RX231 Group

On-chip Flash Memory Programming Solution using Capacitive Touch Sensing Unit and USB Memory

RX Driver Package Application

R01AN2561EJ0102
Rev.1.02
Feb 29. 2016

Introduction

This document is an application note of the on-chip flash memory programming solution using RX231 built-in Capacitive Touch Sensing Unit (CTSU) and USB Memory.

This application note includes the main program that writes the program stored in the USB memory into the RX231 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

RX231 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, and Capacitive Touch Sensor Unit (CTSU).

This application note operates on the Renesas Starter Kit+ for RX231 (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	RX231 Group
Evaluation board	Renesas Starter Kit for RX231 (Part No.: R0K505231S000BE)
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 1-1 lists the FIT modules to be included in the main program. Some modules are included in the sample program.

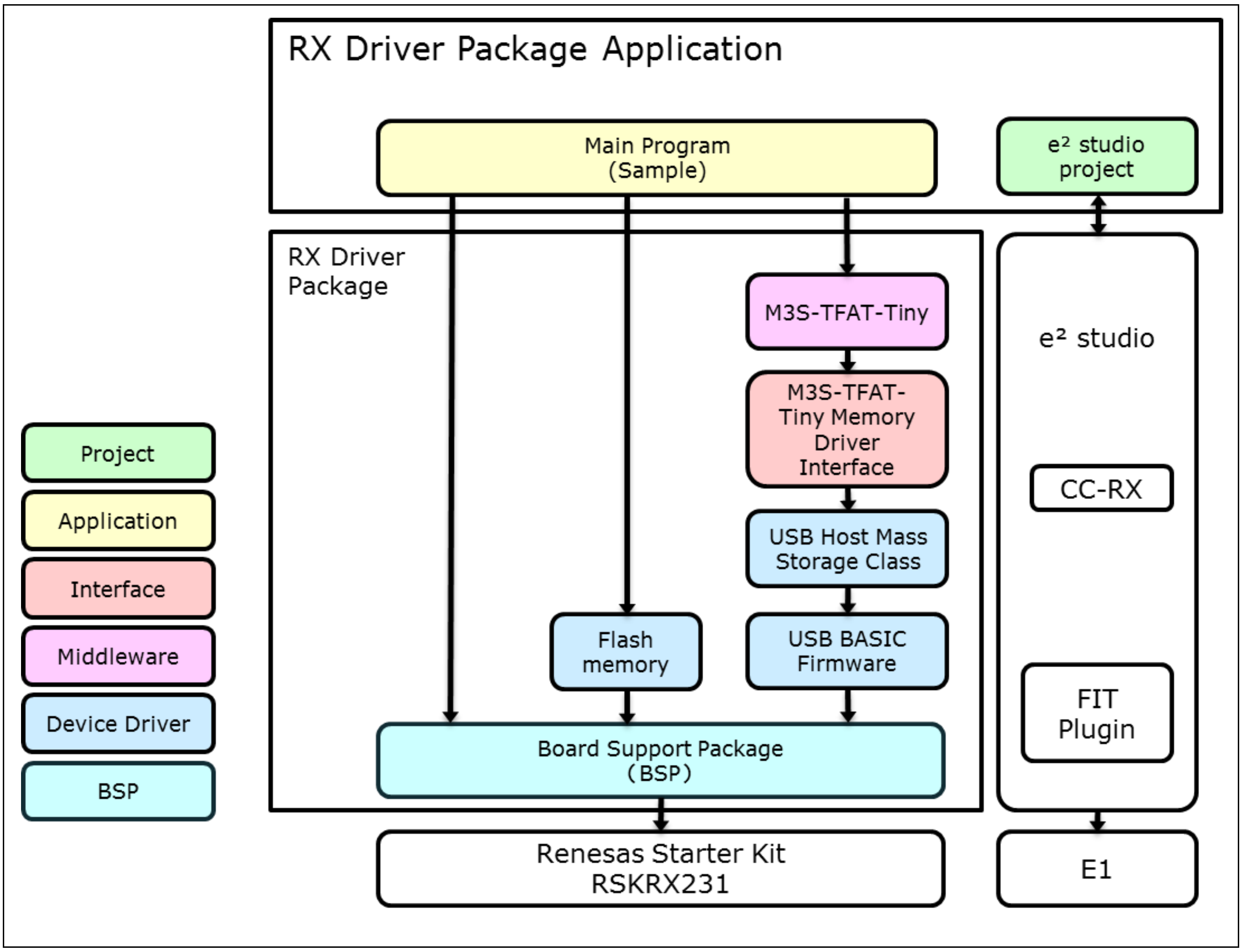


Figure 1-1 Module Structure

Table 1-1 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
	Flash memory (Flash API)	r_flash_rx	1.30
Application	Main program FIT module	r_flash_writer_rx231	1.00

1.4 File Structure

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

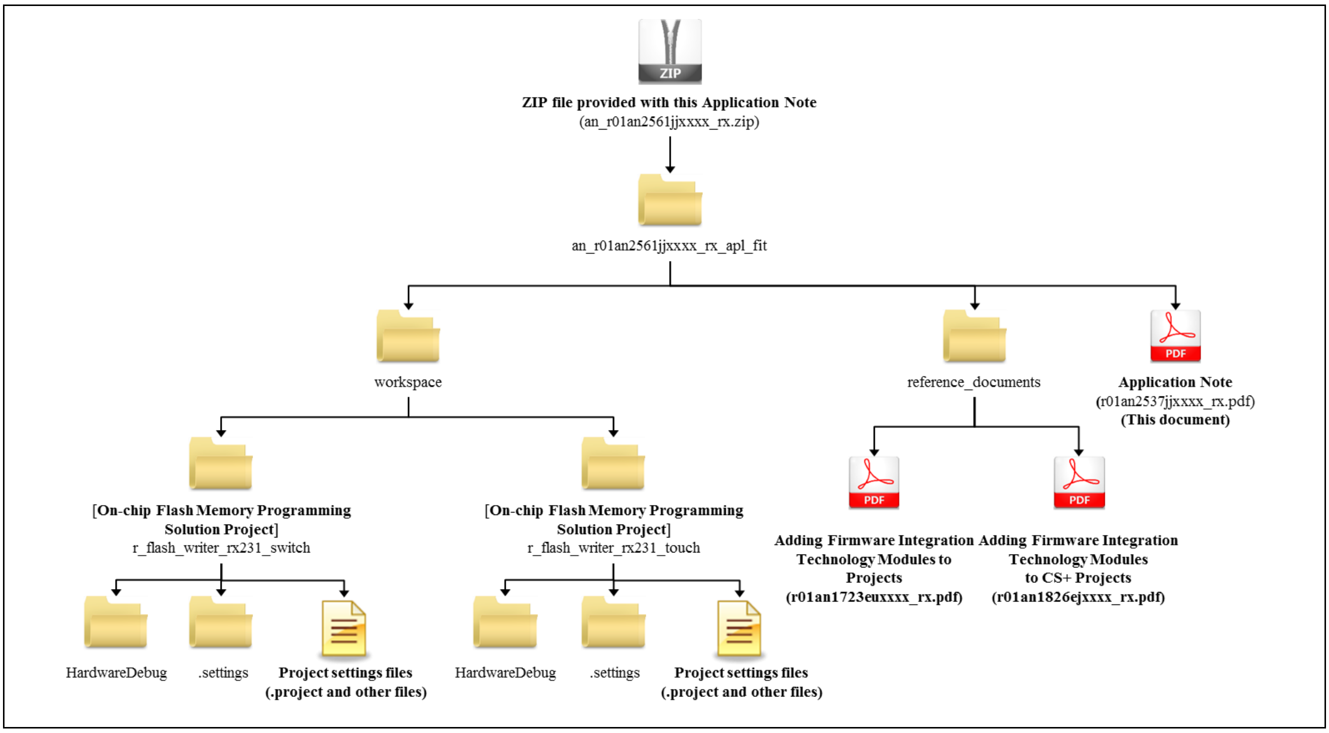


Figure 1-2 File Structure

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

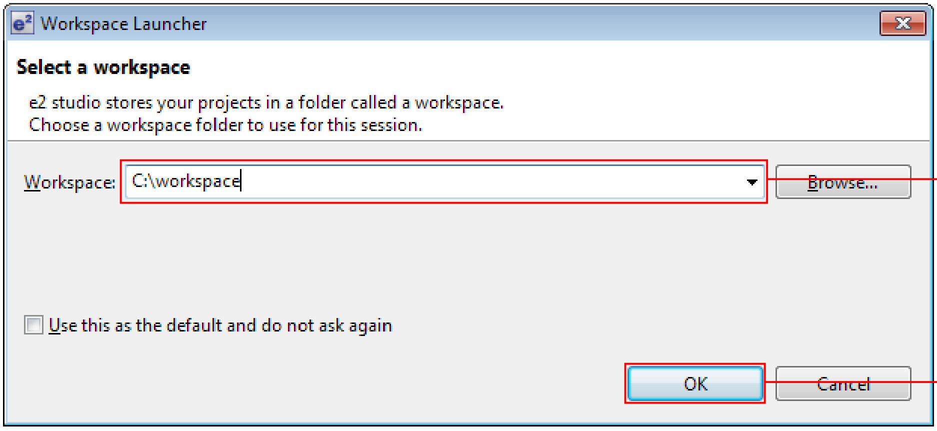
3. Building a Project

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

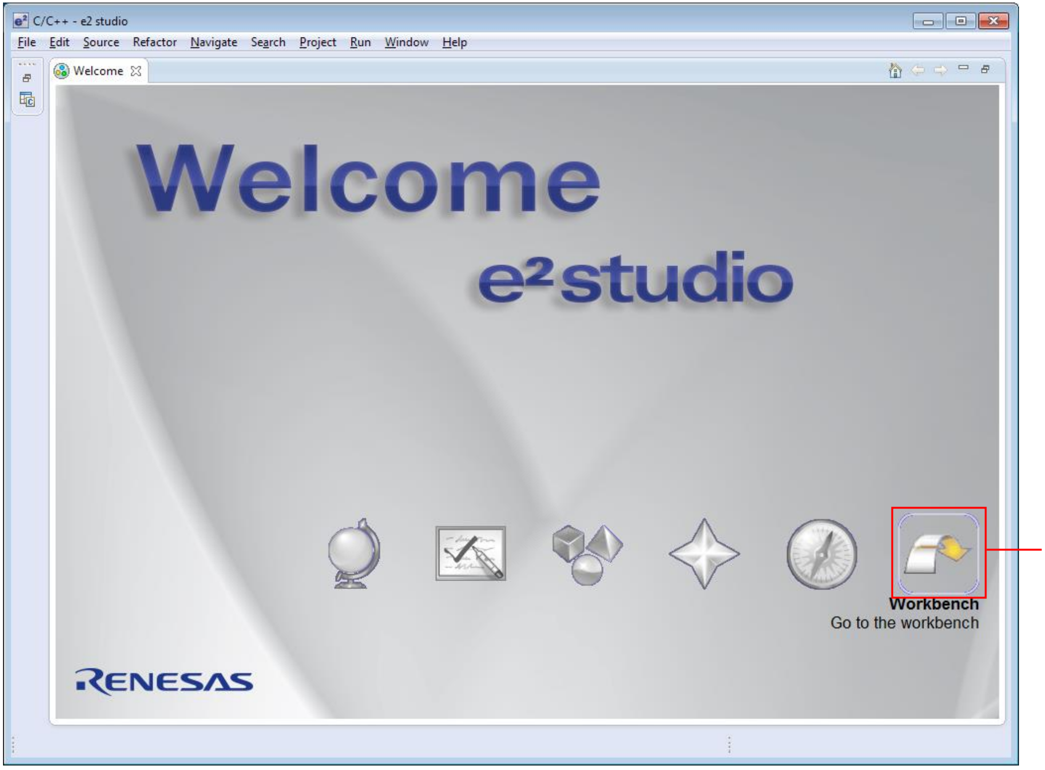
e2 studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\workspace

Enter a workspace folder.

Click **[OK]**.

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



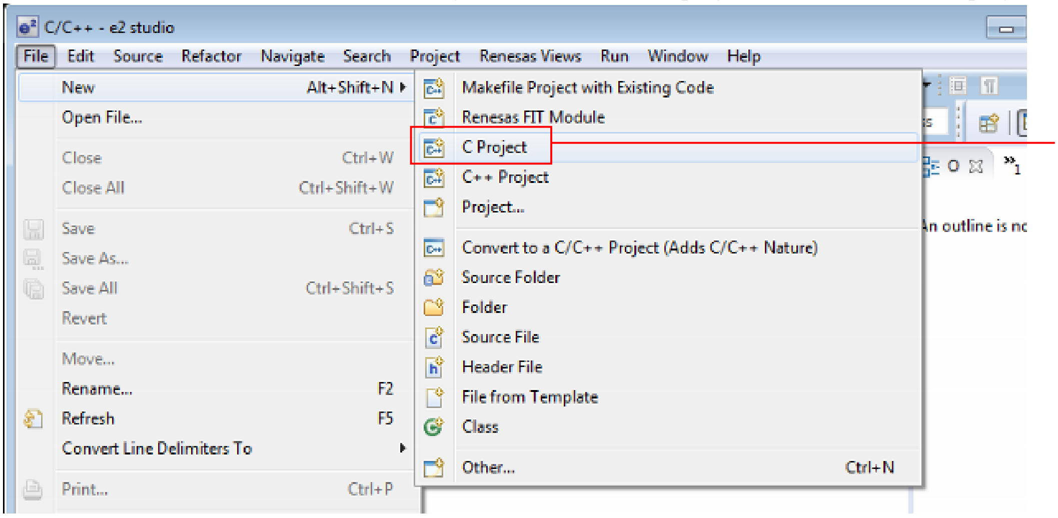
Click **[Workbench]**.

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).

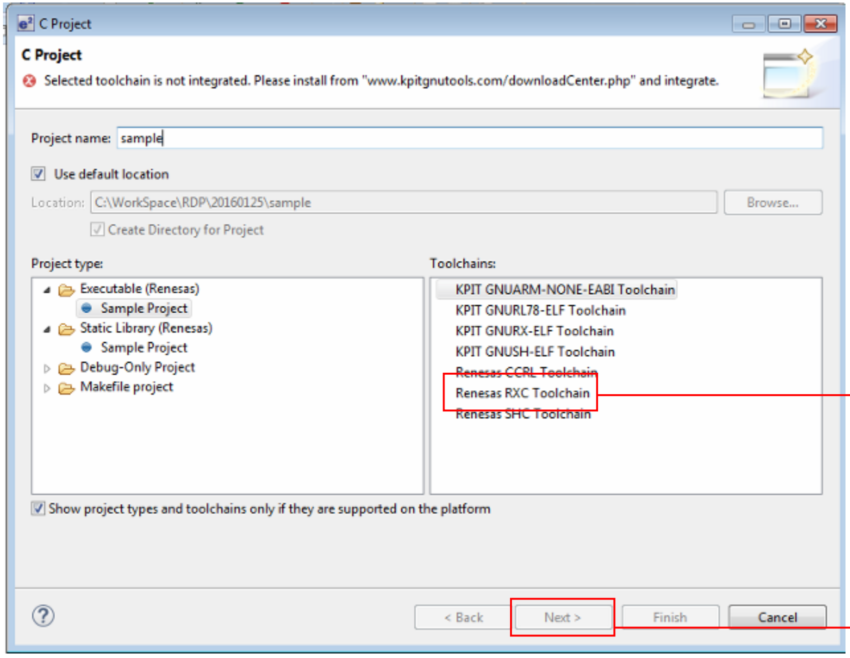
Note 1: The project to be created here is a dummy to use the Smart browser.

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



Click here.

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

☑ Create Directory for Project

Project type:

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

Toolchains:

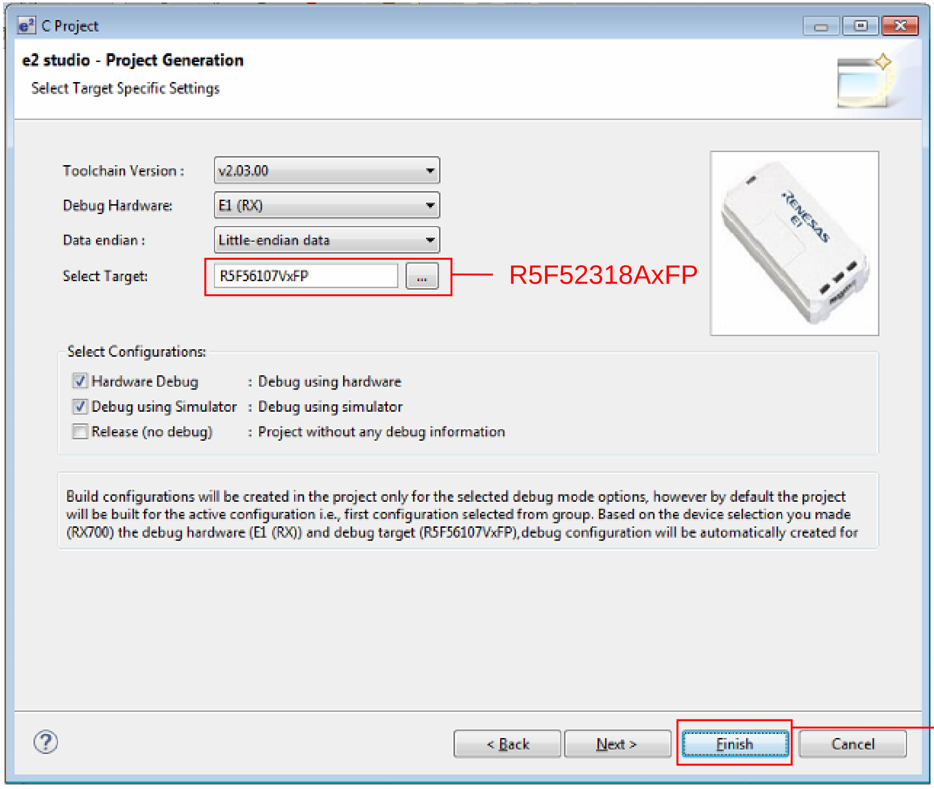
- KPIT GNUARM-NONE-EABI Toolchain
- KPIT GNURL78-ELF Toolchain
- KPIT GNURX-ELF Toolchain
- KPIT GNUSH-ELF Toolchain
- Renesas CCRL Toolchain
- Renesas RXC Toolchain
- Renesas SHC Toolchain

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

Select “Renesas RXC Toolchain”.

Click here.

3. Set Select Target to R5F52318AxFP for RX231 100 Pin device. For other items, any setting is OK. When the setting is completed, click [Finish(F)].



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

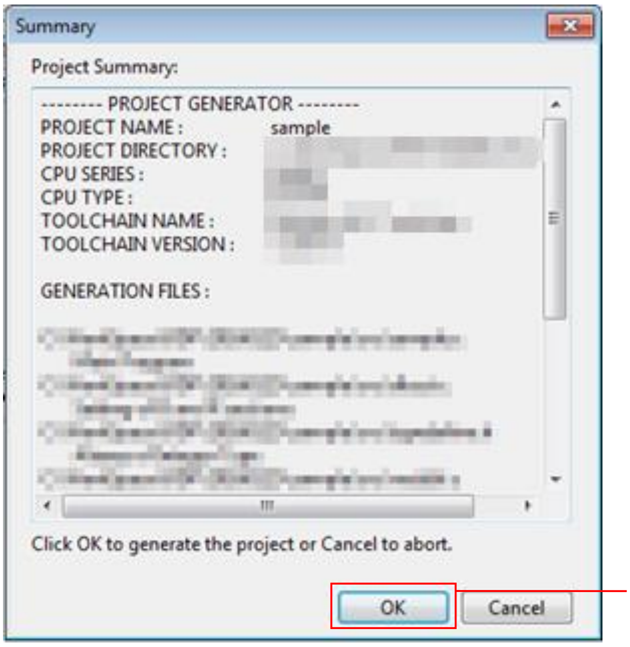
□ Release (no debug) : Project without any debug information

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

R5F52318AxFP

Click here.

4. Click [OK].



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.

Click here.

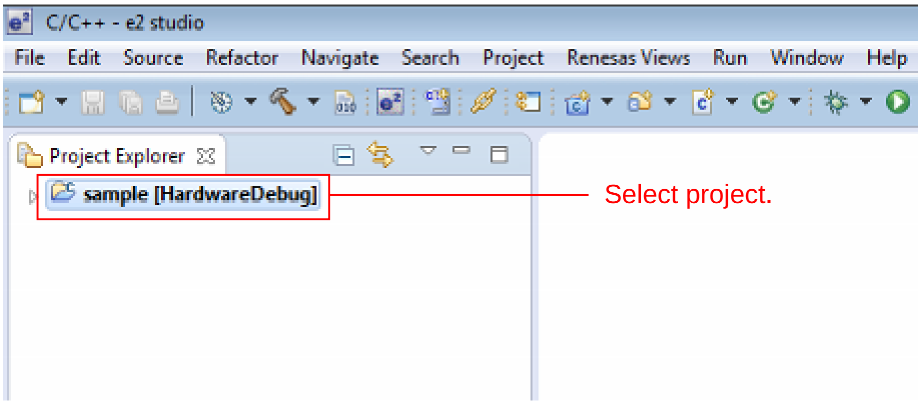
3.3 Import a Project

Import the project of Main program in the workspace created.

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

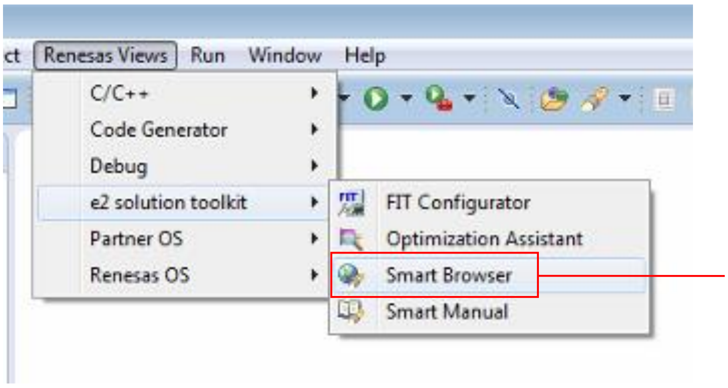
- Capacitive Touch Sensor Unit (CTSU)
- Switch

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]

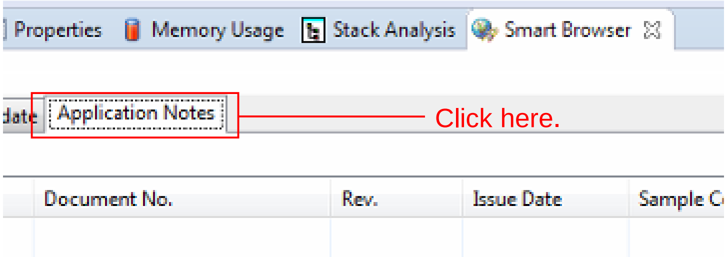
Select project.

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



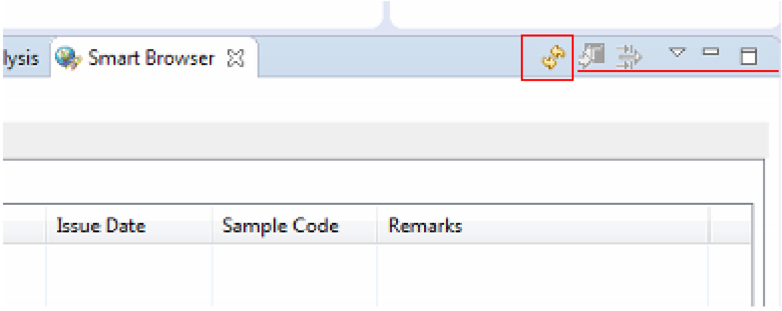
Click here.

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


Properties Memory Usage Stack Analysis Smart Browser
date Application Notes
Document No. Rev. Issue Date Sample C

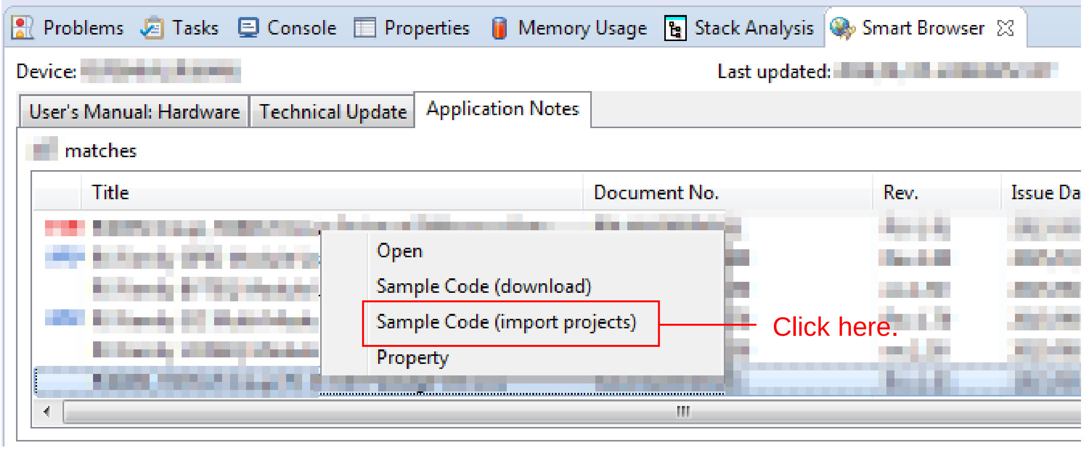
Click here.

4. Click [Update].

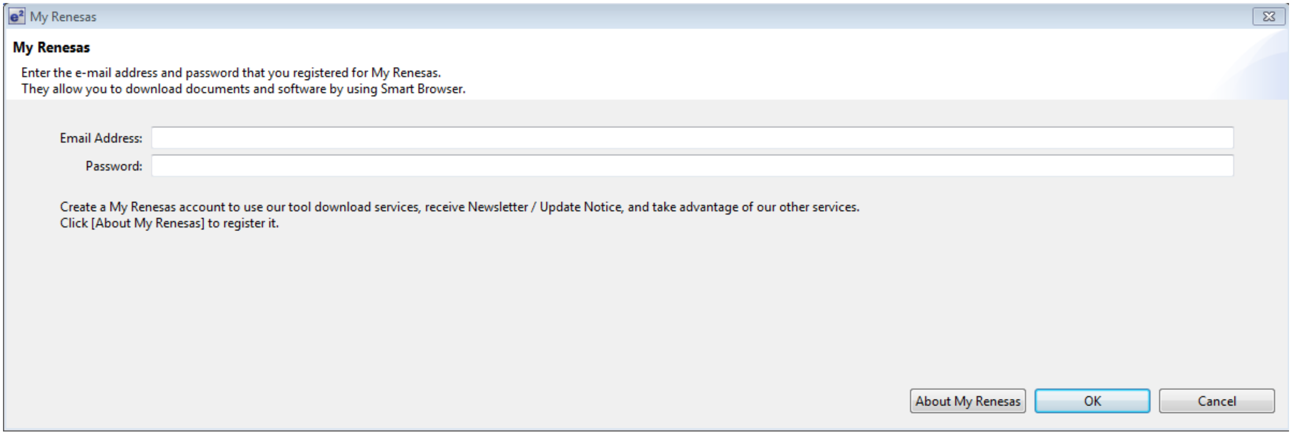


Click here.

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

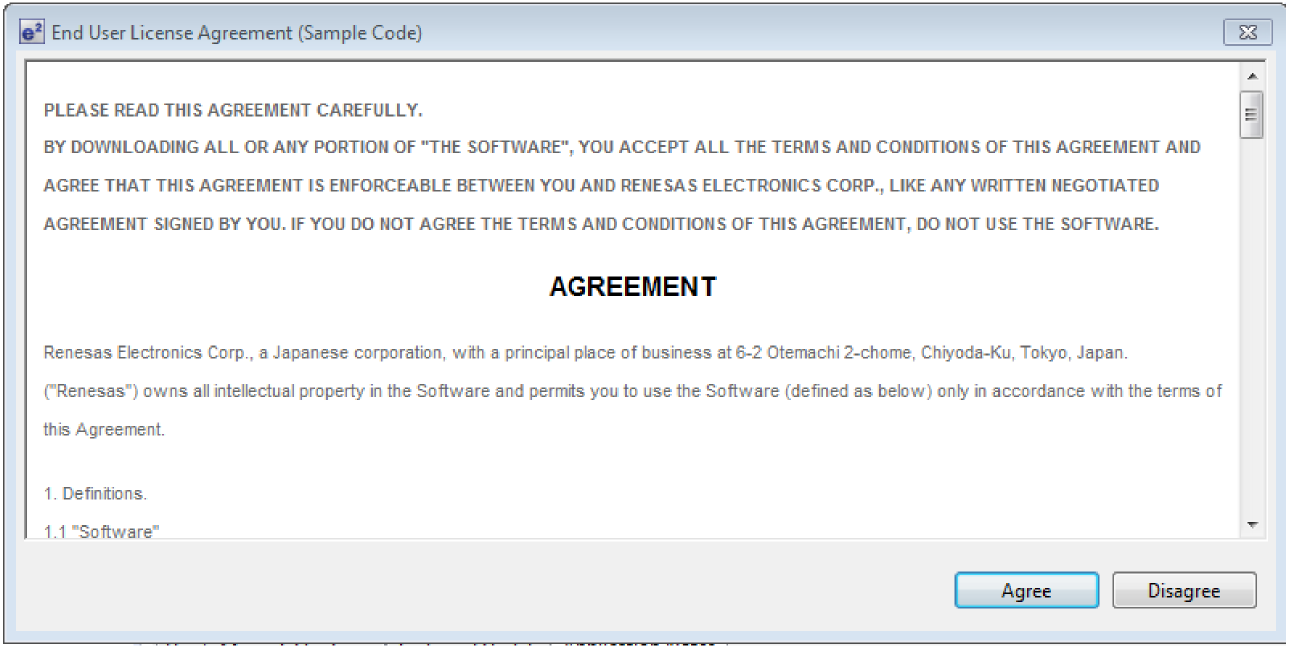
Enter the e-mail address and password that you registered for My Renesas.
They allow you to download documents and software by using Smart Browser.

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].



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.

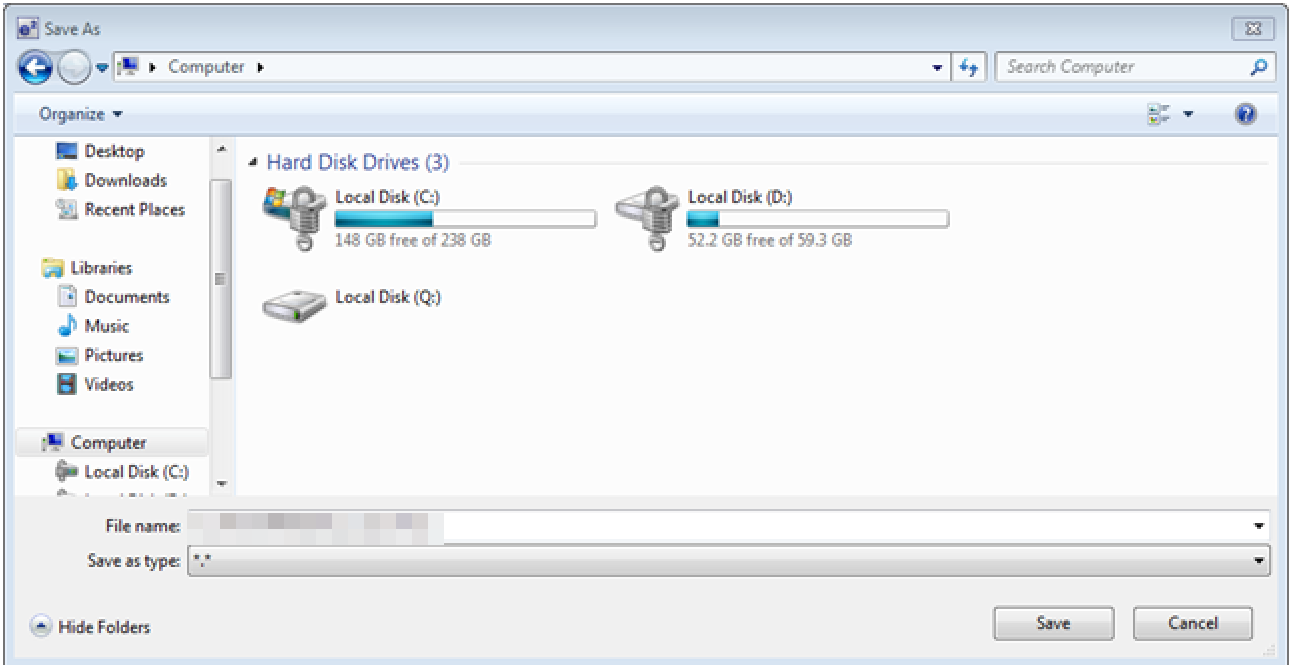
AGREEMENT

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"

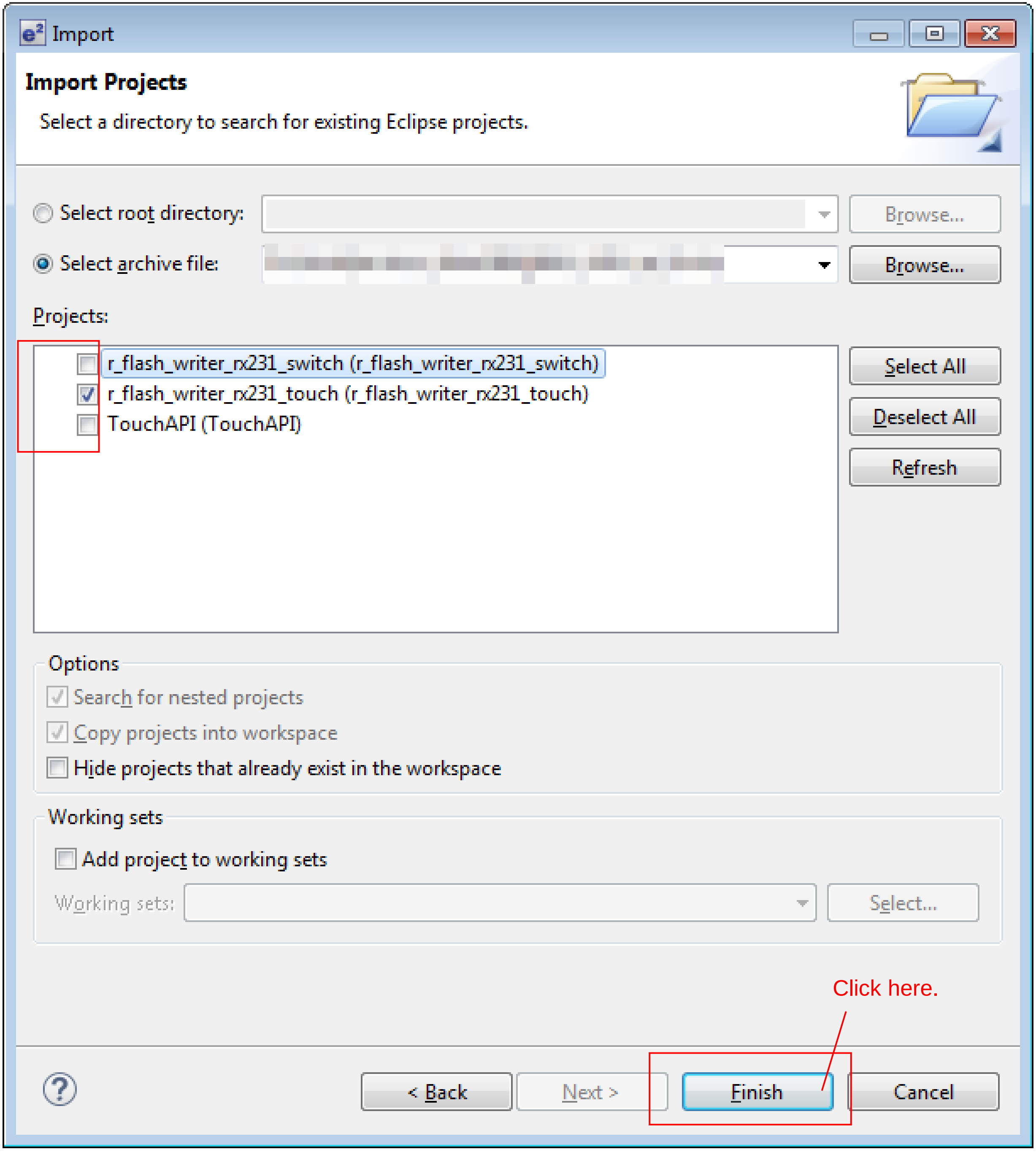
7. Save the application note.



File name: [redacted]

Save as type: *.*

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



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

● Select archive file: [redacted]

Projects:

□ r_flash_writer_rx231_switch (r_flash_writer_rx231_switch)
☑ r_flash_writer_rx231_touch (r_flash_writer_rx231_touch)
□ TouchAPI (TouchAPI)

Options

☑ Search for nested projects
☑ Copy projects into workspace
□ Hide projects that already exist in the workspace

Working sets

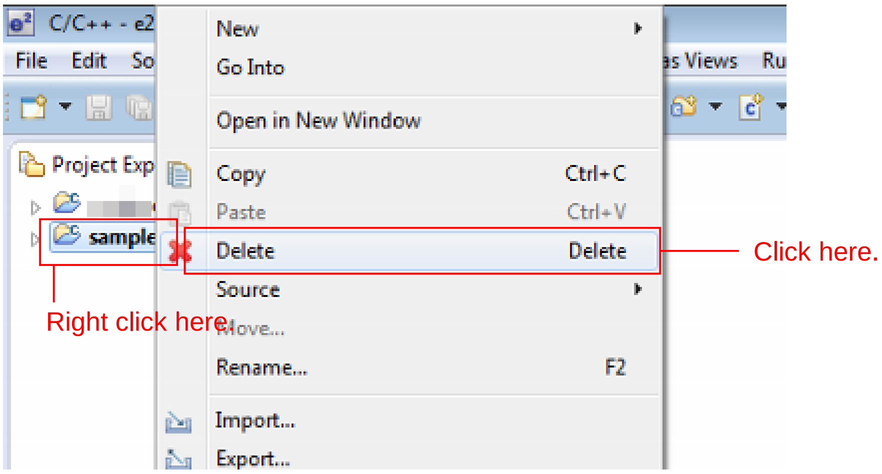
□ Add project to working sets

Working sets:

This application note includes the following projects.

Project name	Contents
r_flash_writer_rx231_switch	Project that selects a file by switch
r_flash_writer_rx231_touch	Project that selects a file by touch
TouchAPI	Project used to control the touch sensor unit using Workbench6. Not used for the sample project.

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



Click here.

Right click here.

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.

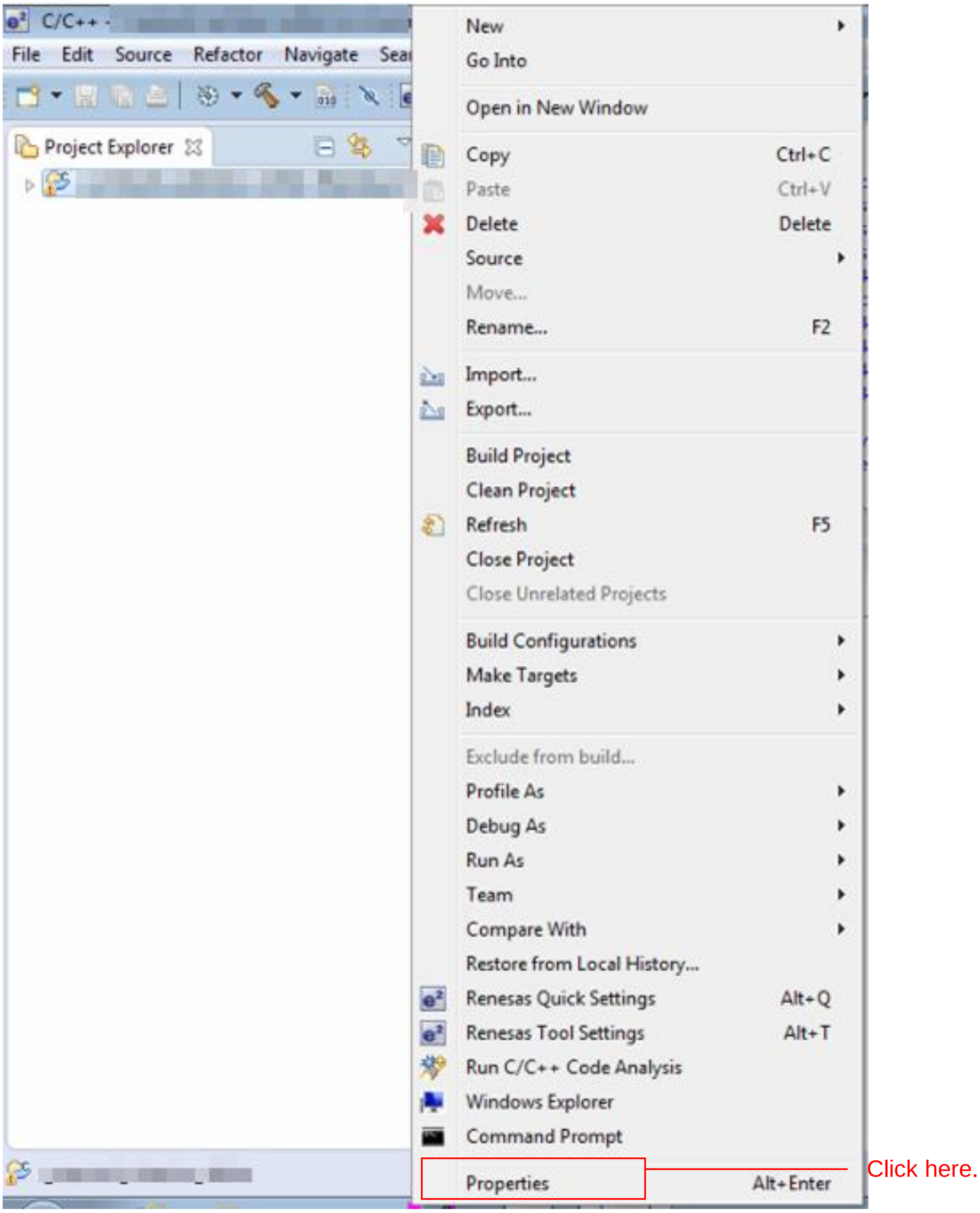
【r_config/r_usb_hmsc_mini_config.h】

```
    #define USB_TFAT_USE_PP
```

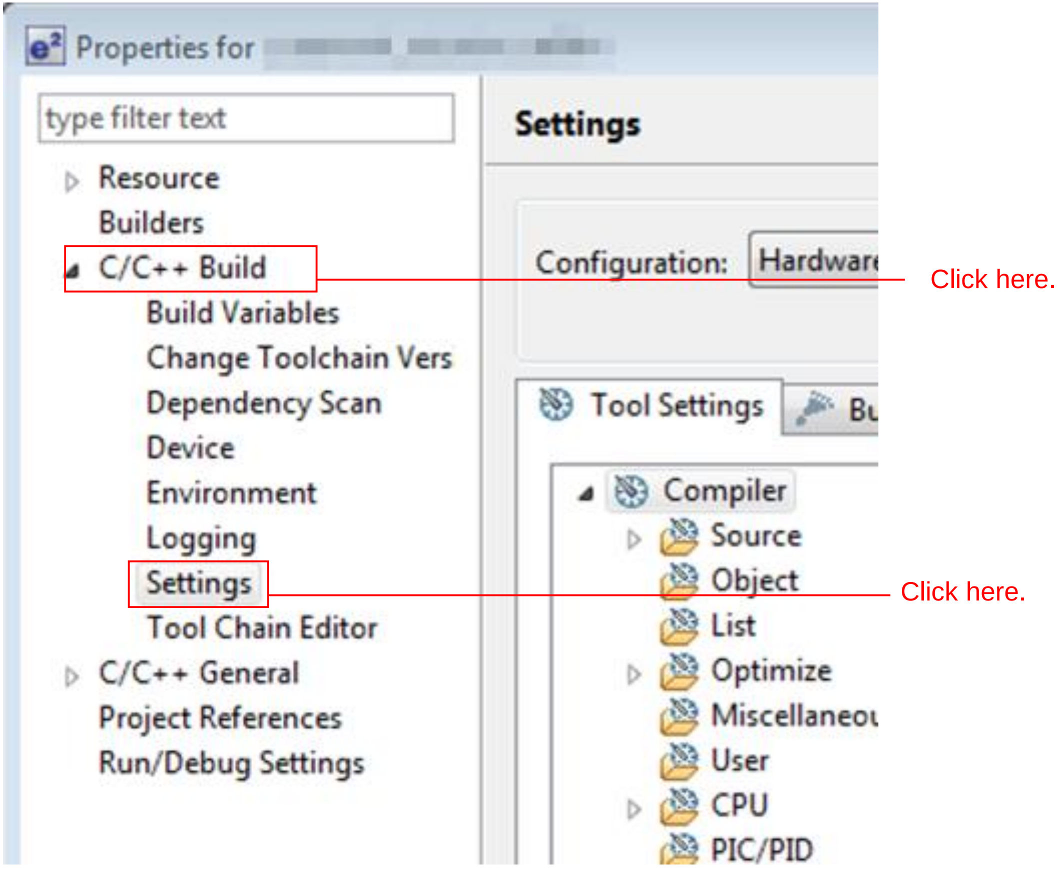
3.4.2 Change Project Setting

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)].



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



Click here.

Click here.

— Project setting of the main program

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

Table 3-1 Changed build setting

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 3-1 Sample1 & sample2, and in Table 3-2 for Sample3.

Table 3-3 Changed build setting (sample1 & sample2)

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 EXCEPTVECT section to "0xFFFF BF80"	After using the start-up program protection feature, set the address that becomes "0xFFFF FF80"
	Change the address of RESETVECT section to "0xFFFF BFFC"	After using the start-up program protection feature, set the address that becomes "0xFFFF FFFC"
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.

Table 3-4 Changed build setting (sample3)

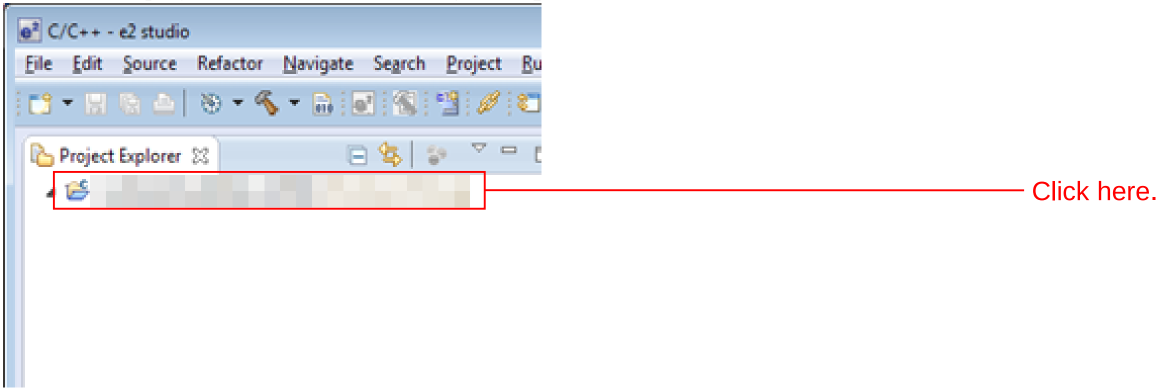
Items	Changed contents	Description
Linker - Section	Remove PResetPRG and PIntPRG from section definition (Note)	Requires the setting when using BSP (required when using FIT)
	Change P Section to P* Section (Note)	Requires the setting when using BSP (required when using FIT)
	Change the address pf C_1 section to “0xFFFF 0000”	Define start address to program
	Add _MDEREG section (address “0xFFFF BF80”)	After using the start-up program protection feature, set the address that becomes “0xFFFF FF80”, and change the source code as follows. Source/HwResource/r_cg_vecttbl.c Old: #pragma address id_code=0xffffffa0 New: #pragma section C ID_CODE
	Add OFS1_LOCATION section (address “0xFFFF BF88”)	After using the start-up program protection feature, set the address that becomes “0xFFFF FF88”, and change the source code as follows. Source/HwResource/r_cg_vecttbl.c Old: #pragma address id_code=0xffffffa0 New: #pragma section C ID_CODE
	Add OFS0_LOCATION section(address “0xFFFF BF8C”)	After using the start-up program protection feature, set the address that becomes “0xFFFF FF8C”, and change the source code as follows. Source/HwResource/r_cg_vecttbl.c Old: #pragma address id_code=0xffffffa0 New: #pragma section C ID_CODE
	Add ID_CODE section (address ” 0xFFFF BFA0”)	After using the start-up program protection feature, set the address that becomes “0xFFFF BFA0”, and change the source code as follows. Source/HwResource/r_cg_vecttbl.c Old: #pragma address id_code=0xffffffa0 New: #pragma section C ID_CODE
	Add FIXEDVECT section (address ”0xFFFF BFD0”)	After using the start-up program protection feature, set the address that becomes “0xFFFF BFD0”.
Linker - Output	Change output file/type to “Binary via absolute”	Set the file type to write in the USB memory.

4. Verify Operation

4.1 Build the Project

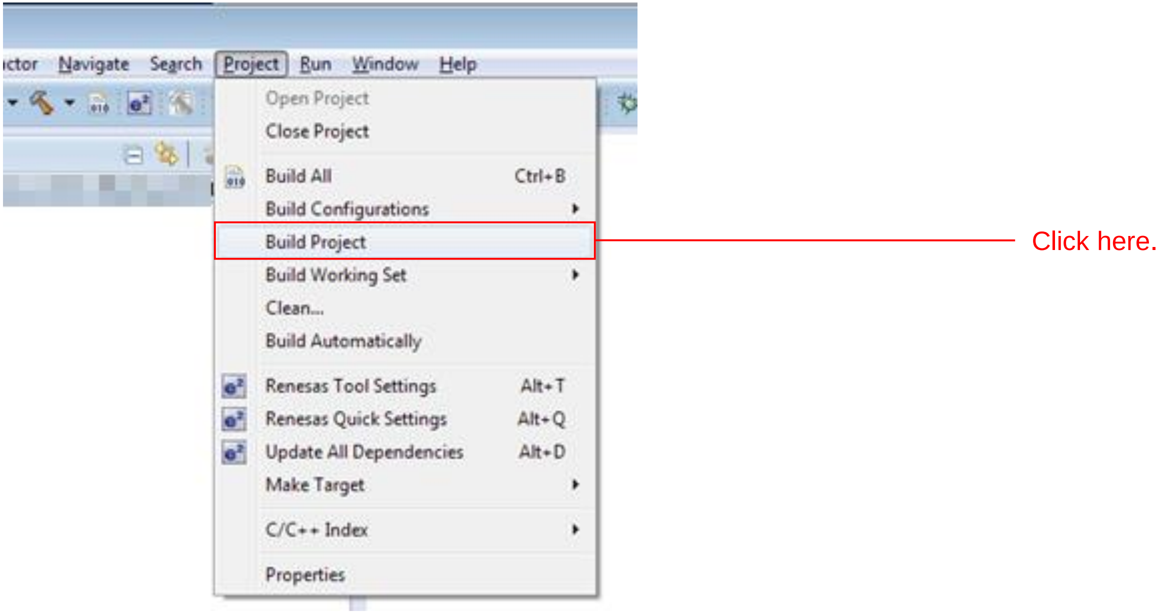
Use the following procedure to build the project and generate a load module.

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



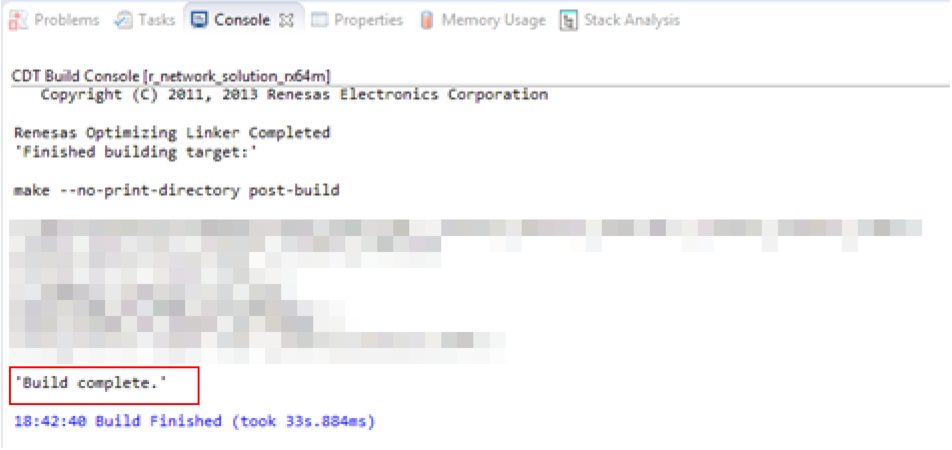
Click here.

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



Click here.

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 RX231
USB memory	Memory that is formatted as either FAT or FAT32.

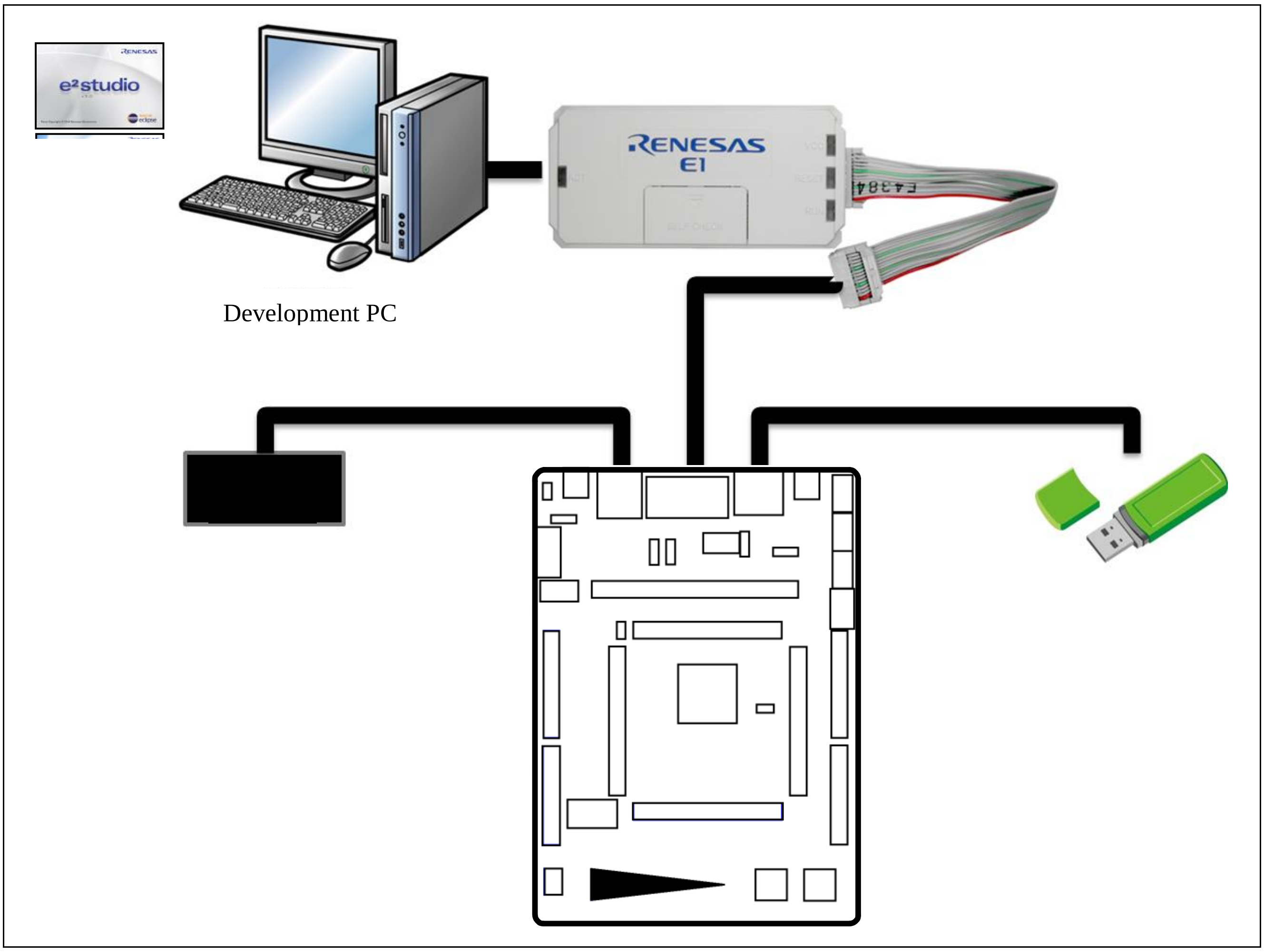


Figure 4-1 Operating environment example

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 J15 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)	J15	Short 1 to 2. (*selected this time)
When use USB in peripheral mode. (USB_FUNCSEL_PP = USB_PERI_PP)	J15	Short 2 to 3.

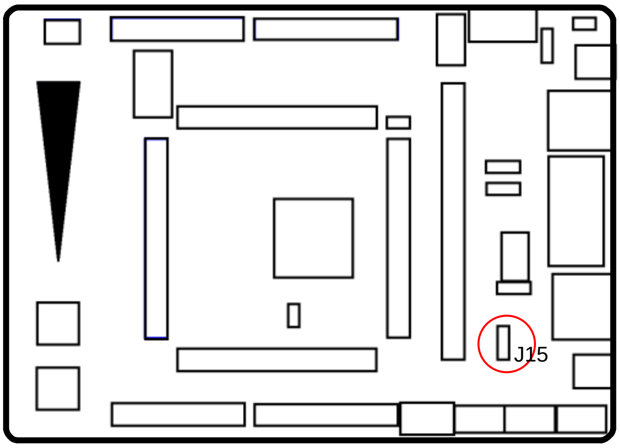
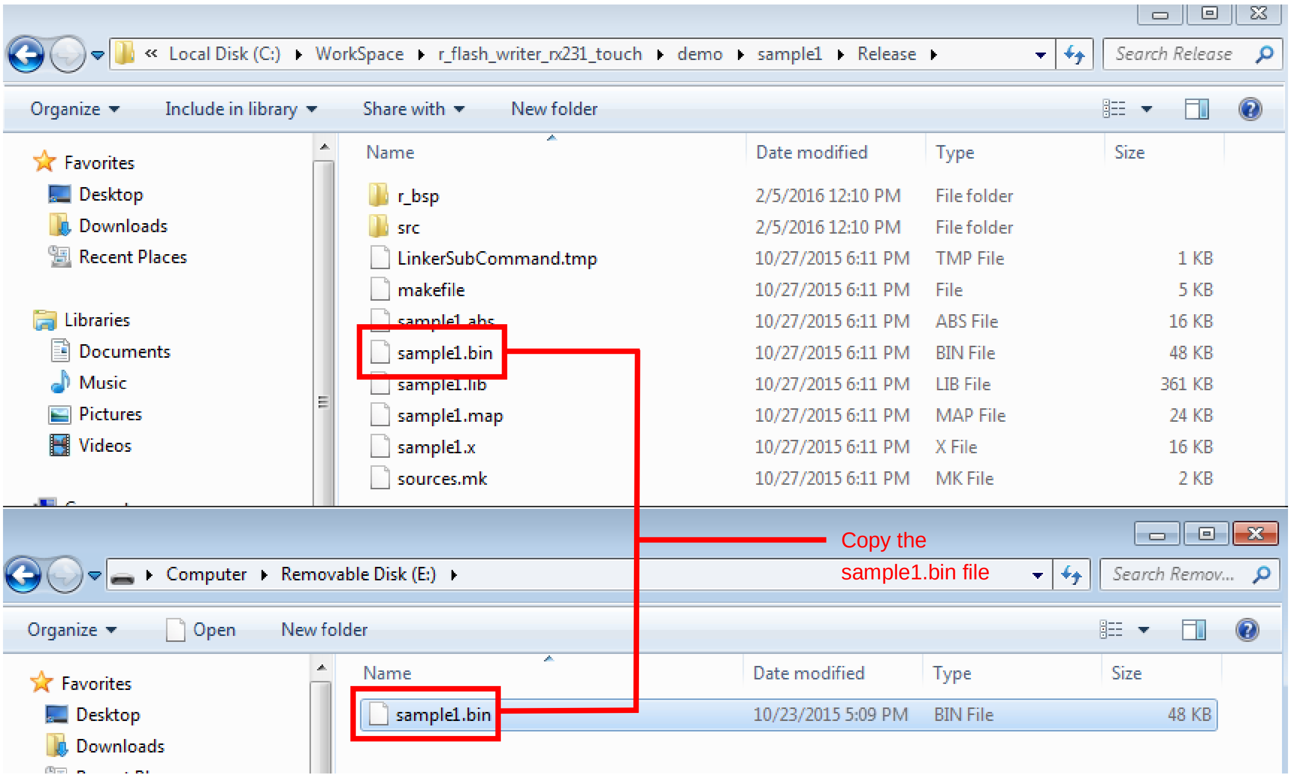


Figure 4-2 RSK Jumper Locations

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 **sample1.bin** file in the decompressed **sample1/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
sample1.abs	10/27/2015 6:11 PM	ABS File	16 KB
sample1.bin	10/27/2015 6:11 PM	BIN File	48 KB
sample1.lib	10/27/2015 6:11 PM	LIB File	361 KB
sample1.map	10/27/2015 6:11 PM	MAP File	24 KB
sample1.x	10/27/2015 6:11 PM	X File	16 KB
sources.mk	10/27/2015 6:11 PM	MK File	2 KB

Copy the sample1.bin file

Name	Date modified	Type	Size
sample1.bin	10/23/2015 5:09 PM	BIN File	48 KB

The **sample2.zip** file included in **demo** folder can be used as a sample program as with **sample1.zip**. For use, copy **sample2.bin** file in the **sample2/release** folder to the USB memory.

For **sample3.zip** file, copy **sample3.bin** file in **sample3/DefaultBuild** folder to the USB memory.

For the operation of sample1 program, sample2 program, and sample3, refer to 4.4 Verify operation of Sample program.

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.

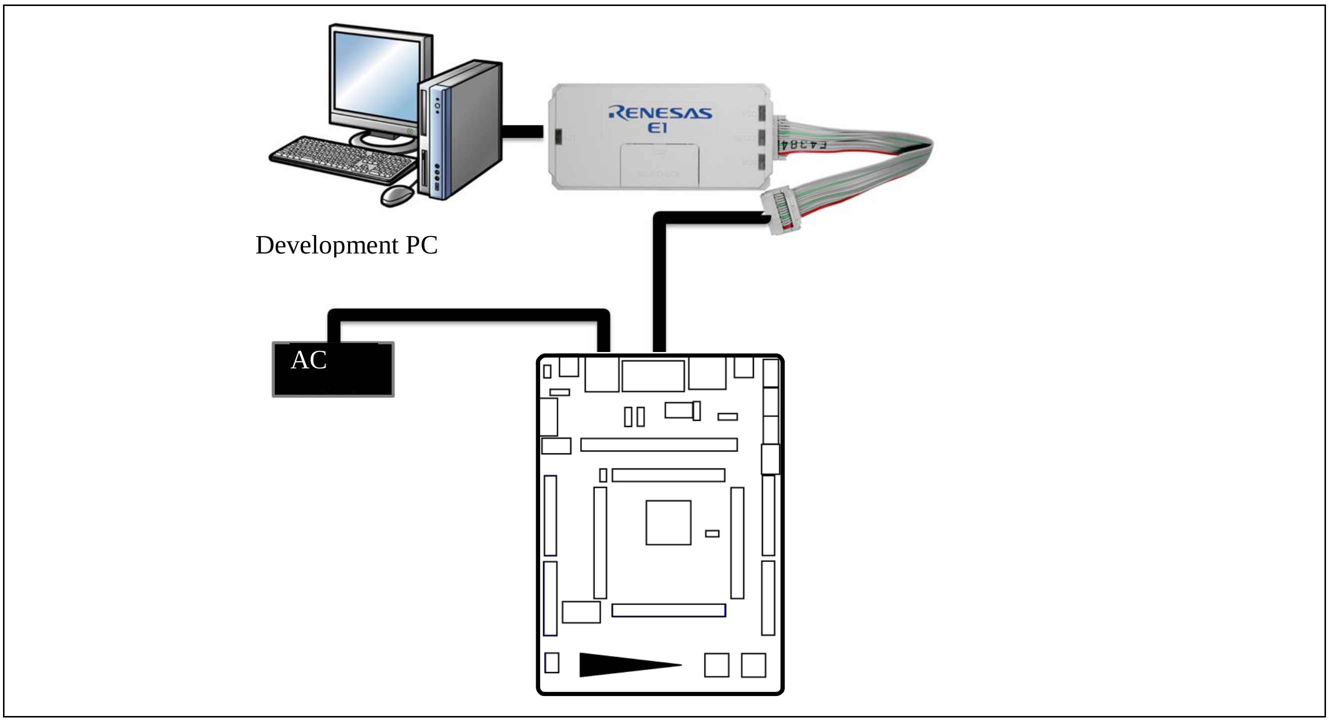
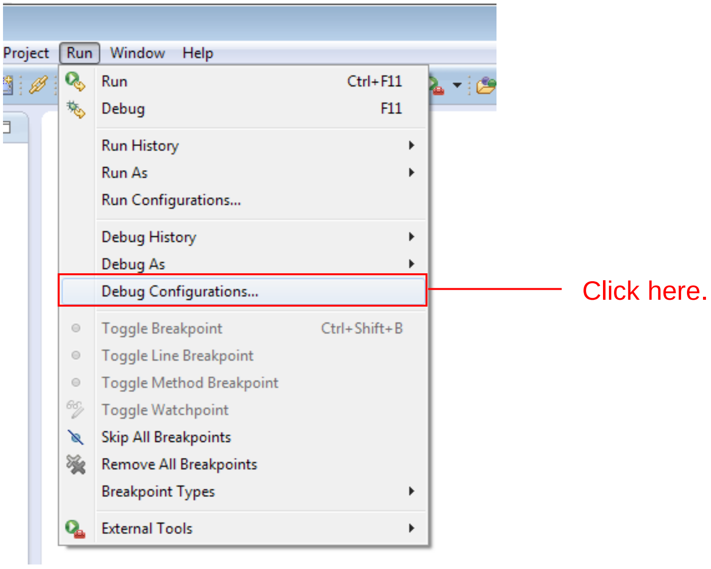
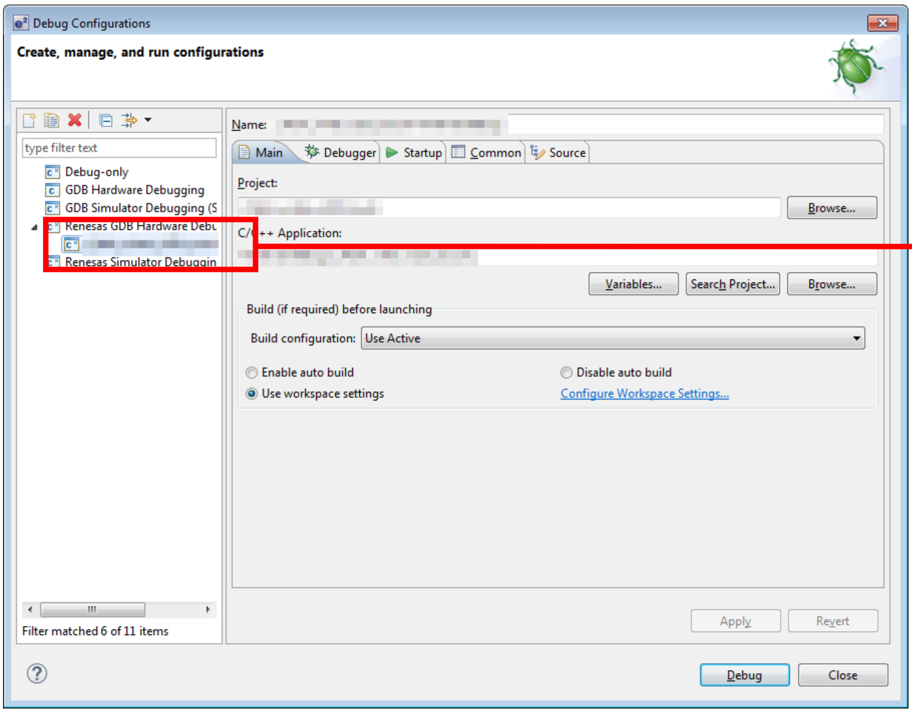


Figure 4-3 RSK Jumper Locations

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



4. Click on [r_flash_writer_rx231.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

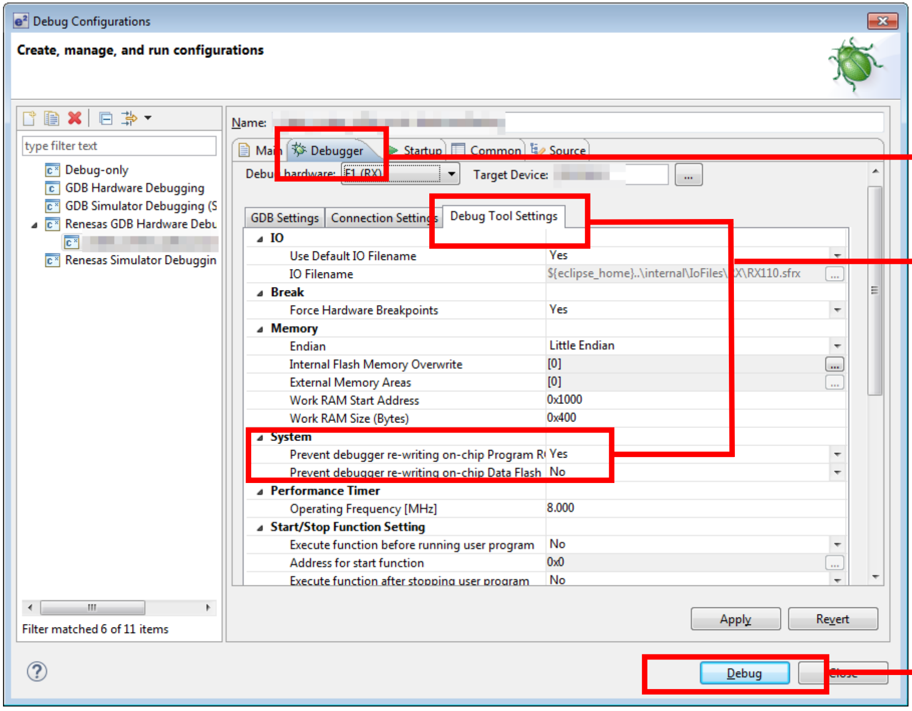
Build configuration: Use Active

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

Select [r_flash_writer_rx231.x]

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



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

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

GDB Settings | Connection Settings | Debug Tool Settings

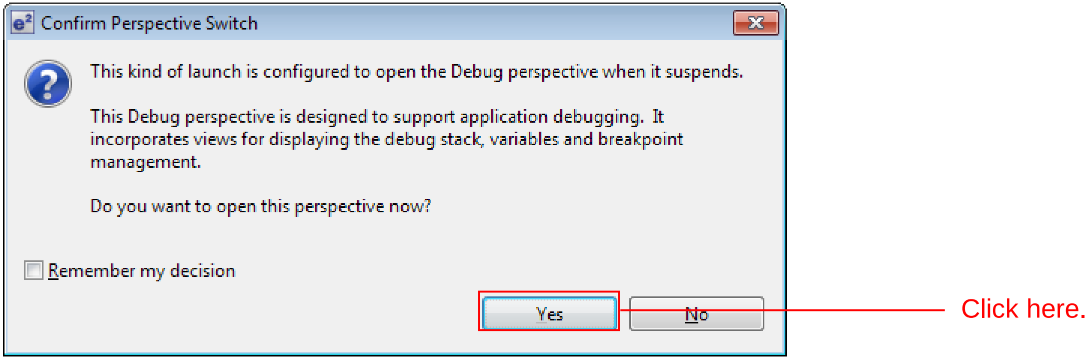
IO	
Use Default IO Filename	Yes
IO Filename	${eclipse_home}..\internal\IoFiles\RX\RX110.sfrx
Break	
Force Hardware Breakpoints	Yes
Memory	
Endian	Little Endian
Internal Flash Memory Overwrite	[0]
External Memory Areas	[0]
Work RAM Start Address	0x1000
Work RAM Size (Bytes)	0x400
System	
Prevent debugger 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

Select [Debugger]

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

Click on [Debug]

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



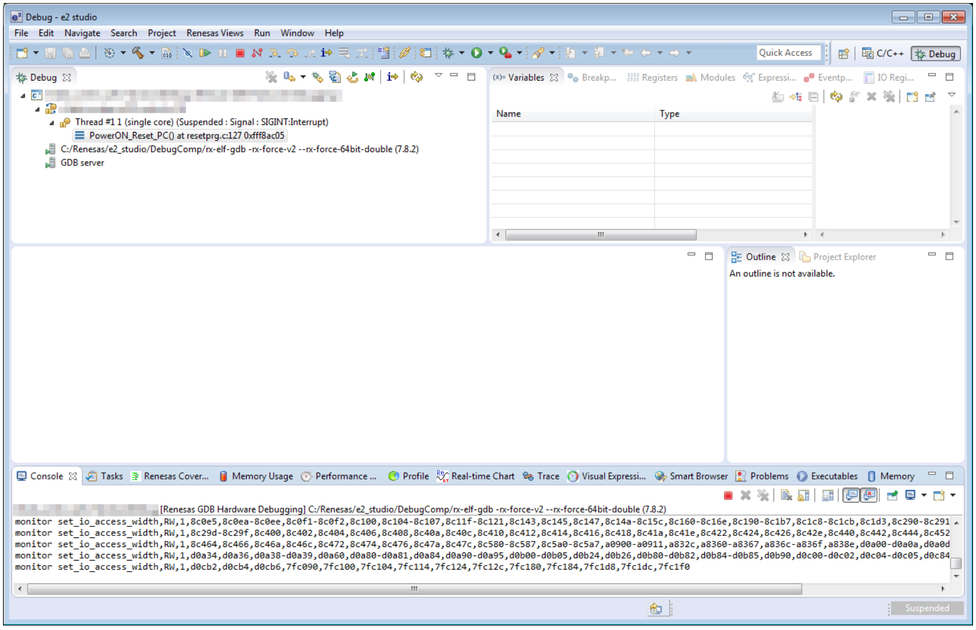
This kind of launch is configured to open the Debug perspective when it suspends.

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?

□ Remember my decision

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

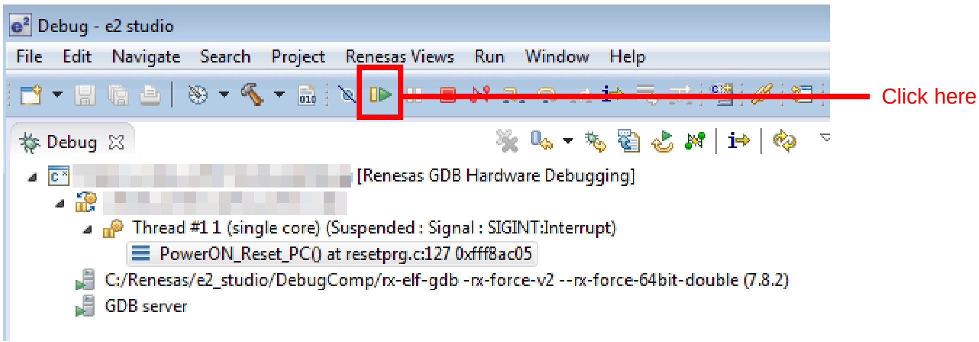


```
[redacted]
  [redacted]
    Thread #1 1 (single core) (Suspended : Signal : SIGINT:Interrupt)
      PowerON_Reset_PC() at resetprg.c:127 0xfff8ac05
  C:/Renesas/e2_studio/DebugComp/rx-elf-gdb -rx-force-v2 --rx-force-64bit-double (7.8.2)
  GDB server
```

An outline is not available.

```
[redacted] [Renesas GDB Hardware Debugging] C:/Renesas/e2_studio/DebugComp/rx-elf-gdb -rx-force-v2 --rx-force-64bit-double (7.8.2)
monitor set_io_access_width,RW,1,8c0e5,8c0ea-8c0ee,8c0f1-8c0f2,8c100,8c104-8c107,8c11f-8c121,8c143,8c145,8c147,8c14a-8c15c,8c160-8c16e,8c190-8c1b7,8c1c8-8c1cb,8c1d3,8c290-8c291
monitor set_io_access_width,RW,1,8c29d-8c29f,8c400,8c402,8c404,8c406,8c408,8c40a,8c40c,8c410,8c412,8c414,8c416,8c418,8c41a,8c41e,8c422,8c424,8c426,8c42e,8c440,8c442,8c444,8c452
monitor set_io_access_width,RW,1,8c464,8c466,8c46a,8c46c,8c472,8c474,8c476,8c47a,8c47c,8c580-8c587,8c5a0-8c5a7,a0900-a0911,a832c,a8360-a8367,a836c-a836f,a838e,d0a00-d0a0a,d0a0d
monitor set_io_access_width,RW,1,d0a34,d0a36,d0a38-d0a39,d0a60,d0a80-d0a81,d0a84,d0a90-d0a95,d0b00-d0b05,d0b24,d0b26,d0b80-d0b82,d0b84-d0b85,d0b90,d0c00-d0c02,d0c04-d0c05,d0c84
monitor set_io_access_width,RW,1,d0cb2,d0cb4,d0cb6,7fc090,7fc100,7fc104,7fc114,7fc124,7fc12c,7fc180,7fc184,7fc1d8,7fc1dc,7fc1f0
```

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



```
[redacted] [Renesas GDB Hardware Debugging]
  [redacted]
    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.
   From then on, run the main program by operating the RSK to check the information displayed on the RSK Pmod LCD.

10. When Pmod LCD displays ”MAIN”, insert RSK USB – A connector (USB0-1) to the USB memory. (Pmod LCD displays ”ATTACH”)

11. For Touch key, to move the cursor up, touch the touch key(+). To move the cursor down, touch the touch key(-). Select the sample to write, and touch the touch key (1).
    For SW, to move the cursor up, press S/W1. To move the cursor down, press S/W2. Select the sample to write and press S/W3.

12. Main program reads the data from the USB memory, and writes the read the data to on-chip flash memory (Pmod LCD displays ”START”, ”STOP”, then, ”DETACH”)

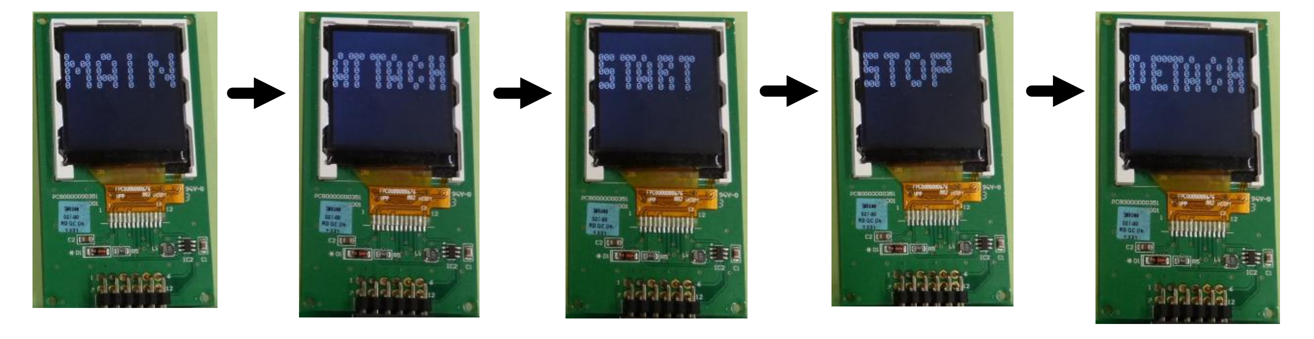


Figure 4-4 LCD display

13. RSK Pmod LCD is initialized and the sample program is displayed.
    (sample program is read from the USB memory and written to on-chip flash memory.)
    For the display contents, refer to “4.4 Verify operation of Sample program”.

4.4 Verify operation of Sample program

(1) For sample1/sample1.bin, and ”TEST!”

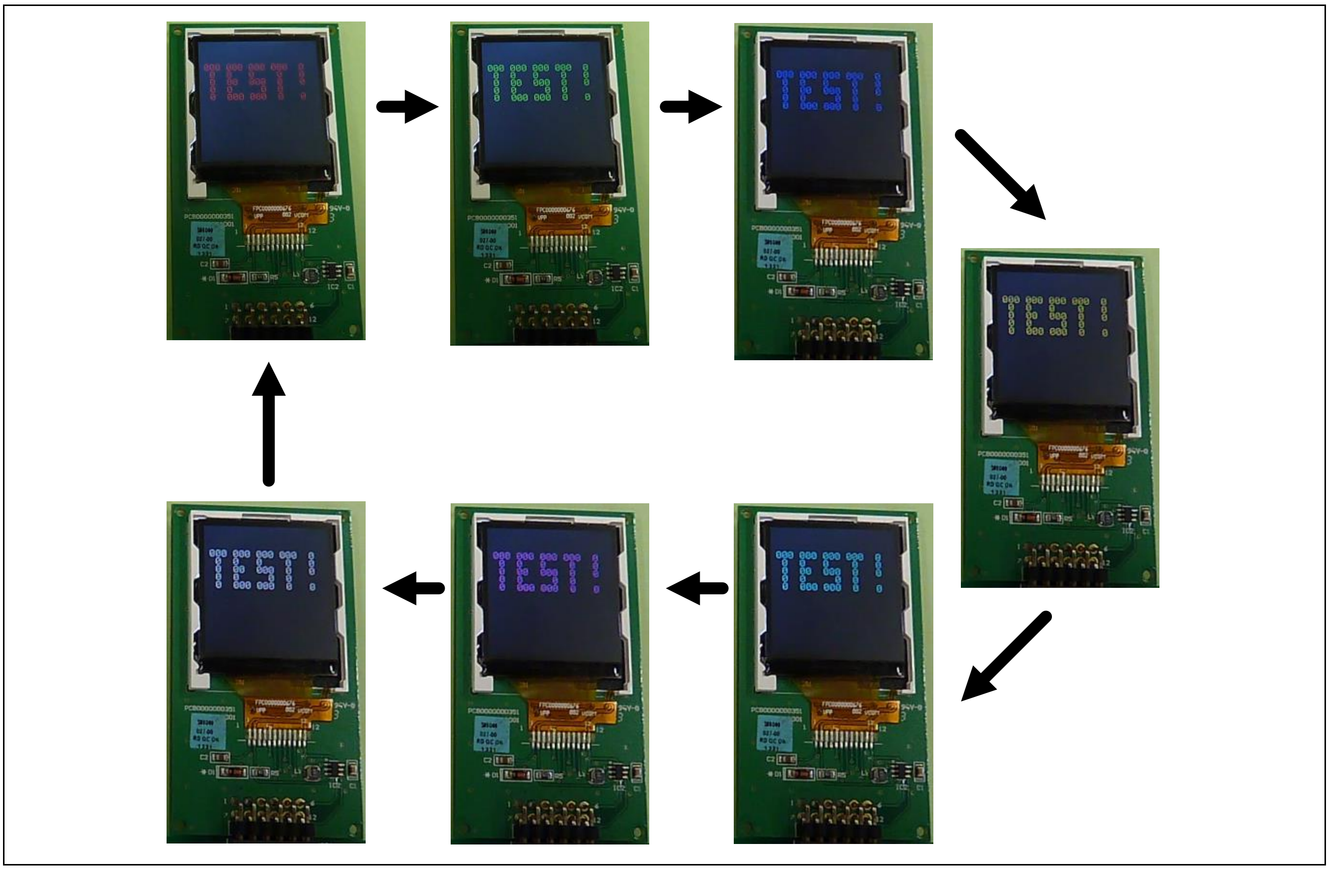


Figure 4-5 LCD display (sample1.bin)

(2) For sample2/sample2.bin, ”DEMO”

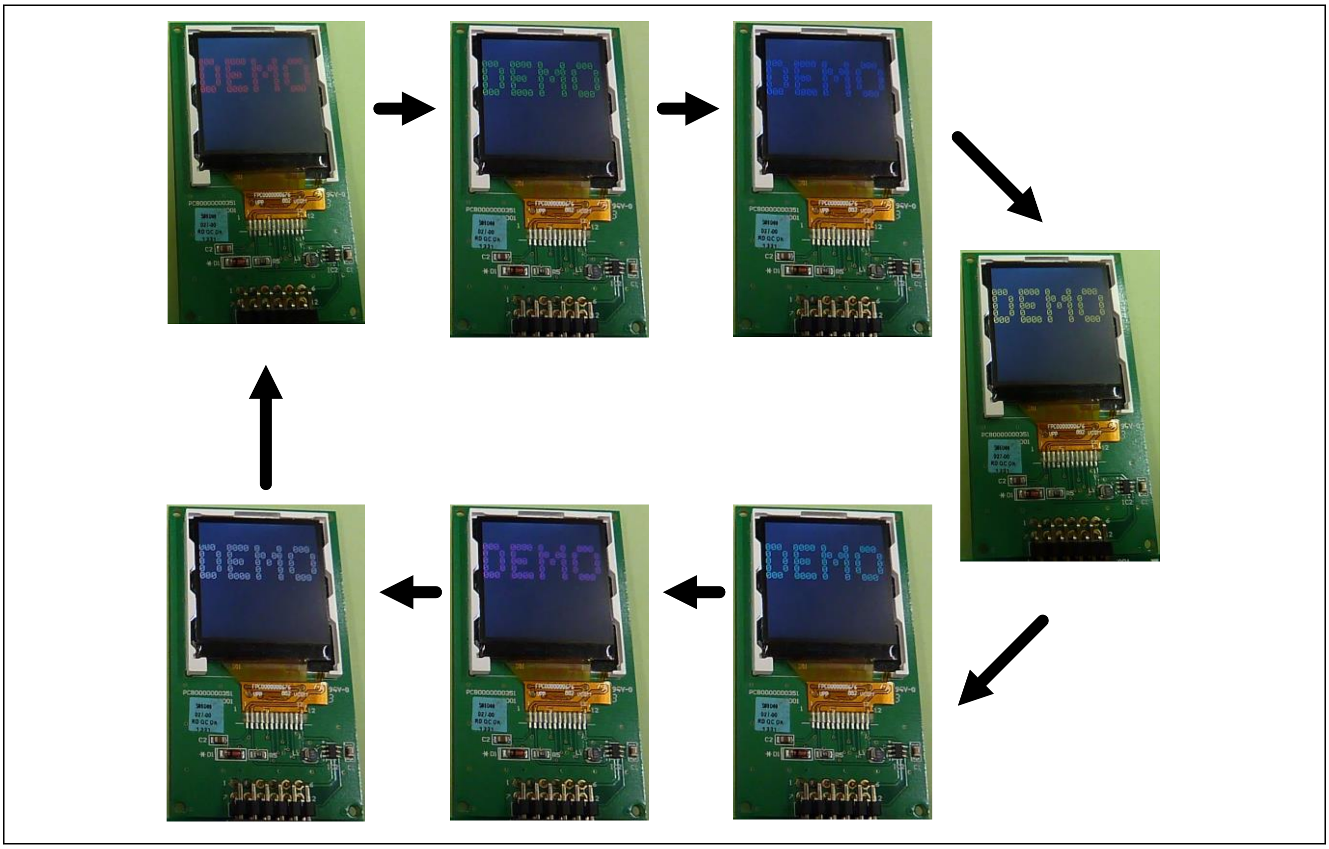


Figure 4-6 LCD display (sample2.bin)

(3) For sample3/sample3.bin, ”Touch”

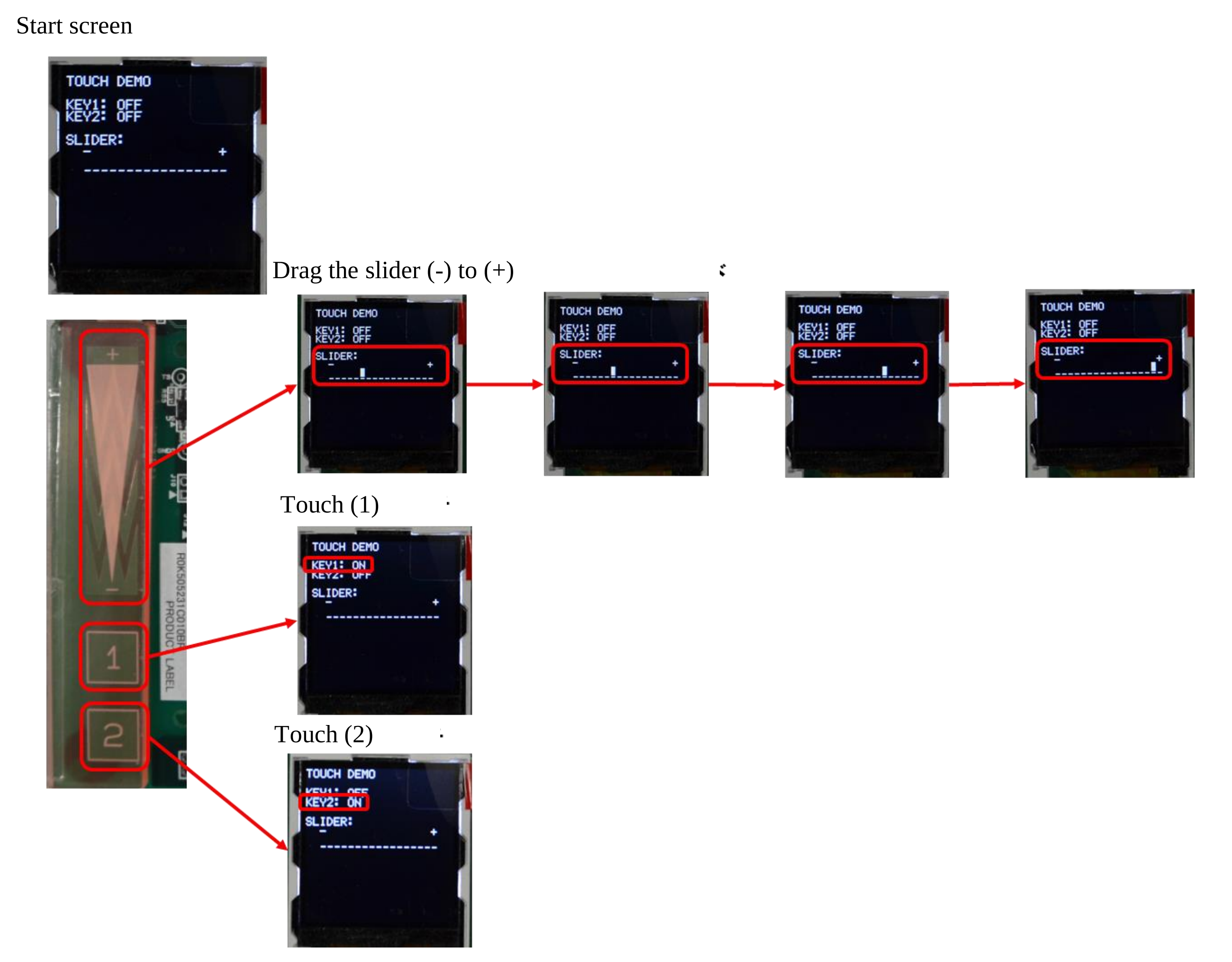


Figure 4-7 LCD display (sample3.bin)

5. Application overview

This application consists of main program and sample program.

Using the main program, binary file of the sample program in the USB memory is read and written to on-chip flash memory. When the write is completed, the sample program is run using the start-up program protection feature. When reset after that, an original program starts.

The start-up program protection feature can select either Retain or Do not retain the reset vector information switching after reset. In this application, Do not retain is selected. Therefore, information change of reset vector returns to the state that the original main program starts when reset. For the details, refer to "Start-up Program Protection feature" section of user's manual (hardware).

Figure 5-1 shows the transition of the application status.

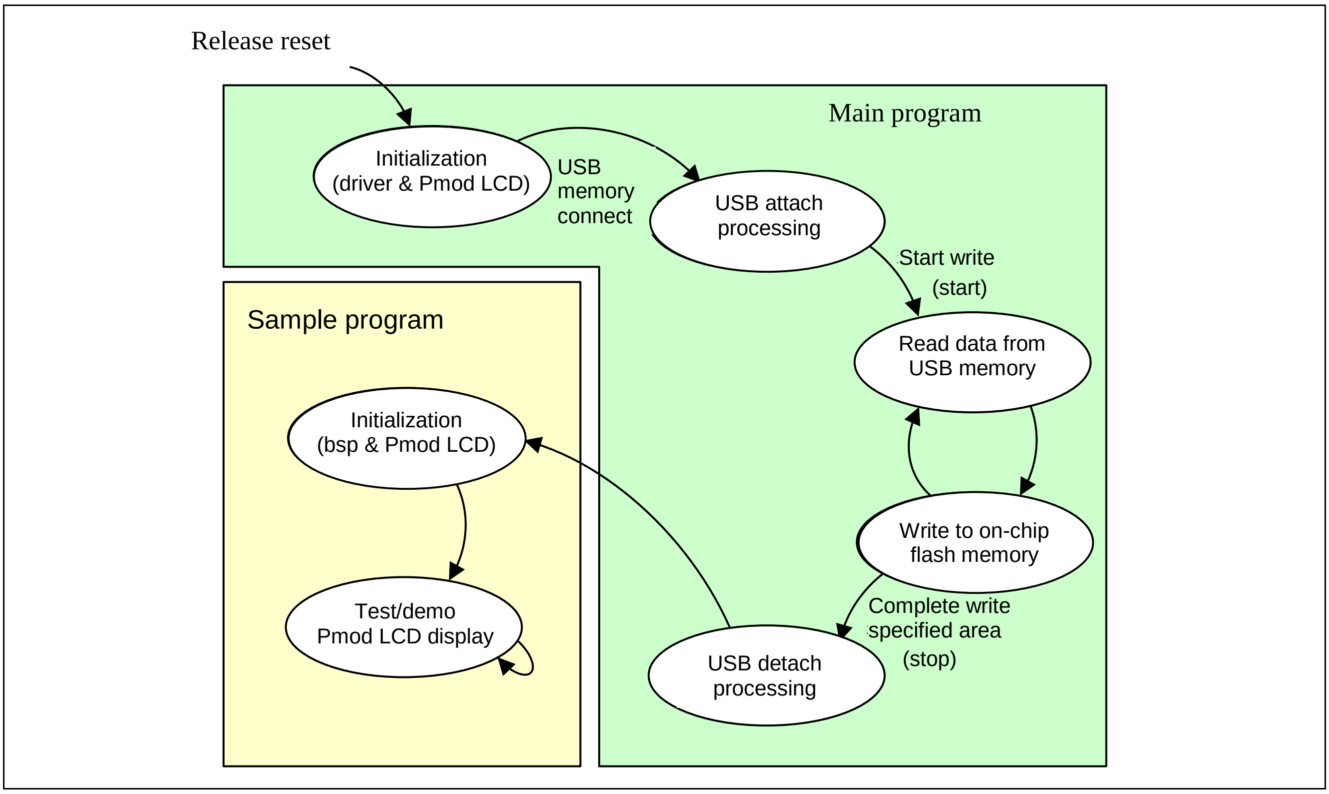


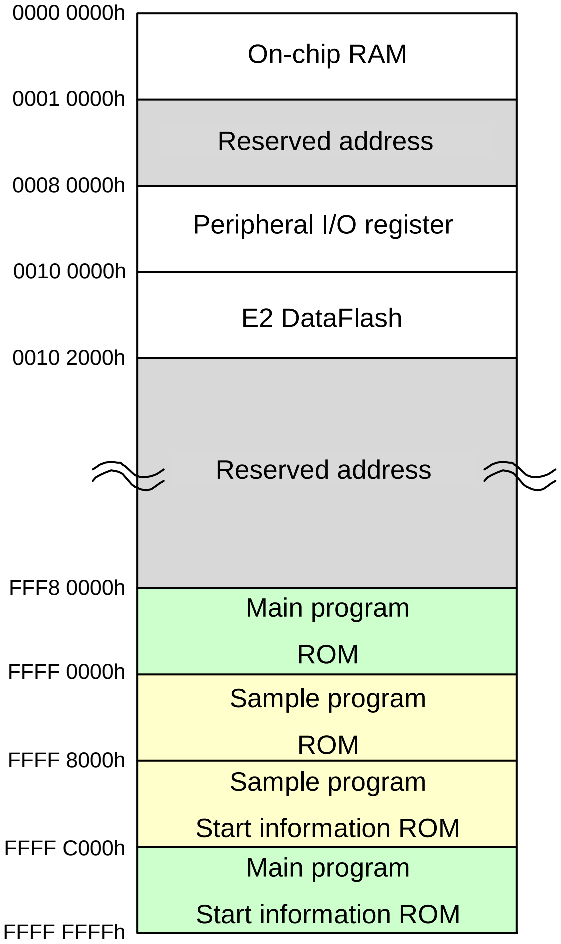
Figure 5-1 Application status

(1) Initialize each driver, and Pmod LCD
(2) Connect USB by using the USB driver
(3) Read the data (Note 1) from the USB memory by using the USB driver and FAT file system
(4) Write the data (Note 1) to on-chip flash memory by using the Flash API.
(5) After disconnecting the USB, switch the start-up program by using the Flash API
(6) Jump to the written data (Note 1)
--- This concludes the execution of the main program ---
(7) Initialize BSP and Pmod LCD
(8) Display the data on Pmod LCD

Note 1. Refers to Sample program execution binary file.

5.1 Memory structure

It shows RX231 MCU memory map on the RSKRX231 used in this application.



Note 1 : The memory structure is for RX231 MCU in the RSKRX231.
Section address is different depending on the CPU model and main program size used.
Also, modification is required depending on the user system.

Figure 5-2 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_rx231”. 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 Name	File Name	Description
Src	main.c	Main processing
	main.h	Application setting header file (Setting for the read file name and the write area)
	r_usb_hmsc_apl.c	USB main program processing
	r_usb_hmsc_apl.h	r_usb_hmsc_apl.c header file
	r_rsk_flashdriver.c	Flash memory program processing
	r_rsk_flashdriver.h	r_rsk_flashdriver.c header file
	r_rsk_leddriver.c	Program for LED output
	r_rsk_leddriver.h	r_rsk_leddriver.c header file
	lcd.c, ascii.c, r_cg_sci.c, r_cg_sci_user.c	Program for Pmod LCD display
	lcd.h, ascii.h, r_cg_sci.h, r_cg_macrodriver.h	header file of the program for Pmod LCD display
	r_rsk_keydrive.c	Program for Push S/W output
	r_rsk_keydriver.h	r_rsk_keydriver.c header file

Table 6-2 FIT modules used

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

6.2 Modules

The following table lists the modules.

Table 6-3 Modules

File Name	Module Name	Description
main.c	Main	Main processing
	usb_mcu_init	Port initialization of USB
	usb_board_init	Initialization of LCD, and USB interrupt enable setting
	apl_init	Initialization of Application management table
	jmp_user_program	Jump to user’s program
r_usb_hmsc_apl.c	usb_driver_init	USB driver initialization
	msc_registration	USB callback function registration
	usb_hmsc_driver	USB HMSC driver task processing
	msc_connect_wait	Wait for USB device detection
	msc_drive	USB device connection, and TFAT file system mount
	msc_data_ready	Setting before read state
	msc_data_read	Data read from USB memory, and data write to flash memory
	msc_detach_device	USB disconnection
	usb_hsmpl_device_state	USB driver callback processing
	msc_configured	USB device detection notice
	msc_drive_complete	USB device connection notice
	msc_detach	USB device disconnection notice
	msc_event_set	Event setting processing
	msc_event_get	Event acquisition processing
r_rsk_flashdriver.c	SAMPLE_FLASH_Write	Flash memory rewrite processing
lcd.c (Note)	Init_LCD	Pmod LCD initialization
	Display_LCD	Pmod LCD display processing
	DisplaySetFontColour	Pmod LCD font color change processing
r_rsk_leddriver.c (Note)	usb_cpu_LedInitial	Initialization of LEDs
	usb_cpu_led_set_data	LED display processing
r_rsk_keydrive.c	usb_cpu_key_read_touch	Touch sensor read processing
	usb_cpu_key_read	S/W read processing
	usb_cpu_sw_data	Discrimination processing for file selection operation
	usb_cpu_sw1_data	Discrimination processing for operation of file selection ”Up”
	usb_cpu_sw2_data	Discrimination processing for operation of file selection ”Down”
	usb_cpu_sw3_data	Discrimination processing for operation of file selection ”Enter”
TouchAPI	-	Touch-related processing Workbench6 output file

(Note) For the LED/LCD processing, refer to Application note for Renesas Starter Kit Sample code.

6.3 Flowcharts

(1) Main processing

Figure 6-1 &Figure 6-2 show the flowcharts for the main processing.

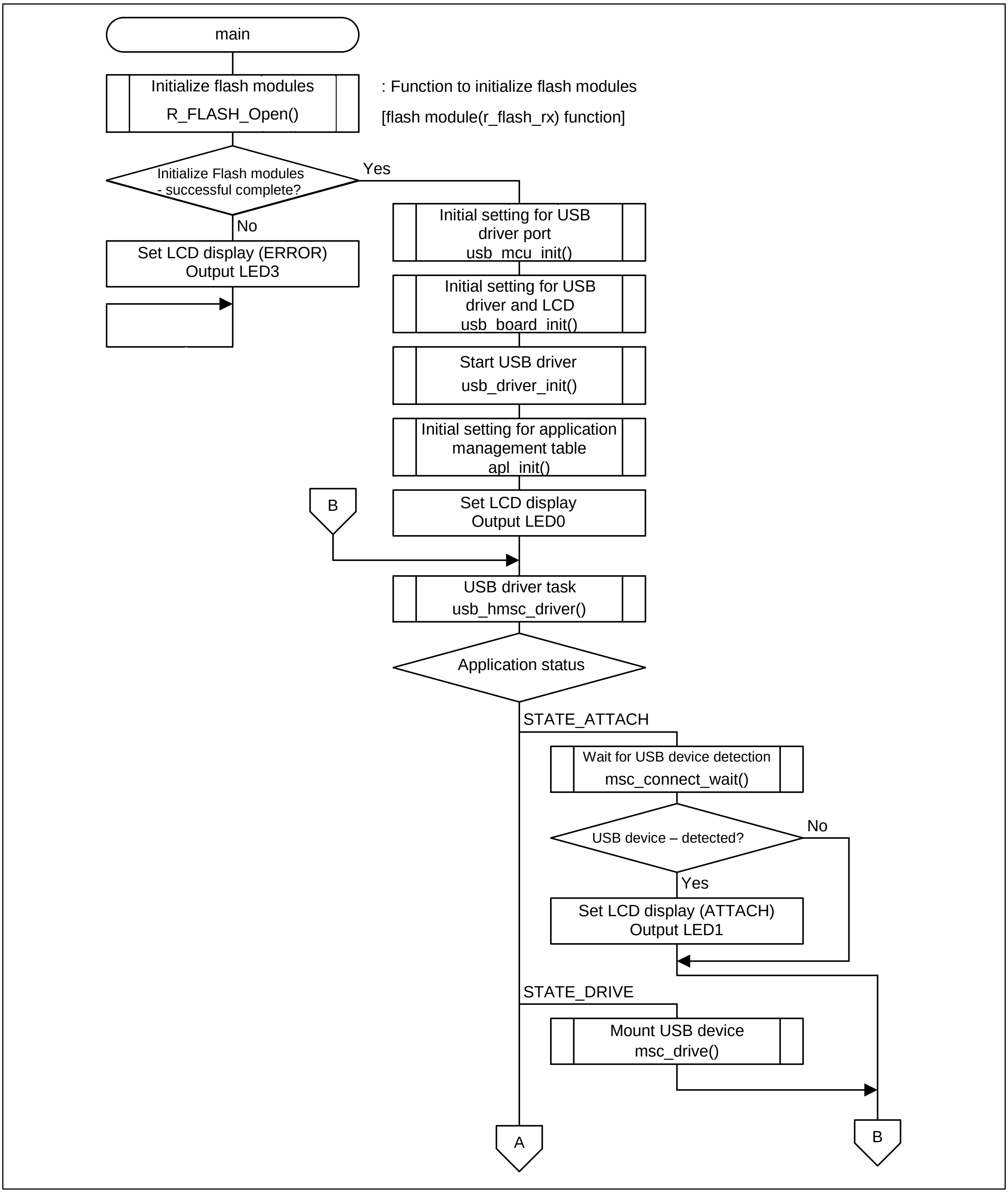


Figure 6-1 Main processing (1)

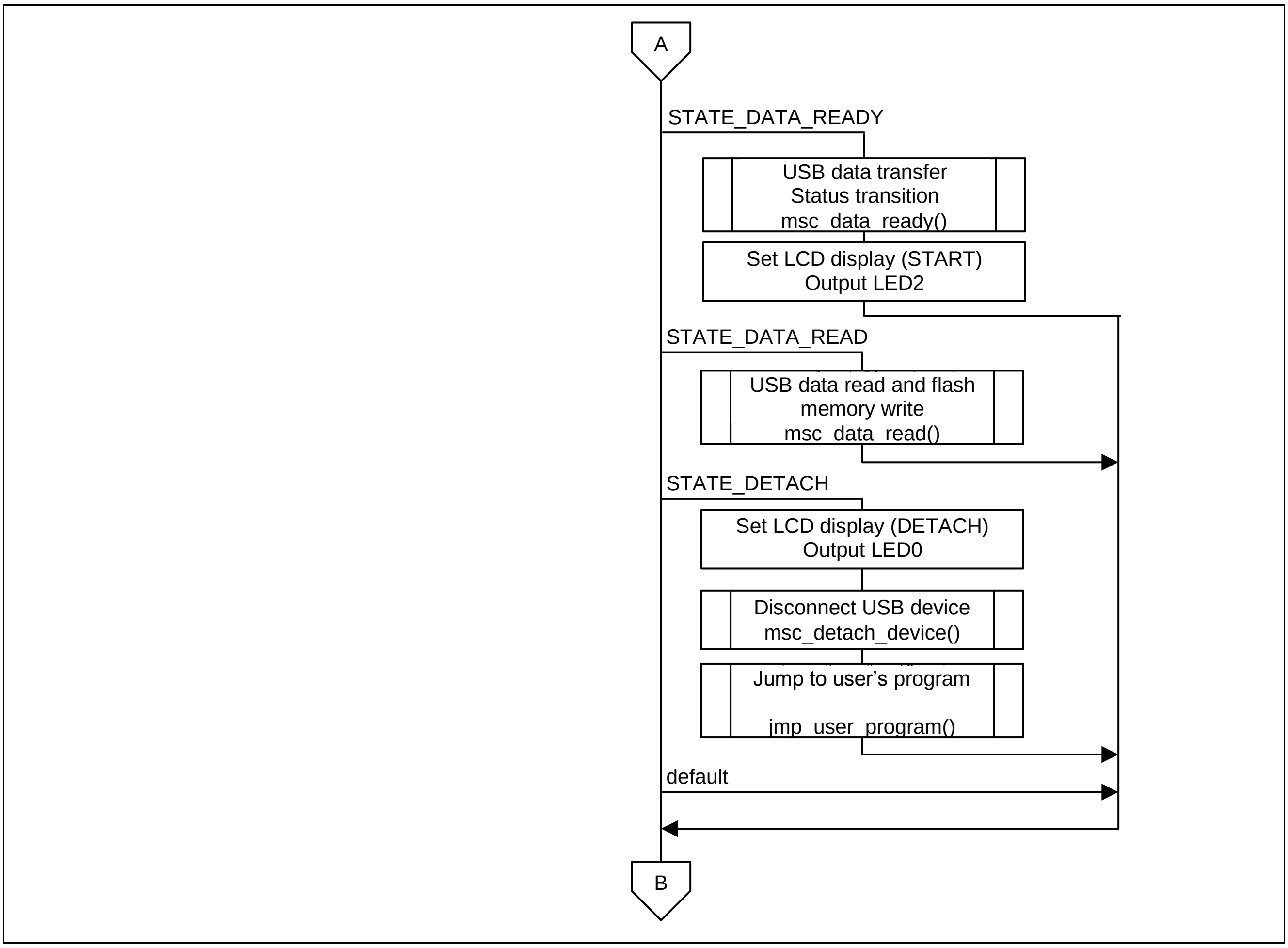


Figure 6-2 Main processing (2)

(2) Port initialization of USB

Figure 6-3shows the flowchart for Port initialization of USB

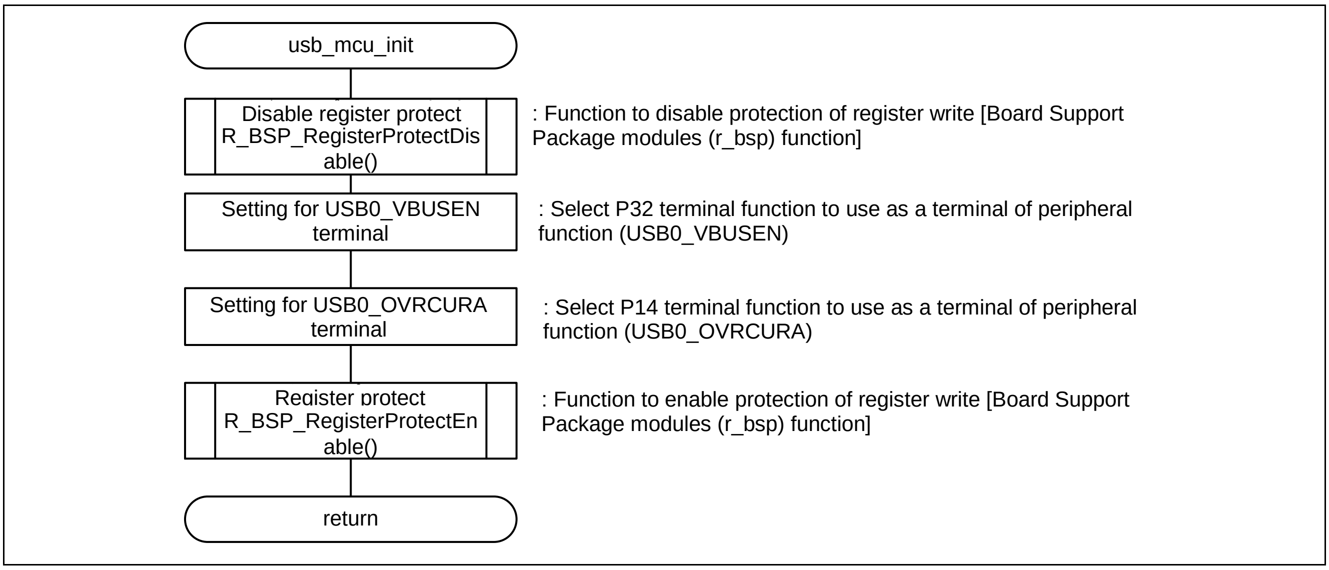

usb_mcu_init
Disable register protect
R_BSP_RegisterProtectDisable()
: Function to disable protection of register write [Board Support Package modules (r_bsp) function]
Setting for USB0_VBUSEN terminal
: Select P32 terminal function to use as a terminal of peripheral function (USB0_VBUSEN)
Setting for USB0_OVRCURA terminal
: Select P14 terminal function to use as a terminal of peripheral function (USB0_OVRCURA)
Register protect
R_BSP_RegisterProtectEnable()
: Function to enable protection of register write [Board Support Package modules (r_bsp) function]
return

Figure 6-3 Port initialization of USB

(3) Initialization of LCD, and USB interrupt enable setting

Figure 6-4 shows the flowchart of Initialization of LCD, and USB interrupt enable setting.

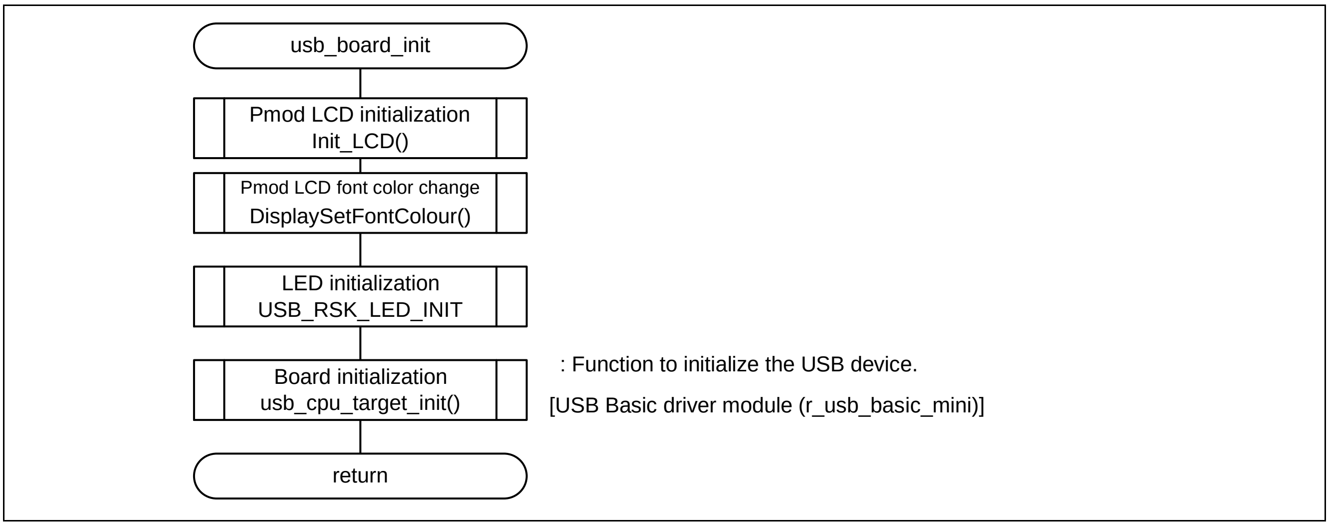


Figure 6-4 Initialization of LED andLCD, and USB interrupt enable setting

(4) Initialization of Application management table

Figure 6-5 shows the flowchart of Initialization of Application management table.

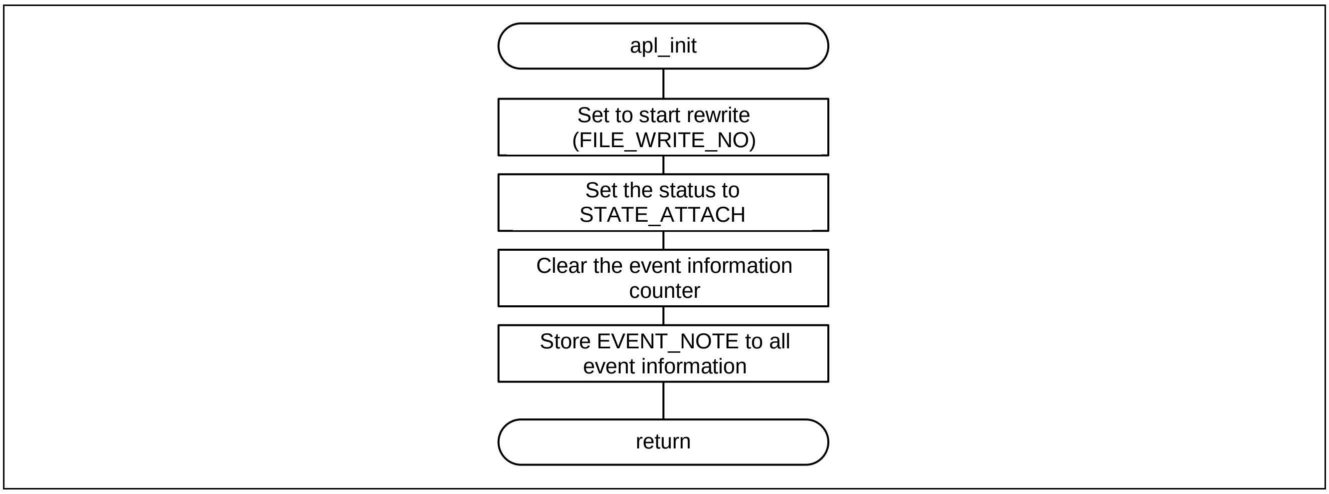


Figure 6-5 Initialization of Application management table

(5) Jump to user's program

Figure 6-6 shows the flowchart of jump to user's program

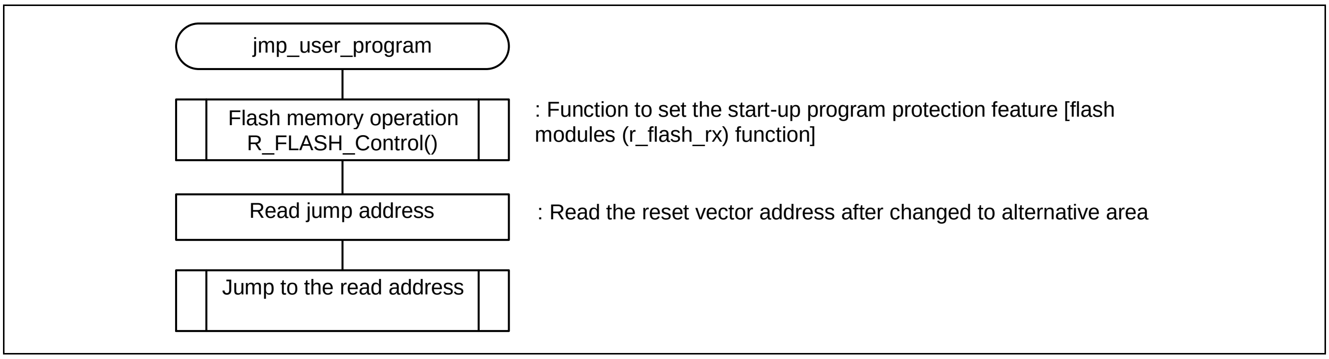

jmp_user_program
Flash memory operation
R_FLASH_Control()
: Function to set the start-up program protection feature [flash modules (r_flash_rx) function]
Read jump address
: Read the reset vector address after changed to alternative area
Jump to the read address

Figure 6-6 Jump to user's program

(6) USB driver initialization

Figure 6.7 shows the flowchart of USB driver initialization .

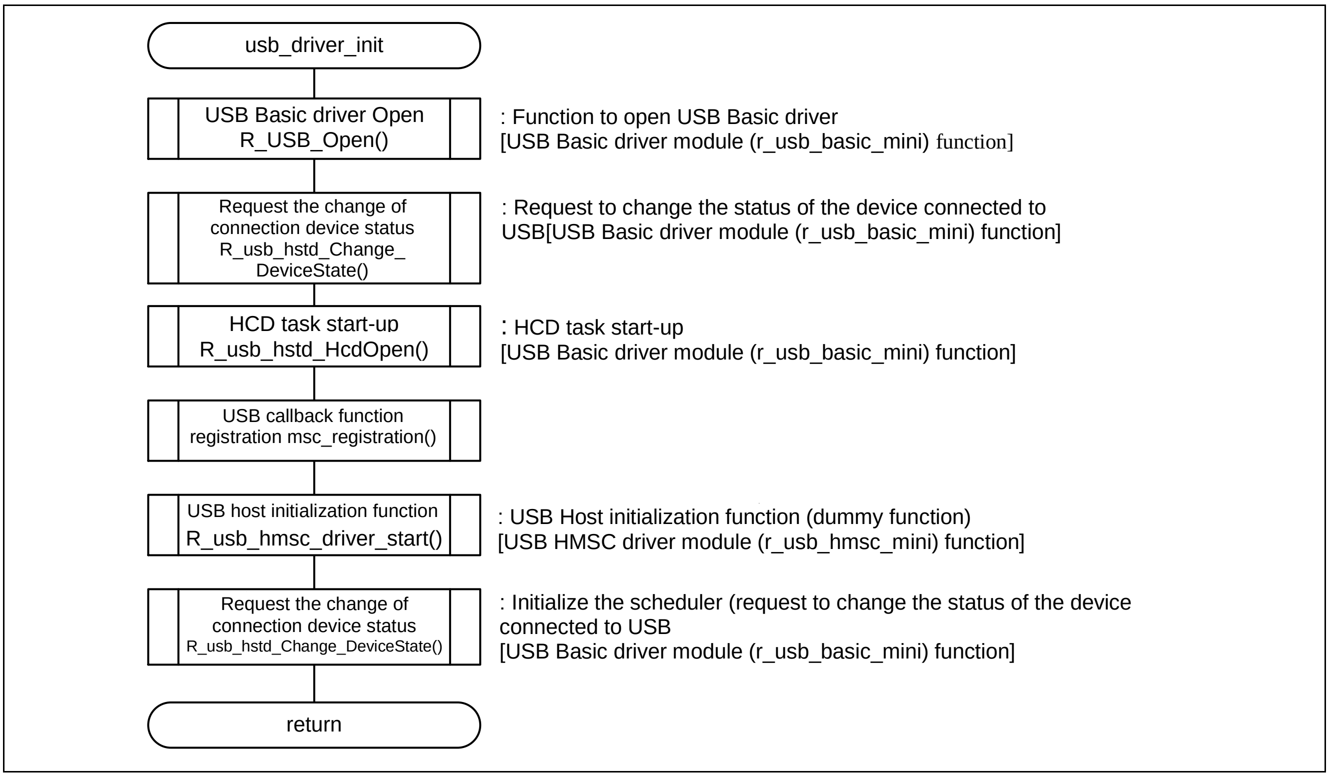

usb_driver_init
USB Basic driver Open
R_USB_Open()
: Function to open USB Basic driver
[USB Basic driver module (r_usb_basic_mini) function]
Request the change of connection device status
R_usb_hstd_Change_
DeviceState()
: Request to change the status of the device connected to USB[USB Basic driver module (r_usb_basic_mini) function]
HCD task start-up
R_usb_hstd_HcdOpen()
: HCD task start-up
[USB Basic driver module (r_usb_basic_mini) function]
USB callback function registration msc_registration()
USB host initialization function
R_usb_hmsc_driver_start()
: USB Host initialization function (dummy function)
[USB HMSC driver module (r_usb_hmsc_mini) function]
Request the change of connection device status
R_usb_hstd_Change_DeviceState()
: Initialize the scheduler (request to change the status of the device connected to USB
[USB Basic driver module (r_usb_basic_mini) function]
return

Figure 6.7 USB driver initialization

(7) USB callback function registration

Figure 6-8 shows the flowchart of USB callback function registration.

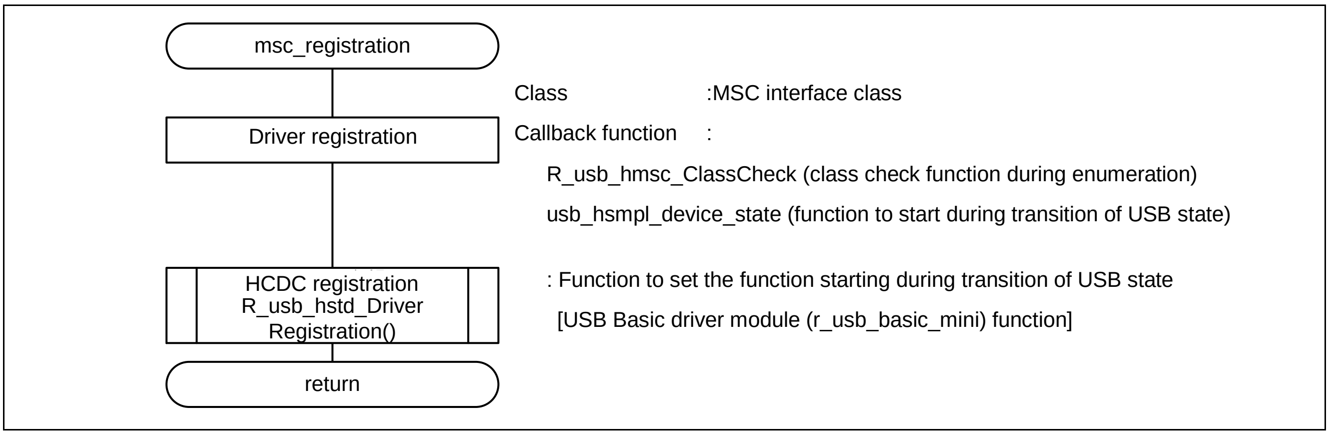

msc_registration
Driver registration
HCDC registration
R_usb_hstd_Driver
Registration()
return
Class :MSC interface class
Callback function :
R_usb_hmsc_ClassCheck (class check function during enumeration)
usb_hsmpl_device_state (function to start during transition of USB state)
: Function to set the function starting during transition of USB state
[USB Basic driver module (r_usb_basic_mini) function]

Figure 6-8 USB callback function registration

(8) USB HMSC Driver task

Figure 6-9 shows the flowchart of USB HMSC Driver task.

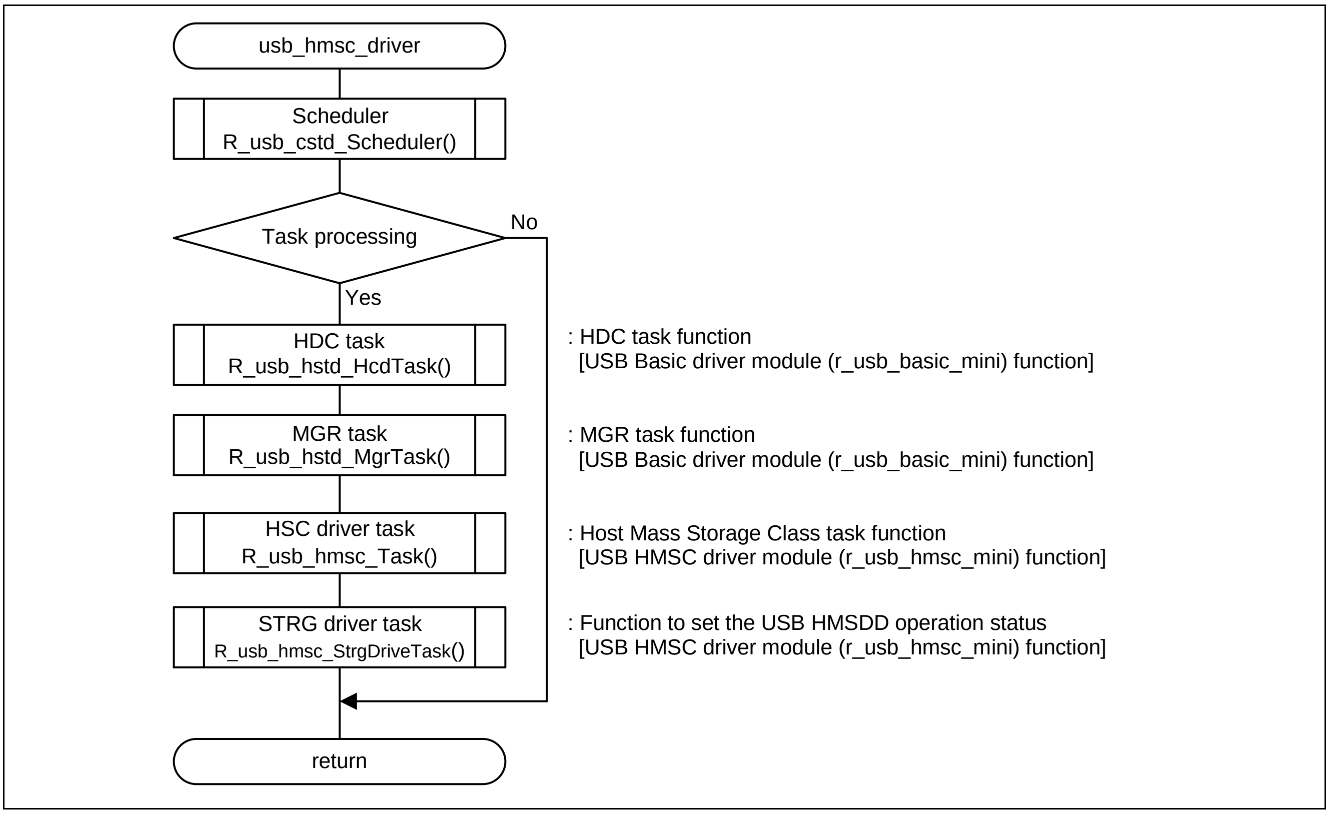

usb_hmsc_driver
Scheduler
R_usb_cstd_Scheduler()
Task processing
No
Yes
HDC task
R_usb_hstd_HcdTask()
MGR task
R_usb_hstd_MgrTask()
HSC driver task
R_usb_hmsc_Task()
STRG driver task
R_usb_hmsc_StrgDriveTask()
return
: HDC task function
[USB Basic driver module (r_usb_basic_mini) function]
: MGR task function
[USB Basic driver module (r_usb_basic_mini) function]
: Host Mass Storage Class task function
[USB HMSC driver module (r_usb_hmsc_mini) function]
: Function to set the USB HMSDD operation status
[USB HMSC driver module (r_usb_hmsc_mini) function]

Figure 6-9 USB HMSC Driver task

(9) Wait for USB device detection

Figure 6-10 shows the flowchart of Wait processing for USB device detection

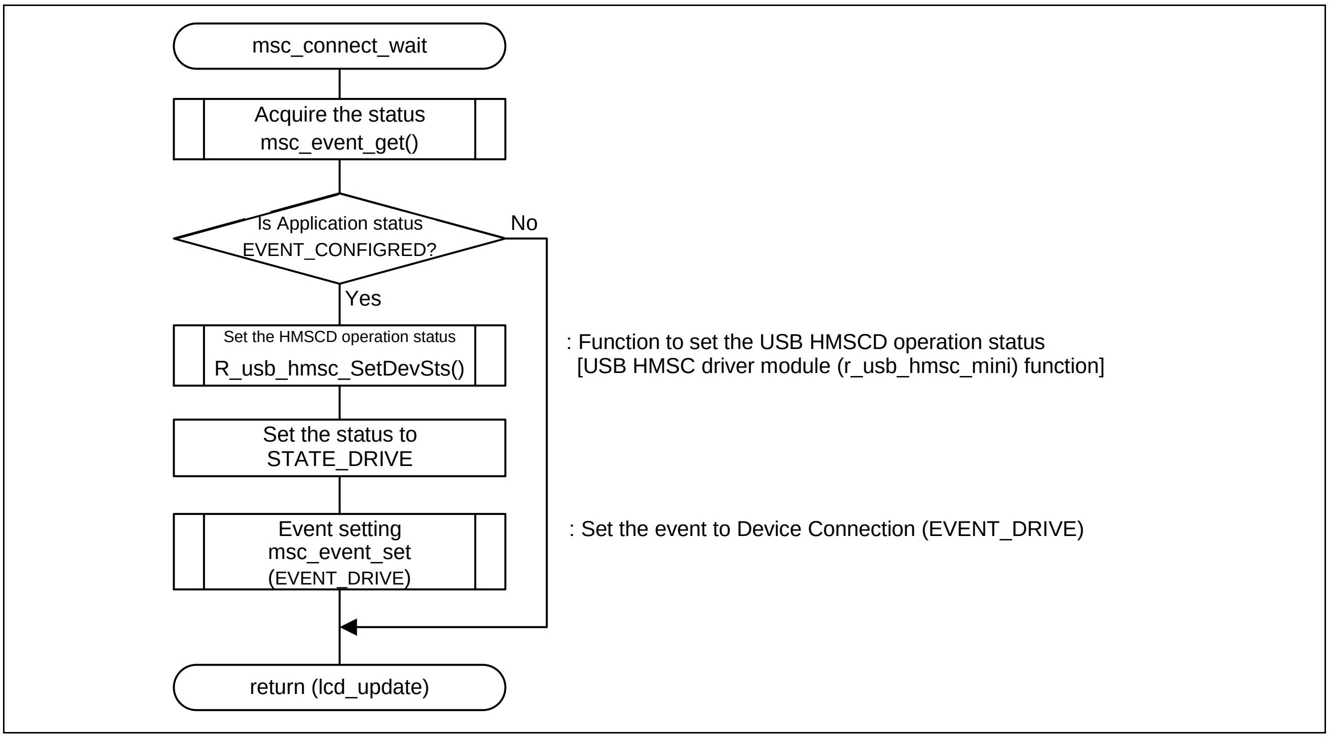


Figure 6-10 Wait for USB device detection

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

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

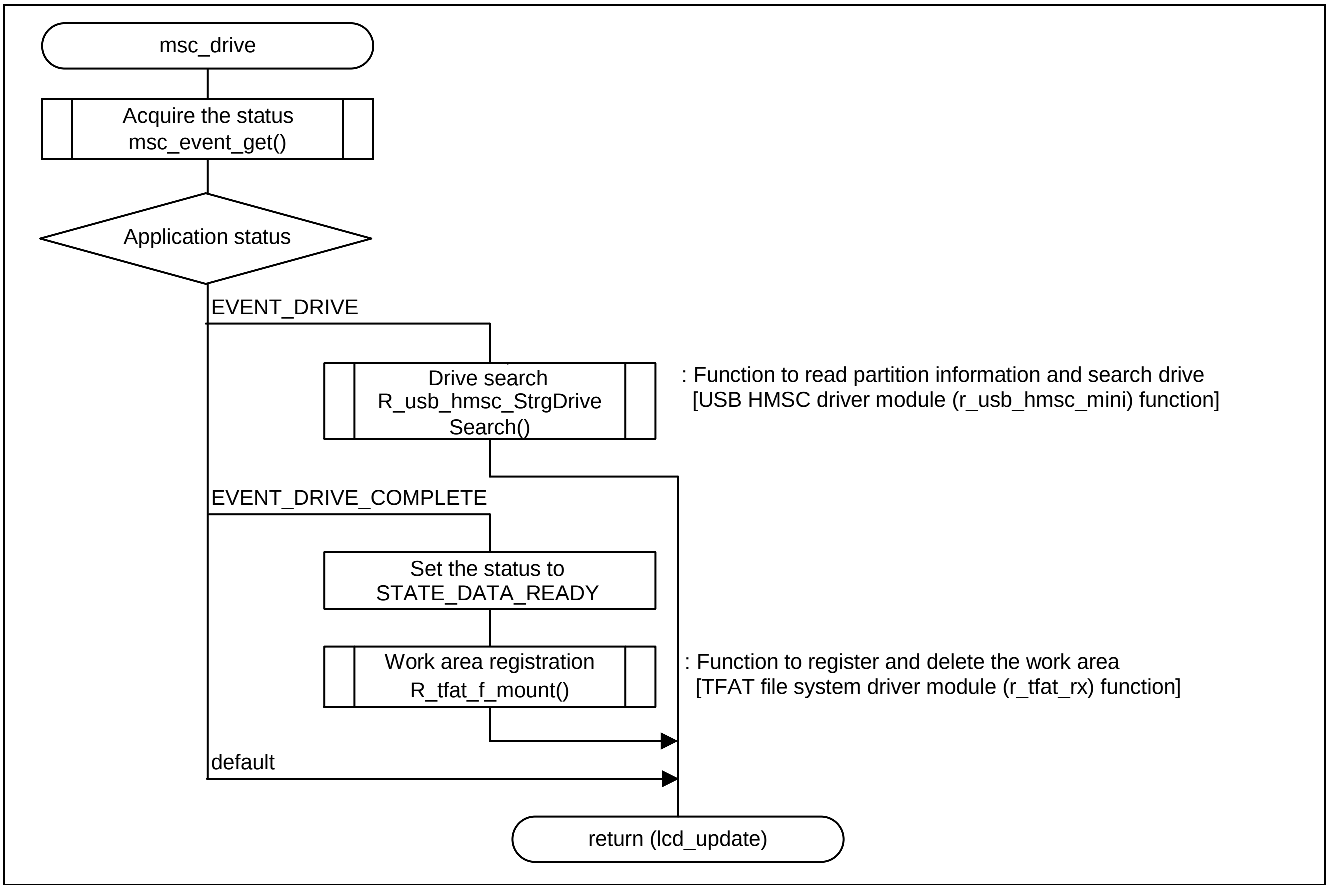

msc_drive
Acquire the status
msc_event_get()
Application status
EVENT_DRIVE
Drive search
R_usb_hmsc_StrgDrive Search()
: Function to read partition information and search drive
[USB HMSC driver module (r_usb_hmsc_mini) function]
EVENT_DRIVE_COMPLETE
Set the status to
STATE_DATA_READY
Work area registration
R_tfat_f_mount()
: Function to register and delete the work area
[TFAT file system driver module (r_tfat_rx) function]
default
return (lcd_update)

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

(11) Setting before read state

Figure 6-12 shows the flowchart of Setting processing before read state.

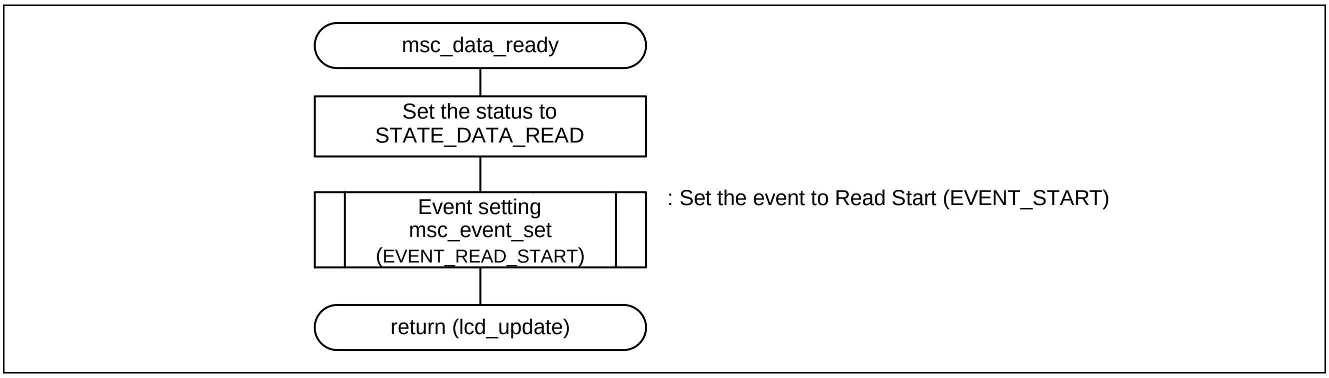


Figure 6-12 Setting before read state

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

Figure 6-13 and Figure 6-14 show the flowchart of Data read from USB memory, and data write to flash memory.

The file mane stored in the USB memory, and the start address of rewrite are set by this function.

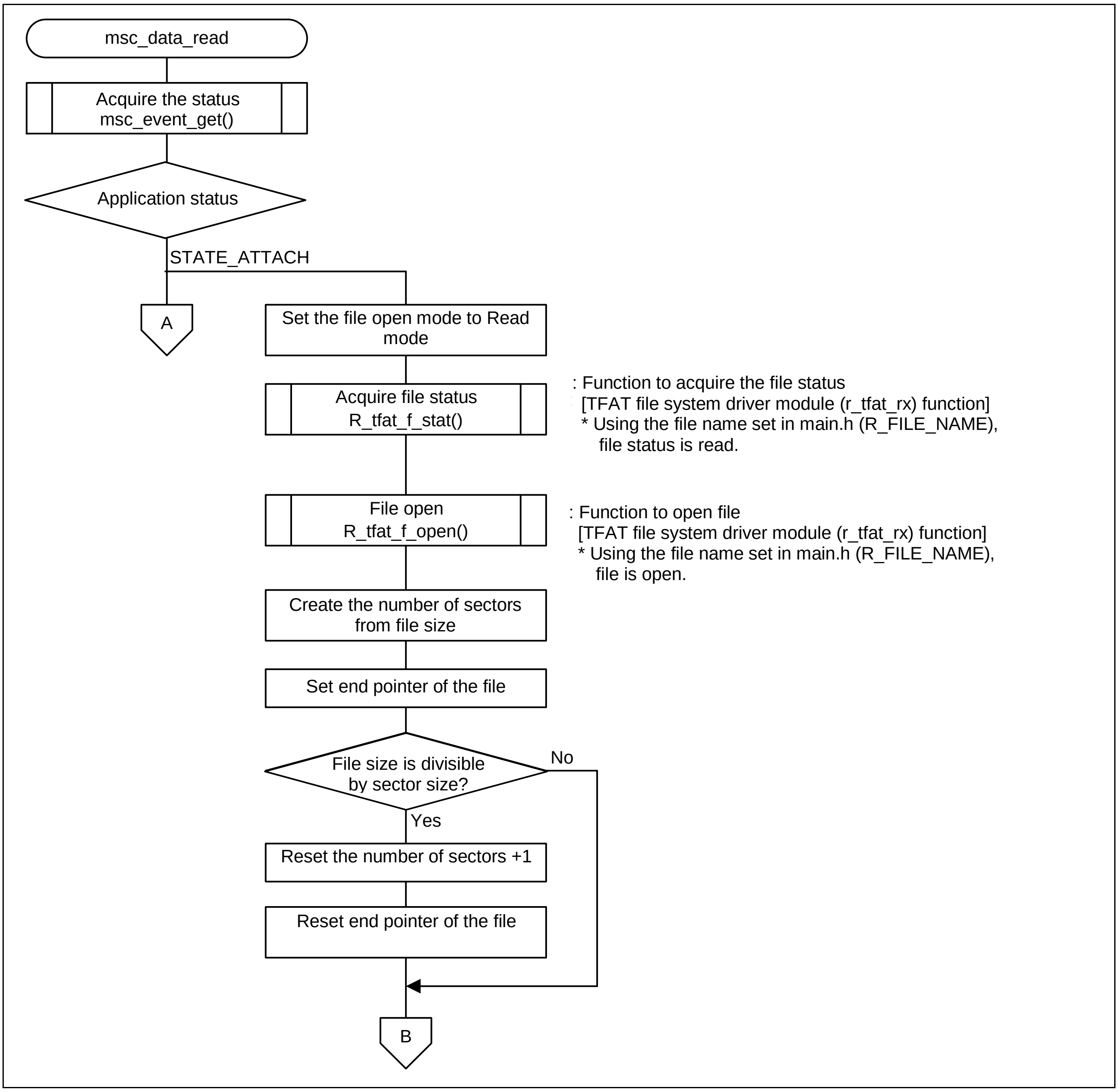

msc_data_read
Acquire the status
msc_event_get()
Application status
STATE_ATTACH
A
Set the file open mode to Read mode
Acquire file status
R_tfat_f_stat()
: Function to acquire the file status
[TFAT file system driver module (r_tfat_rx) function]
* Using the file name set in main.h (R_FILE_NAME), file status is read.
File open
R_tfat_f_open()
: Function to open file
[TFAT file system driver module (r_tfat_rx) function]
* Using the file name set in main.h (R_FILE_NAME), file is open.
Create the number of sectors from file size
Set end pointer of the file
File size is divisible by sector size?
No
Yes
Reset the number of sectors +1
Reset end pointer of the file
B

Figure 6-13 Data read from USB memory, and data write to flash memory(1)

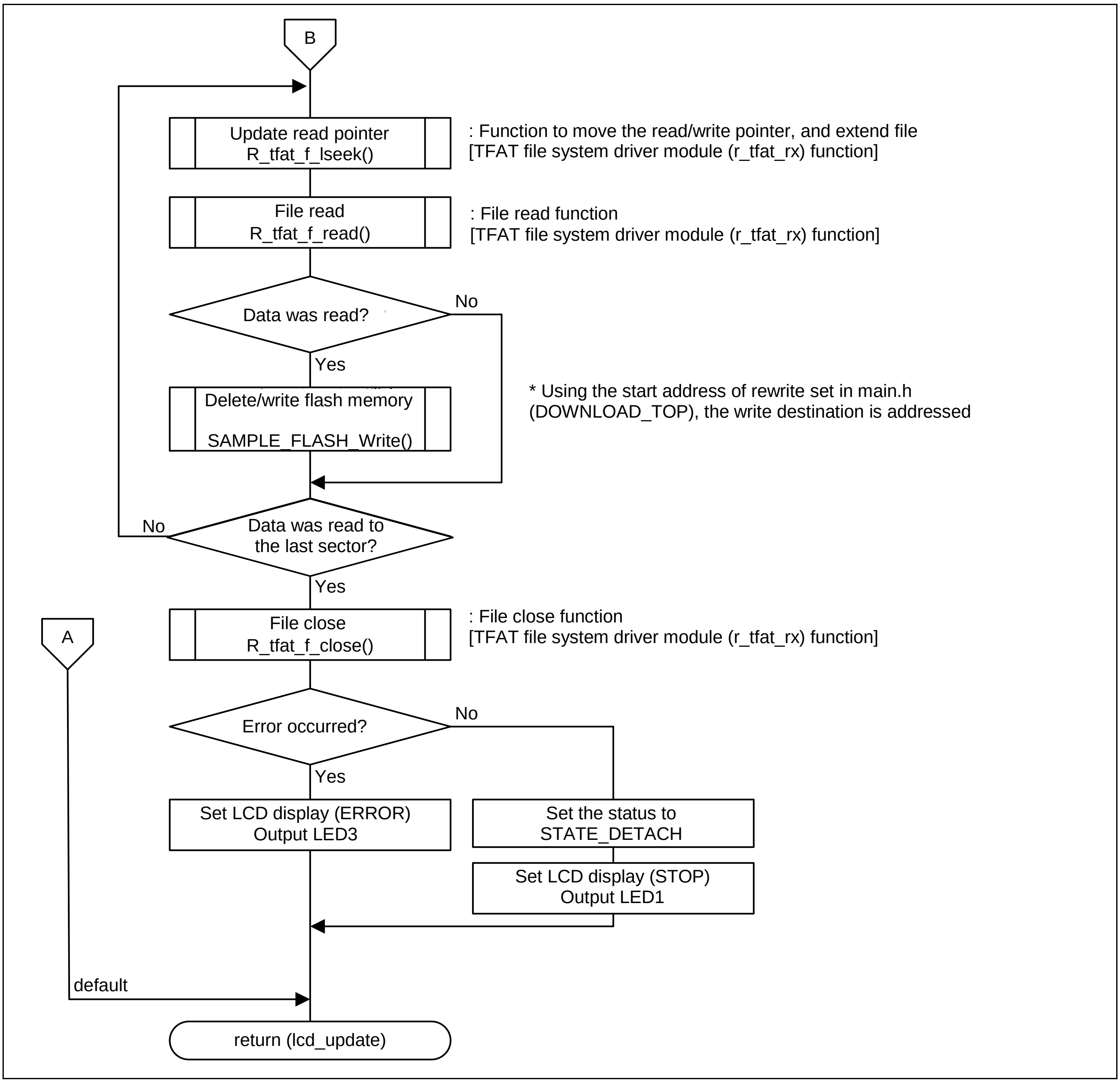

B
Update read pointer
R_tfat_f_lseek()
: Function to move the read/write pointer, and extend file
[TFAT file system driver module (r_tfat_rx) function]
File read
R_tfat_f_read()
: File read function
[TFAT file system driver module (r_tfat_rx) function]
Data was read?
No
Yes
Delete/write flash memory
SAMPLE_FLASH_Write()
* Using the start address of rewrite set in main.h (DOWNLOAD_TOP), the write destination is addressed
No
Data was read to the last sector?
Yes
A
File close
R_tfat_f_close()
: File close function
[TFAT file system driver module (r_tfat_rx) function]
Error occurred?
No
Yes
Set LCD display (ERROR)
Output LED3
Set the status to STATE_DETACH
Set LCD display (STOP)
Output LED1
default
return (lcd_update)

Figure 6-14 Data read from USB memory, and data write to flash memory(2)

(13) USB disconnection

Figure 6-15 shows the flowchart of USB disconnection.

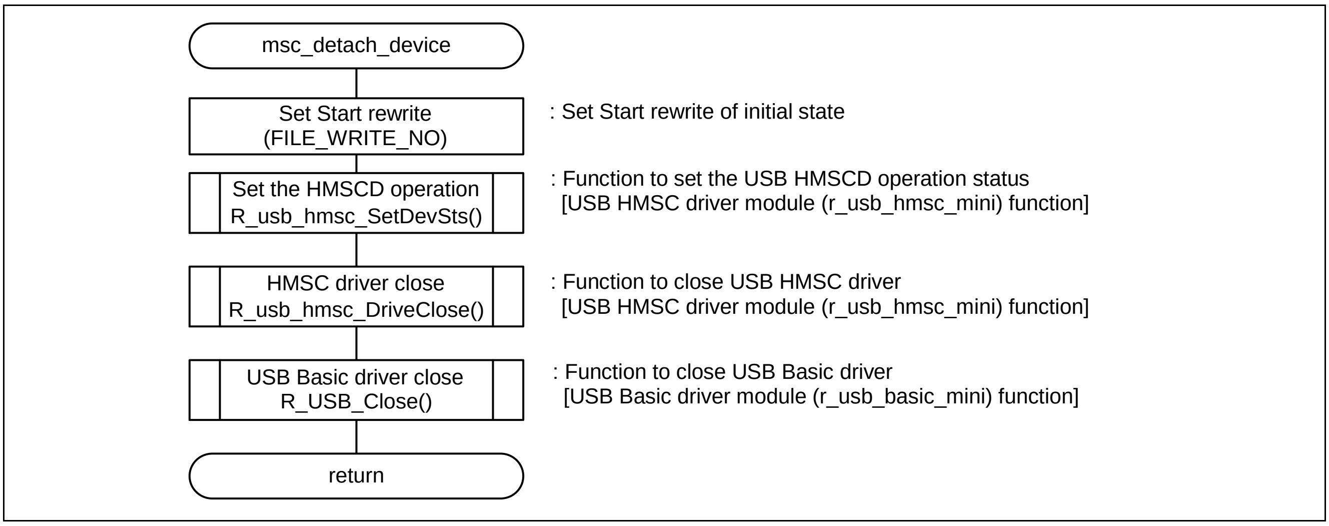

msc_detach_device
Set Start rewrite (FILE_WRITE_NO)
: Set Start rewrite of initial state
Set the HMSCD operation R_usb_hmsc_SetDevSts()
: Function to set the USB HMSCD operation status [USB HMSC driver module (r_usb_hmsc_mini) function]
HMSC driver close R_usb_hmsc_DriveClose()
: Function to close USB HMSC driver [USB HMSC driver module (r_usb_hmsc_mini) function]
USB Basic driver close R_USB_Close()
: Function to close USB Basic driver [USB Basic driver module (r_usb_basic_mini) function]
return

Figure 6-15 USB disconnection

(14) USB driver callback processing

Figure 6-16 shows the flowchart of USB callback function registration.

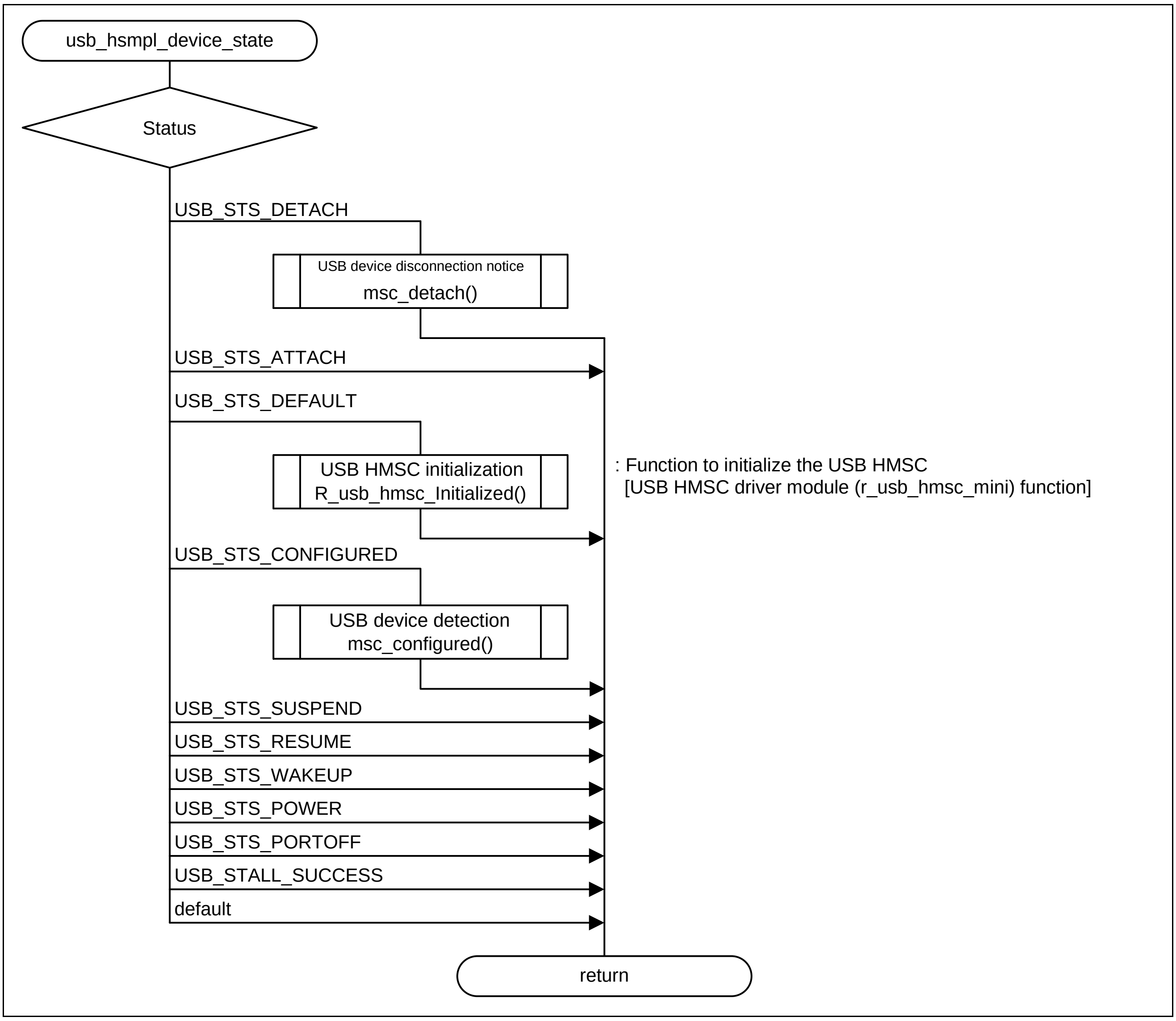


Figure 6-16 USB driver callback processing

(15) USB device detection notice

Figure 6-17 shows the flowchart of USB device detection notice

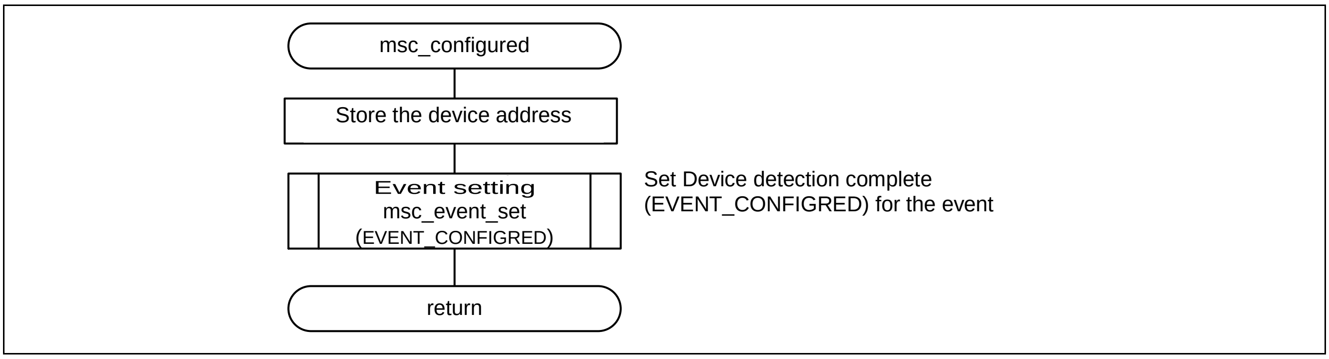


Figure 6-17 USB device detection notice

(16) USB device connection notice

Figure 6-18 shows the flowchart of USB device detection notice.

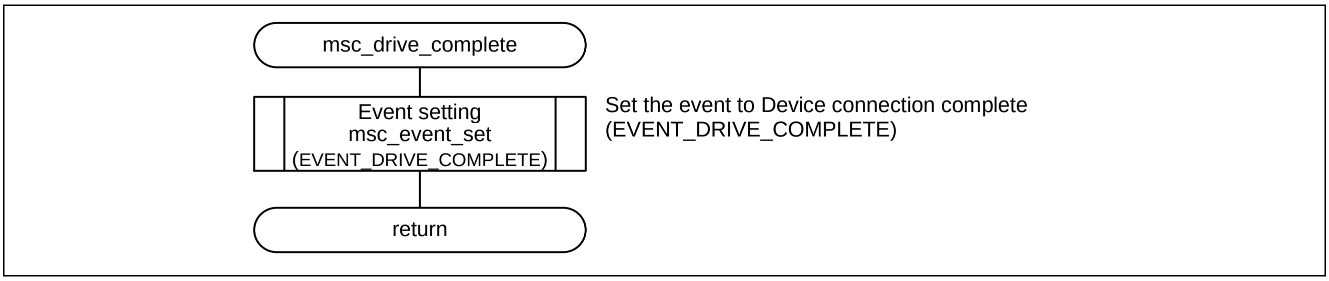


Figure 6-18 USB device connection notice

(17) USB device disconnection notice

Figure 6-19 shows the flowchart of USB device disconnection notice.

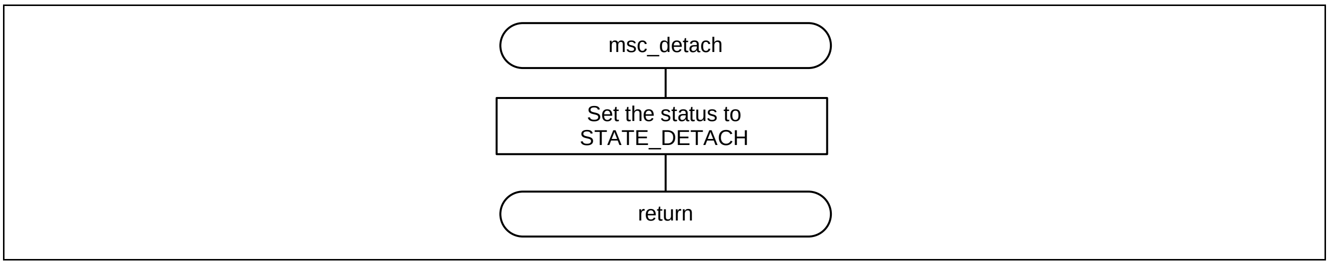


Figure 6-19 USB device disconnection notice

(18) Event setting

Figure 6-20 shows the flowchart of Event setting processing.

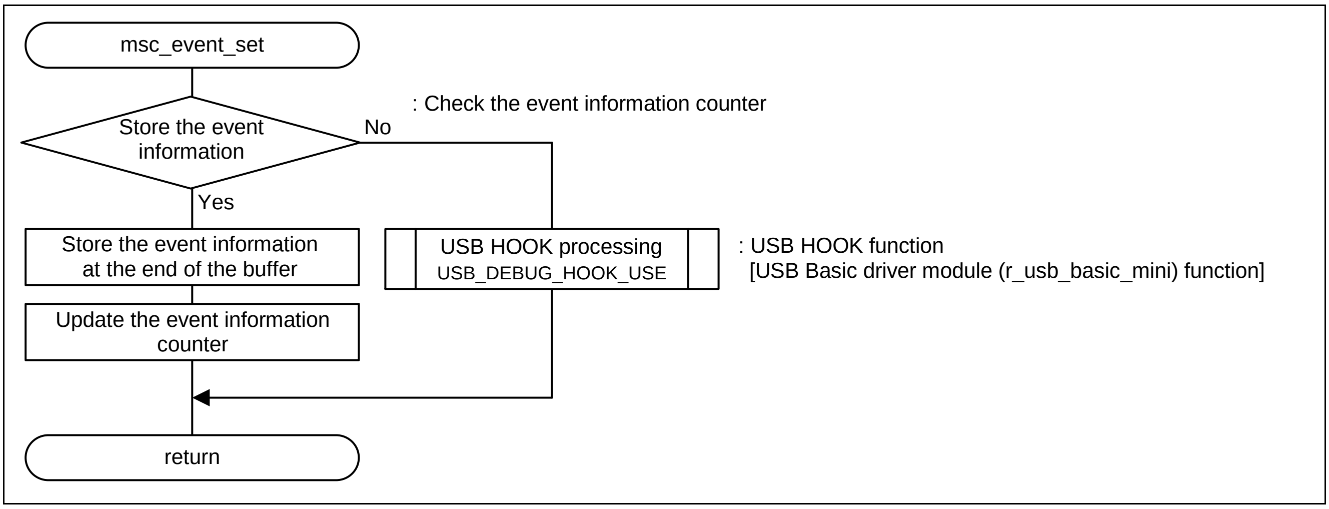

msc_event_set
Store the event information
No
: Check the event information counter
Yes
Store the event information at the end of the buffer
USB HOOK processing
USB_DEBUG_HOOK_USE
: USB HOOK function
[USB Basic driver module (r_usb_basic_mini) function]
Update the event information counter
return

Figure 6-20 Event setting

(19) Event notice

Figure 6-21 shows the flowchart of Event notice processing.

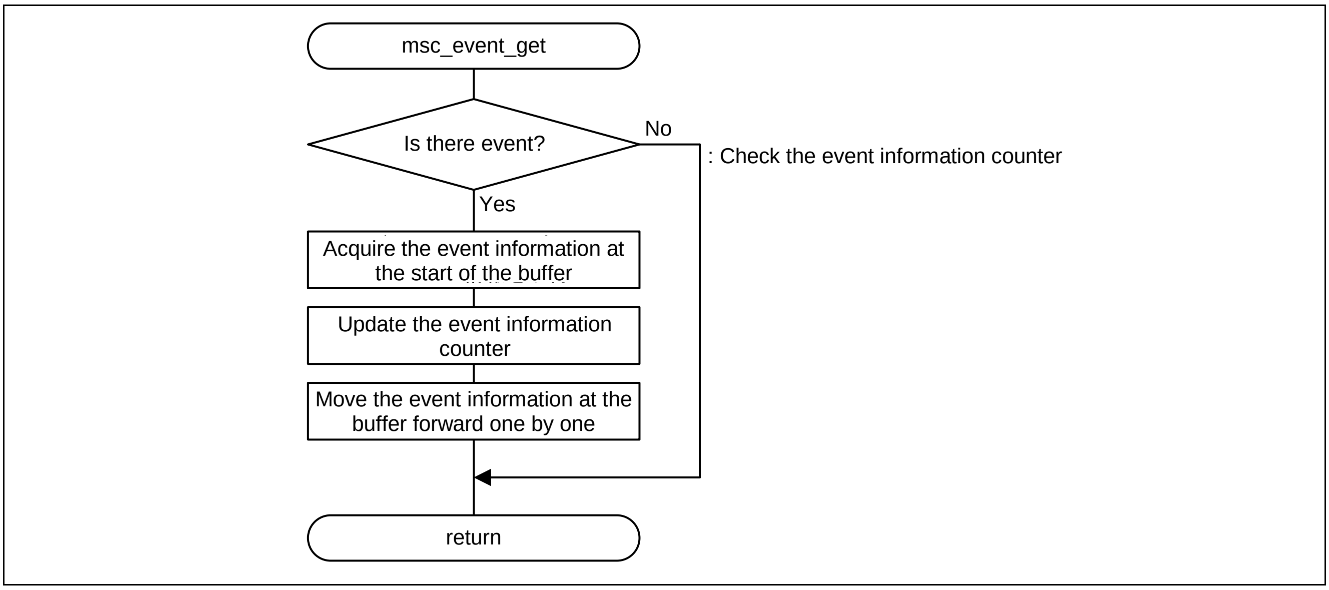

msc_event_get
Is there event?
No
: Check the event information counter
Yes
Acquire the event information at the start of the buffer
Update the event information counter
Move the event information at the buffer forward one by one
return

Figure 6-21 Event notice

(20) Flash memory rewrite processing

Figure 6-22 shows the flowchart of Flash memory rewrite processing. The last address for rewrite is set by this function.

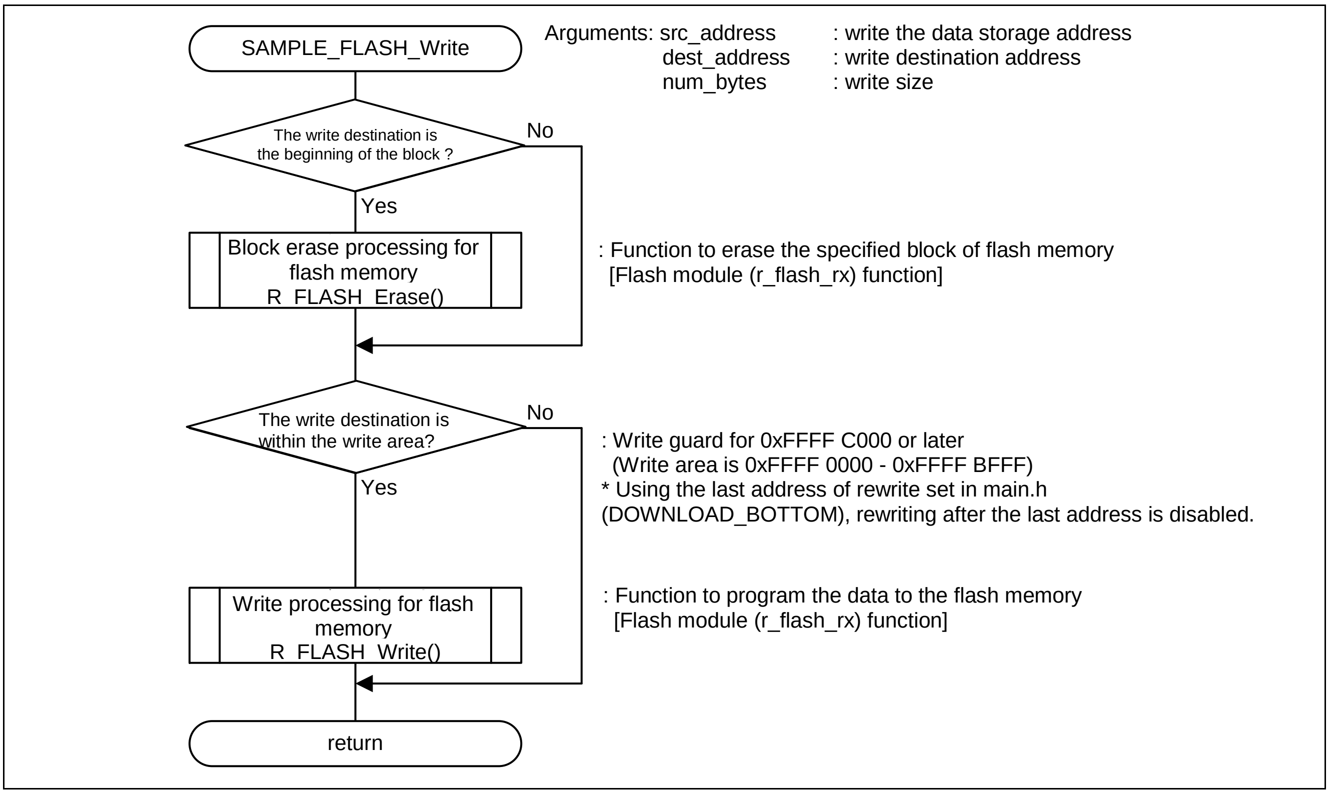

SAMPLE_FLASH_Write
Arguments: src_address : write the data storage address
dest_address : write destination address
num_bytes : write size
The write destination is the beginning of the block ?
No
Yes
Block erase processing for flash memory
R_FLASH_Erase()
: Function to erase the specified block of flash memory
[Flash module (r_flash_rx) function]
The write destination is within the write area?
No
Yes
: Write guard for 0xFFFF C000 or later
(Write area is 0xFFFF 0000 - 0xFFFF BFFF)
* Using the last address of rewrite set in main.h
(DOWNLOAD_BOTTOM), rewriting after the last address is disabled.
Write processing for flash memory
R_FLASH_Write()
: Function to program the data to the flash memory
[Flash module (r_flash_rx) function]
return

Figure 6-22 Flash memory rewrite processing

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	Feb 29, 2016	-	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.

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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

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 5.0