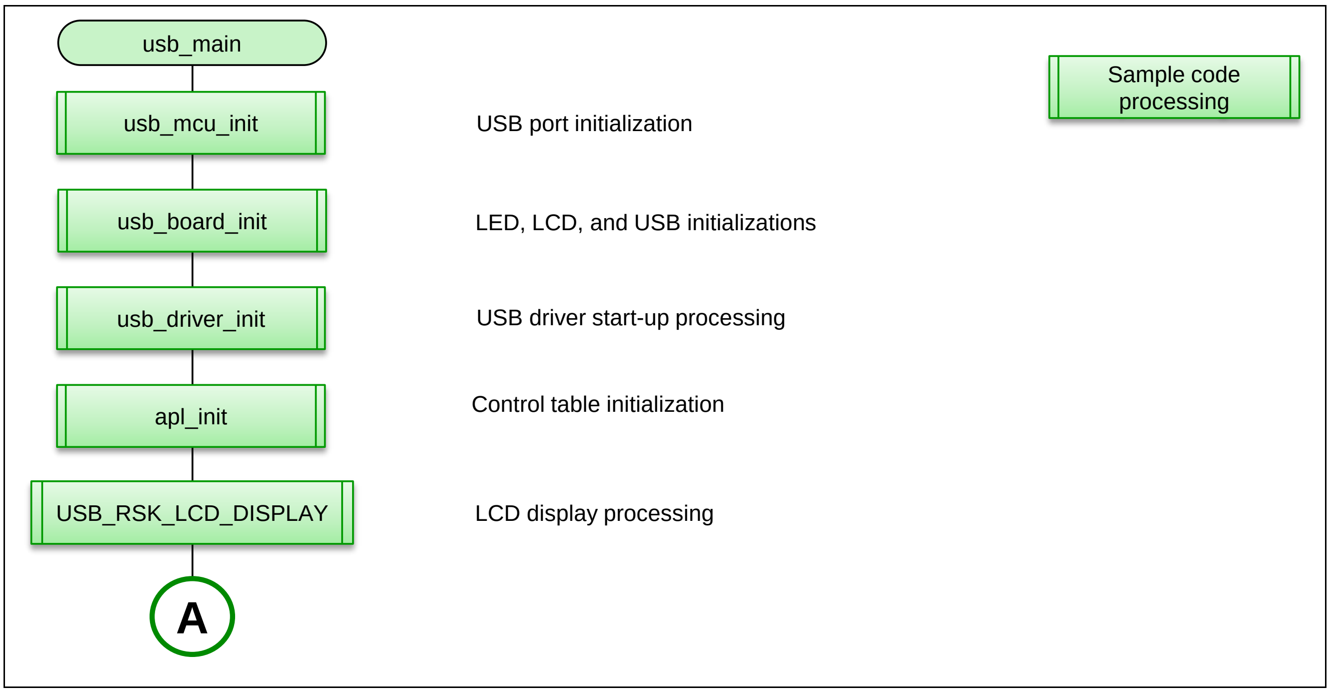


Figure 6-2 Main processing of USB (1)

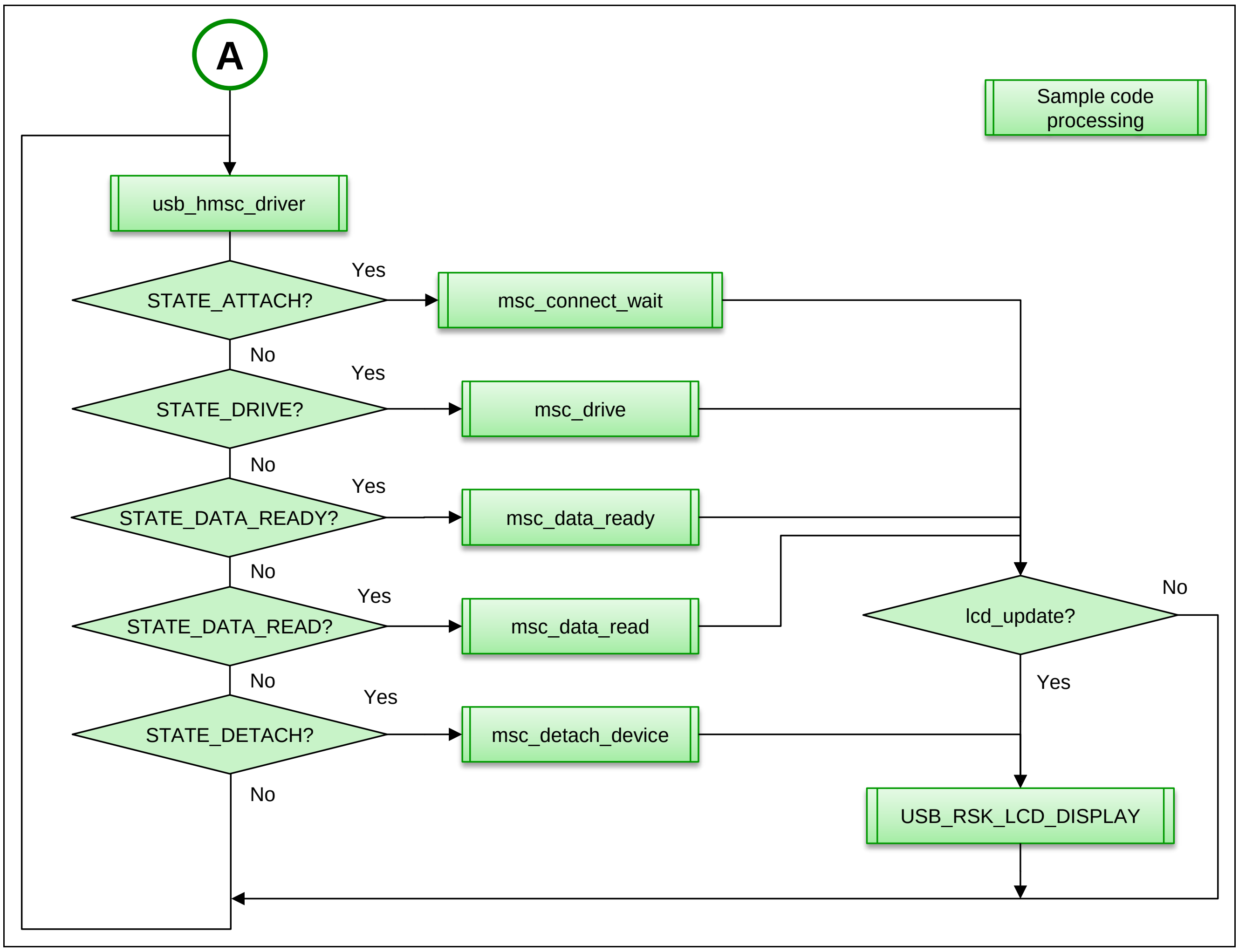


Figure 6-3 Main processing of USB (2)

(3) Port initialization of USB driver

This performs initial port setting for the USB driver.

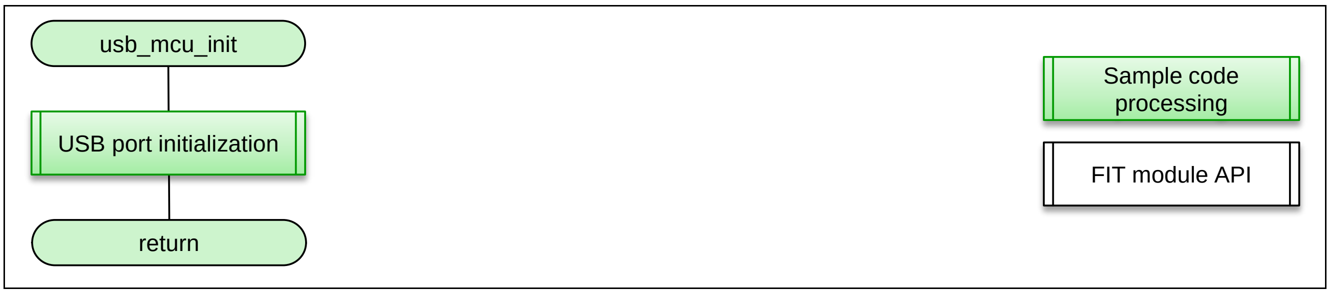


Figure 6-4 Port initialization of USB driver

(4) Initialization of LCD driver, LED driver and USB driver

This initializes the LCD, LED, and USB drivers.

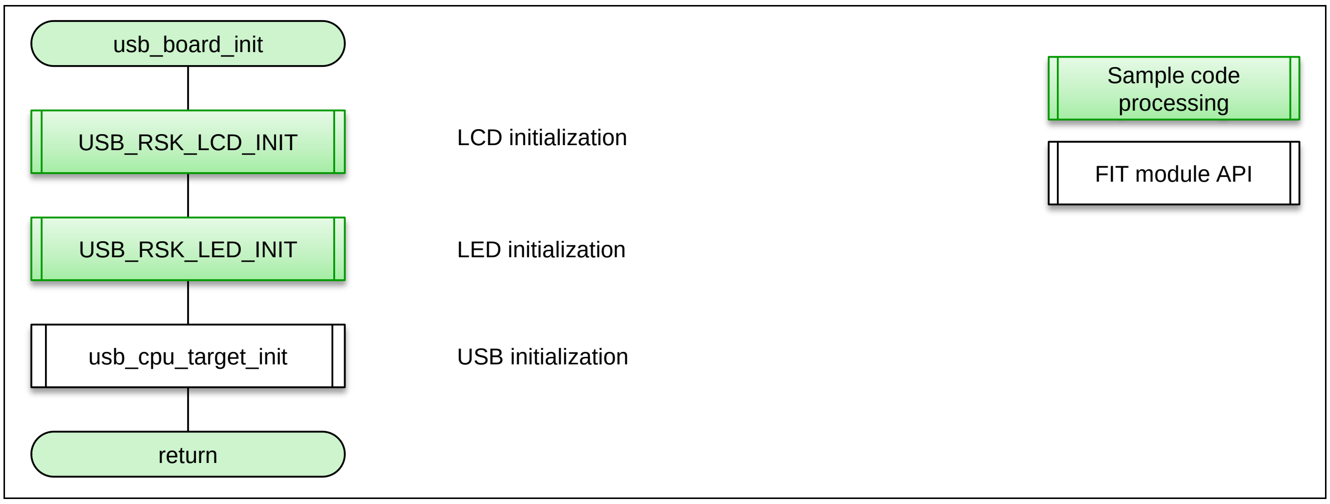


Figure 6-5 Initialization of LCD driver, LED driver and USB driver

(5) USB driver initialization

This is processing for starting up the USB driver.

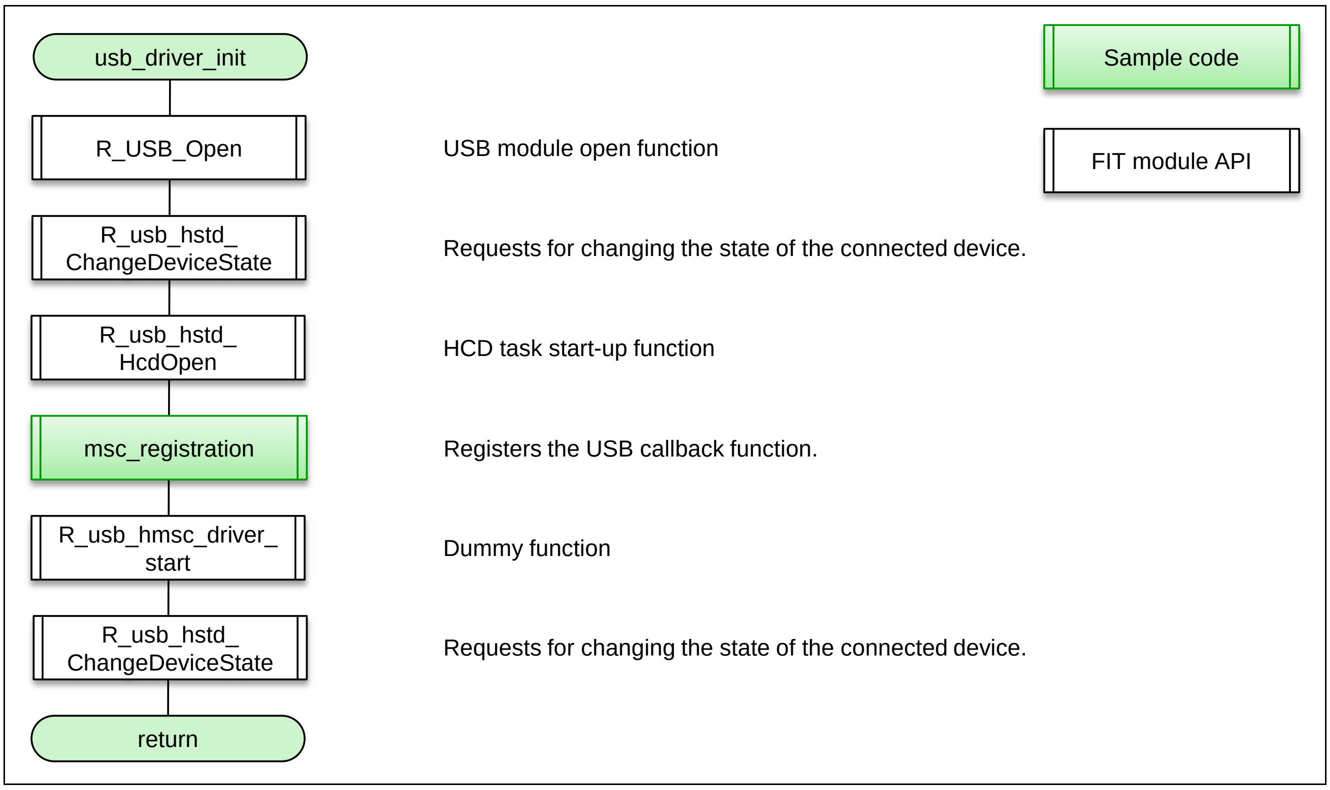

usb_driver_init
R_USB_Open
USB module open function
R_usb_hstd_ ChangeDeviceState
Requests for changing the state of the connected device.
R_usb_hstd_ HcdOpen
HCD task start-up function
msc_registration
Registers the USB callback function.
R_usb_hmsc_driver_ start
Dummy function
R_usb_hstd_ ChangeDeviceState
Requests for changing the state of the connected device.
return
Sample code
FIT module API

Figure 6-6 USB driver initialization

(6) Control table Initialization

This initializes the control table.

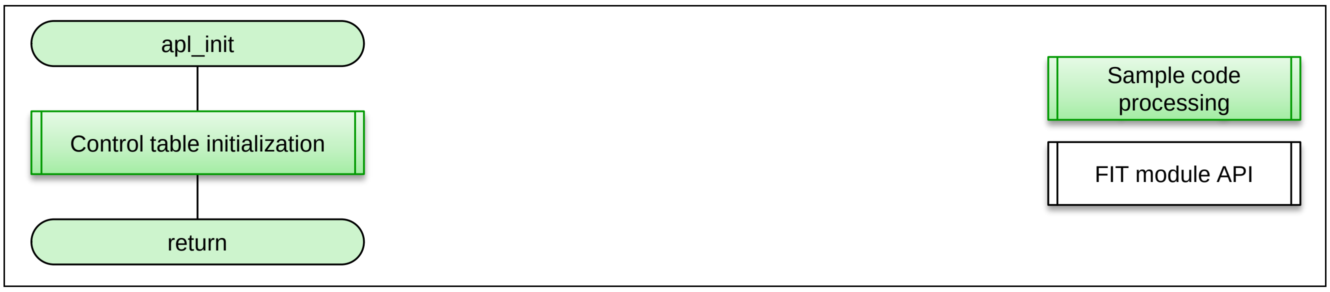


Figure 6-7 Control table initialization

(7) Registration of class driver

This registers the class driver.

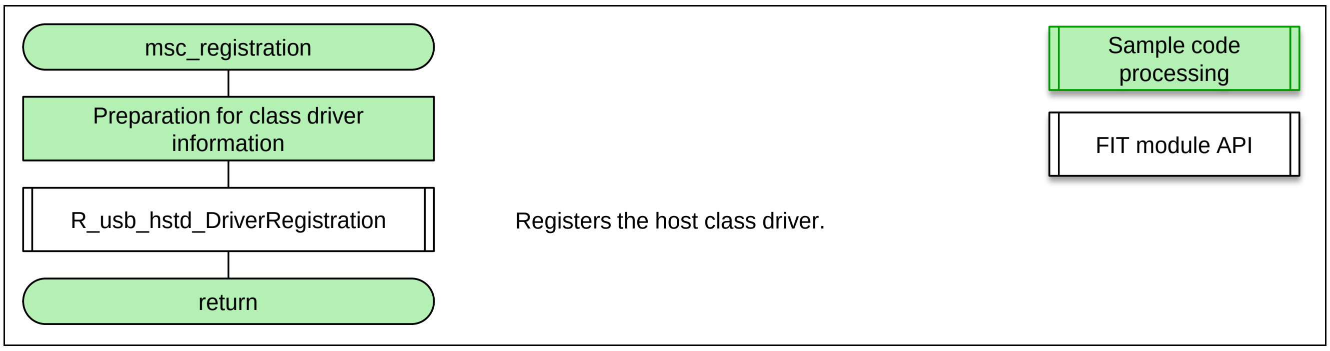

msc_registration
Preparation for class driver information
R_usb_hstd_DriverRegistration
Registers the host class driver.
return
Sample code processing
FIT module API

Figure 6-8 Registration of class driver

(8) USB HMSC driver task

This is processing for the USB HMSC driver task.

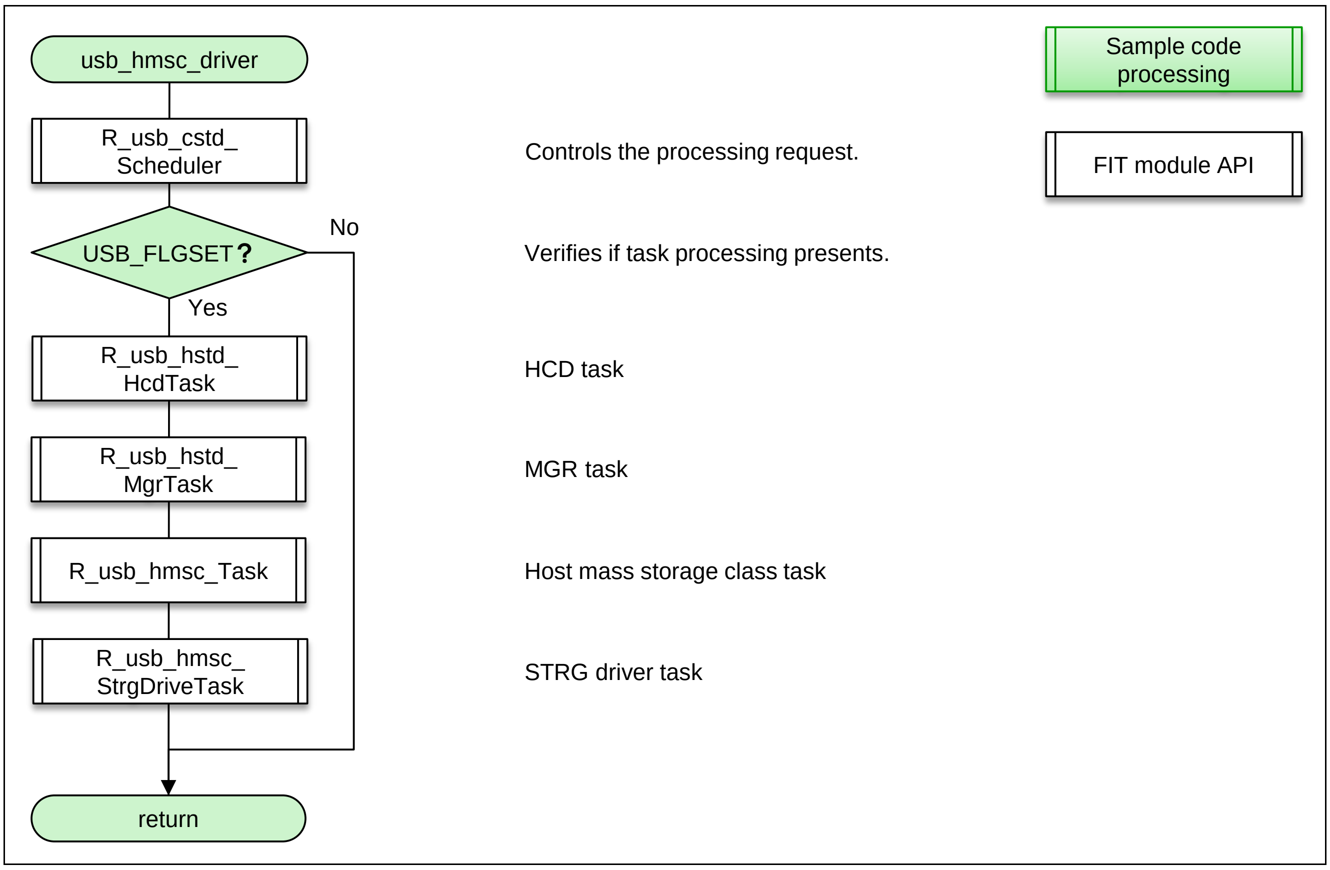


Figure 6-9 USB HMSC driver task

(9) Wait for USB device detection

This is the function to wait for USB device connection.

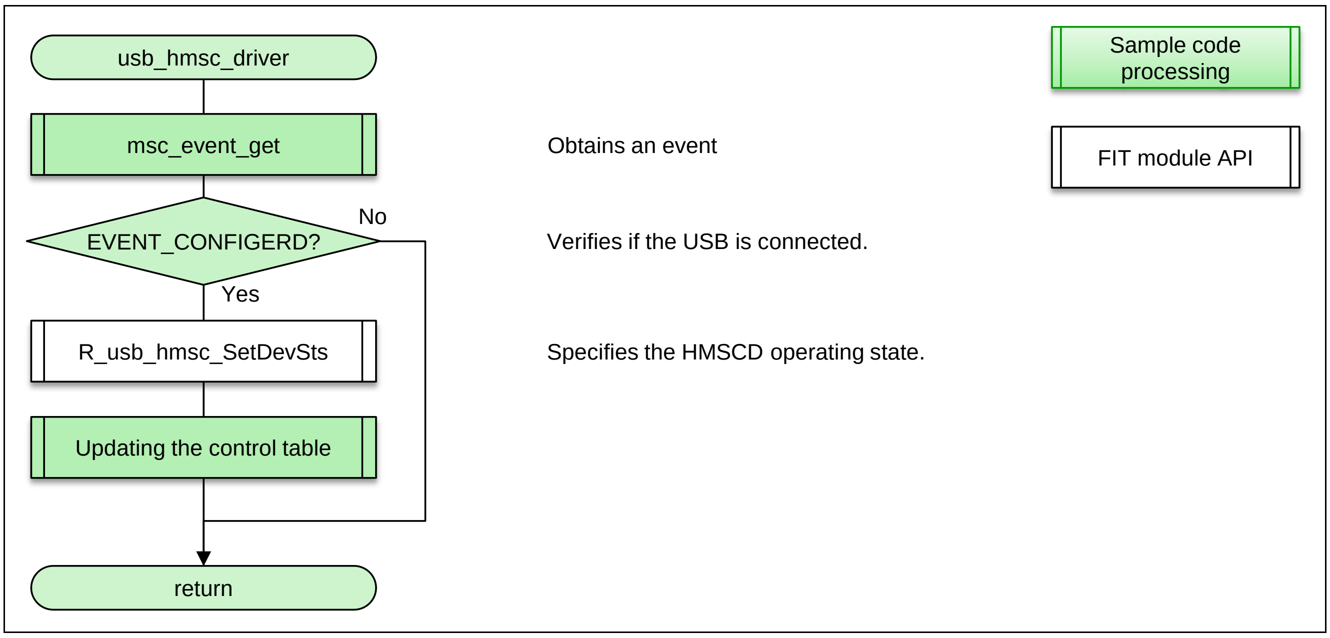


Figure 6-10 Wait for USB device detection

(10) USB device connection, and TFAT file system mount

This reads the USB drive information and mounts the TFAT file system.

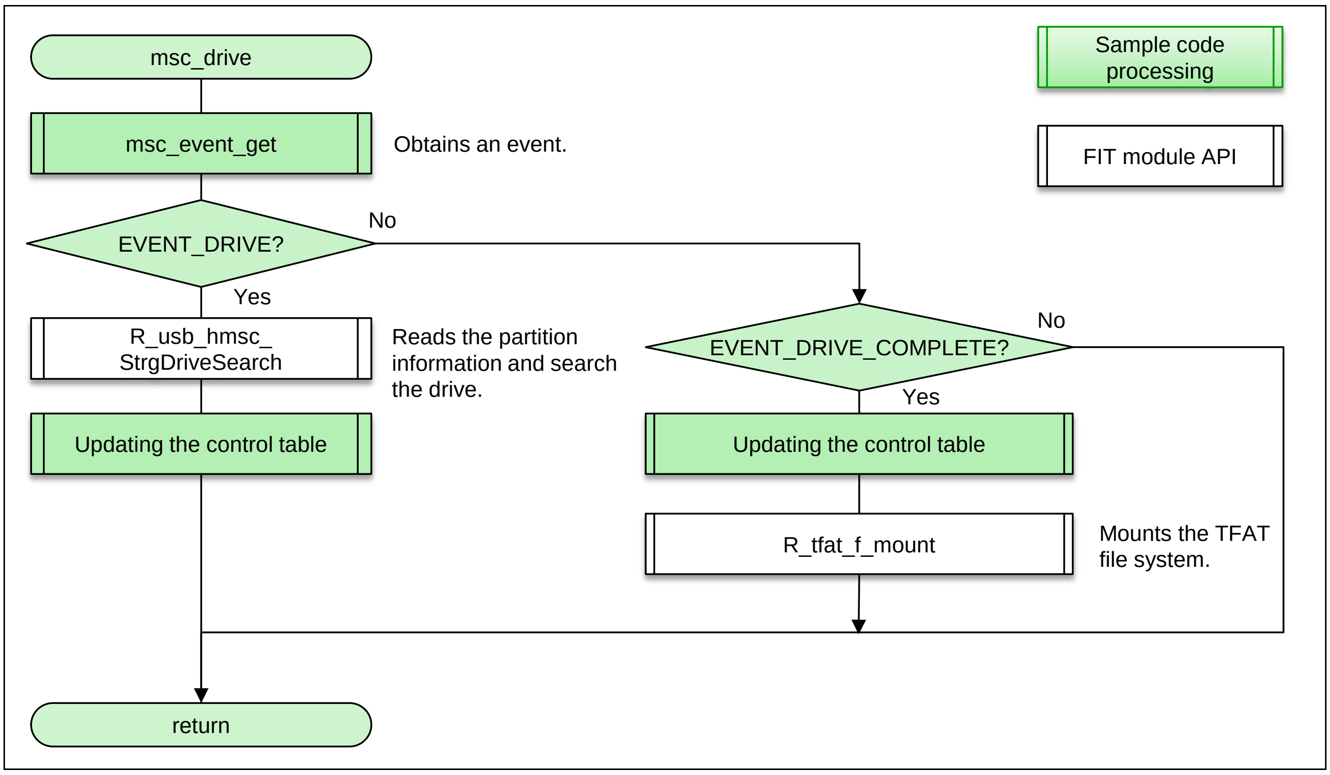

msc_drive
msc_event_get
Obtains an event.
EVENT_DRIVE?
No
Yes
R_usb_hmsc_StrgDriveSearch
Reads the partition information and search the drive.
Updating the control table
EVENT_DRIVE_COMPLETE?
No
Yes
Updating the control table
R_tfat_f_mount
Mounts the TFAT file system.
return
Sample code processing
FIT module API

Figure 6-11 USB device connection, and TFAT file system mount

(11) Setting before read state

This is the dummy control function.

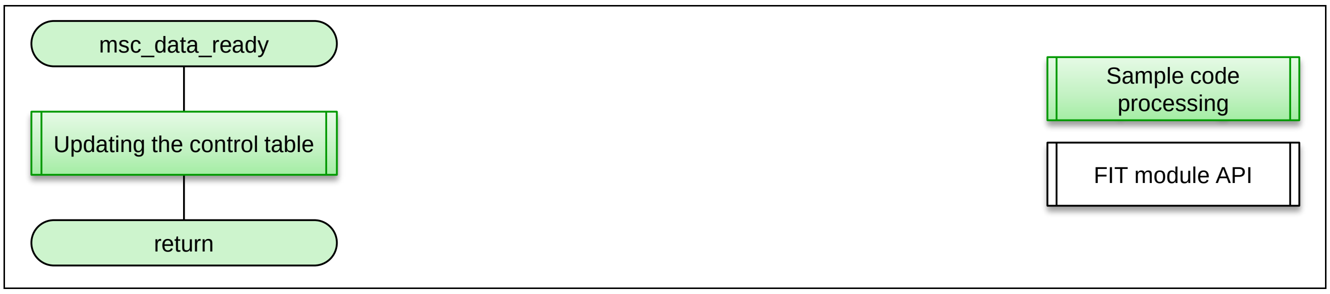


Figure 6-12 Setting before read state

(12) Data read from USB memory, and data write to flash memory

This reads the data from the USB memory and writes the data to the flash.

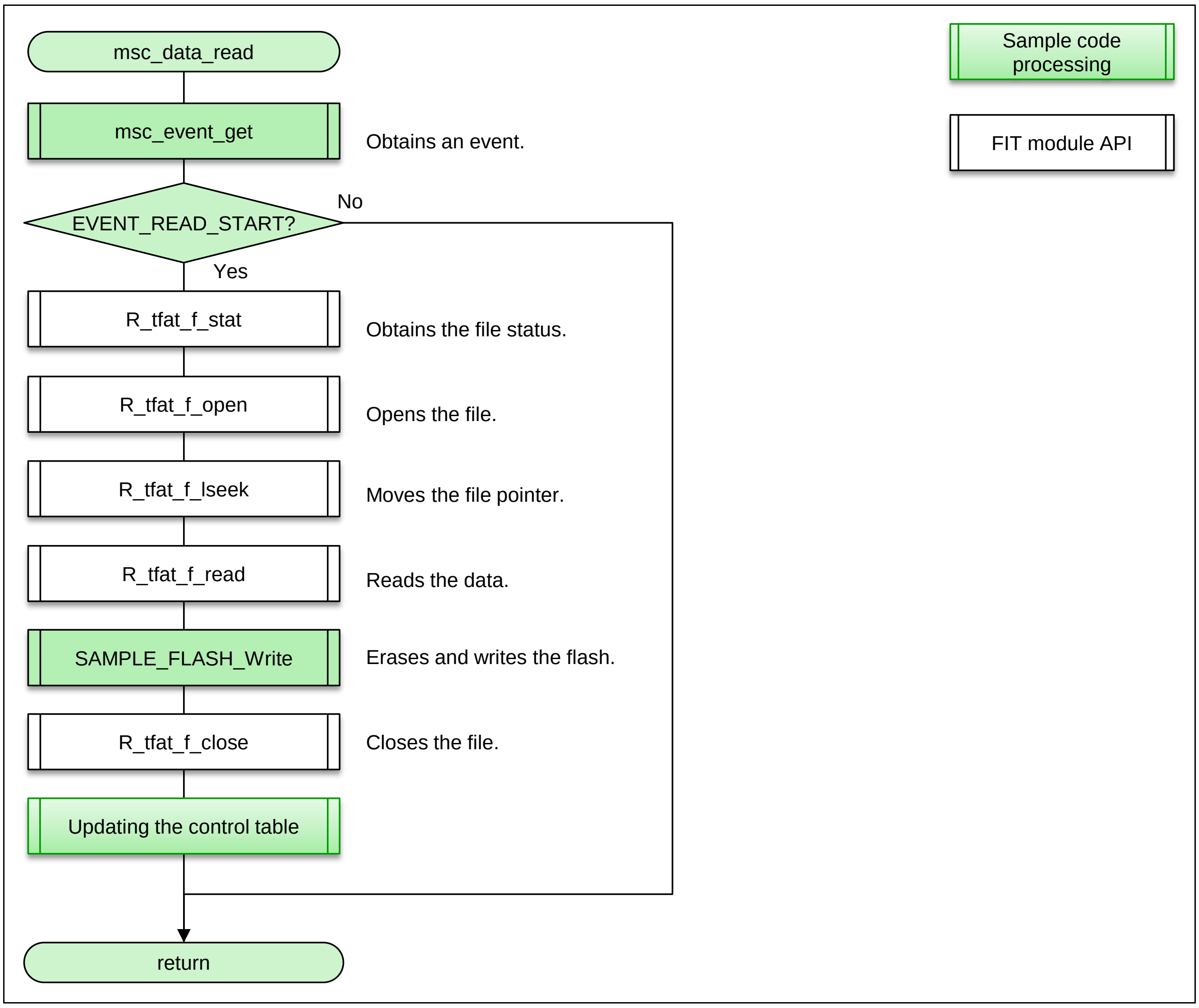
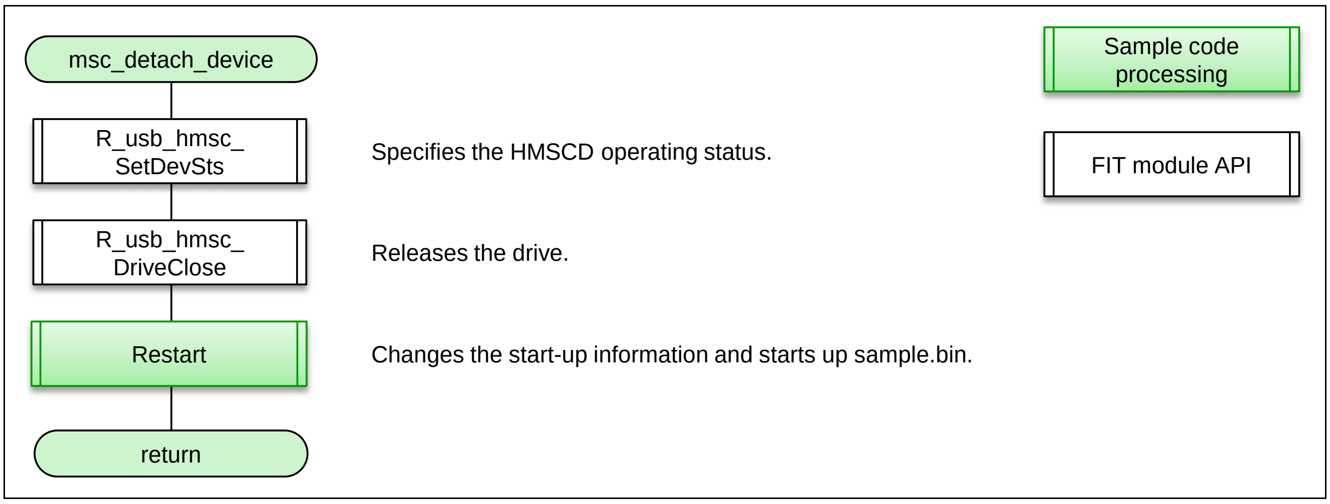


Figure 6-13 Data read from USB memory, and data write to Flash memory

(13) USB disconnection

This disconnects the USB device and starts up the sample program.



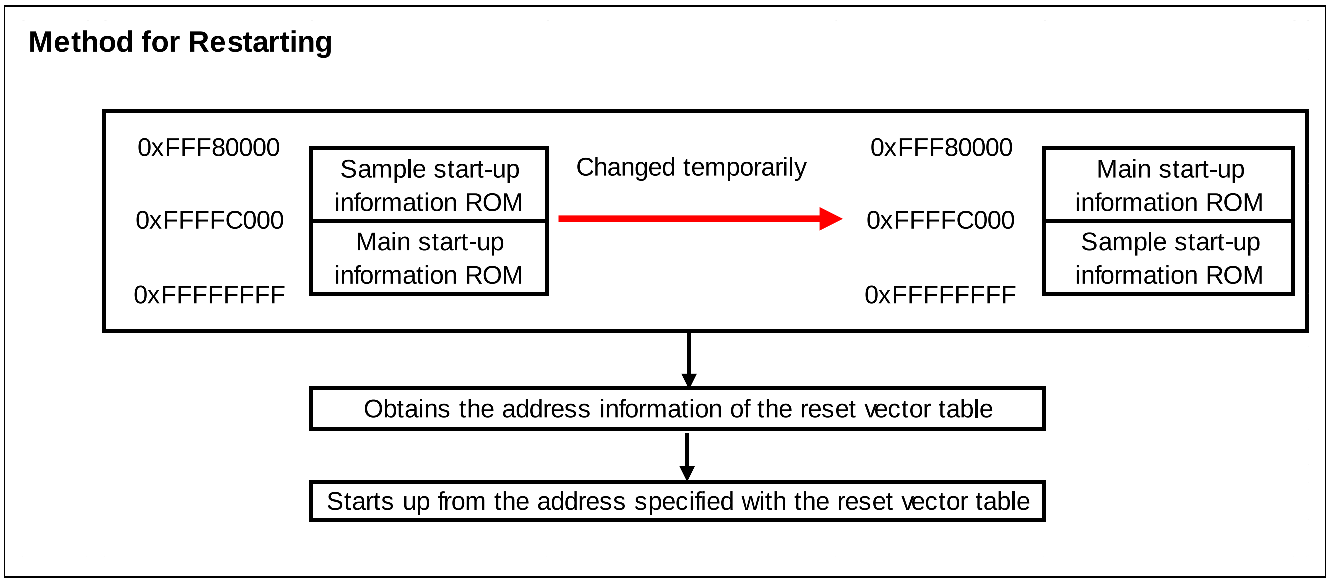

Method for Restarting
0xFFF80000
0xFFFFC000
0xFFFFFFFF
Sample start-up information ROM
Main start-up information ROM
Changed temporarily
0xFFF80000
0xFFFFC000
0xFFFFFFFF
Main start-up information ROM
Sample start-up information ROM
Obtains the address information of the reset vector table
Starts up from the address specified with the reset vector table

Figure 6-14 USB disconnection

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.02	Mar 31, 2017	-	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.
5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0