R01AN2294EJ0130
Rev.1.30
Jul 31, 2024

RX Family

Sample Program using USB Host Human Interface Device Class Driver (HHID) for USB Mini Firmware to communicate via USB with HID device Firmware Integration Technology

Introduction

This document describes the following sample firmware: USB Host Human Interface Devices Class Driver using Firmware Integration Technology. The sample firmware is referred to below as the HHID.

When developing an actual software, be sure to use the “USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note” (Document number: R01AN2166) together with the user’s manual for each MCU (Hardware). In addition, also refer to the " USB Host Human Interface Device Class Driver for USB Mini Firmware using Firmware Integration Technology Application Note” (Document number: R01AN2168), if necessary. “USB Basic Mini Host and Peripheral Driver (USB Mini Firmwae) using Firmware Integration Technology Application Note” (Document number: R01AN2166) is located in the "reference_documents" folder within the package.

Target Device

RX111 Group
RX113 Group
RX231 Group
RX23W Group
RX261 Group

The operation of this program has been confirmed using the Renesas Starter Kit (RSK), the Renesas Solution Starter Kit (RSSK) or EK.

Contents

1. Introduction

1.1 Functions

The HHID performs communication with HID devices in conformance with the USB human interface devices class specification (HID).
It transfers HID class data when a mouse or keyboard is connected.

1.2 FIT Module Configuration

The HHID comprises the following FIT modules and a sample application:

Table 1-1 FIT Module Configuration

FIT Module	Folder Name
Board Support Package Module Firmware Integration Technology	r_bsp
RX Family USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology	r_usb_basic_mini
RX Family USB Host Human Interface Devices Class Driver for USB Mini Firmware using Firmware Integration Technology	r_usb_hhid_mini

Refer to the related documentation for details of each FIT module. Note that the latest versions of the FIT modules used by the sample firmware are available for download from the following website:

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

1.3 Note

This driver is not guaranteed to provide USB communication operation. The customer should verify operation when utilizing it in a system and confirm the ability to connect to a variety of different types of devices.

1.4 Operating Confirmation Environment

The operating confirmation environment for the HHID is described below:

Table 1-2 Operation Confirmation Environment

Item	Contents
C compiler	Renesas Electronics C/C++ compiler for RX Family (The option "-lang=C99" is added to the default setting of IDE)
	GCC for Renesas RX (The option "-std=gnu99" is added to the default setting of IDE)
	IAR C/C++ Compiler for Renesas RX
Real-Time OS	FreeRTOS RI600V4
Endian	Little Endian, Big Endian
USB Driver Revision Number	Rev.1.30
Using Board	Renesas Starter Kit for RX111 Renesas Starter Kit for RX113 Renesas Starter Kit for RX231 Renesas Solution Starter Kit for RX23W EK-RX261

2. Software Configuration

2.1 Module Configuration

The HHID comprises an HID class driver as well as mouse and keyboard device drivers. When data is received from a connected HID device, the result is reported to the APL via the HCD. When the APL generates a data transfer request, it is reported to the HID device via the HCD.
Figure 2-1 shows the module configuration of the HHID, Table 2-1 lists the functions of the modules.

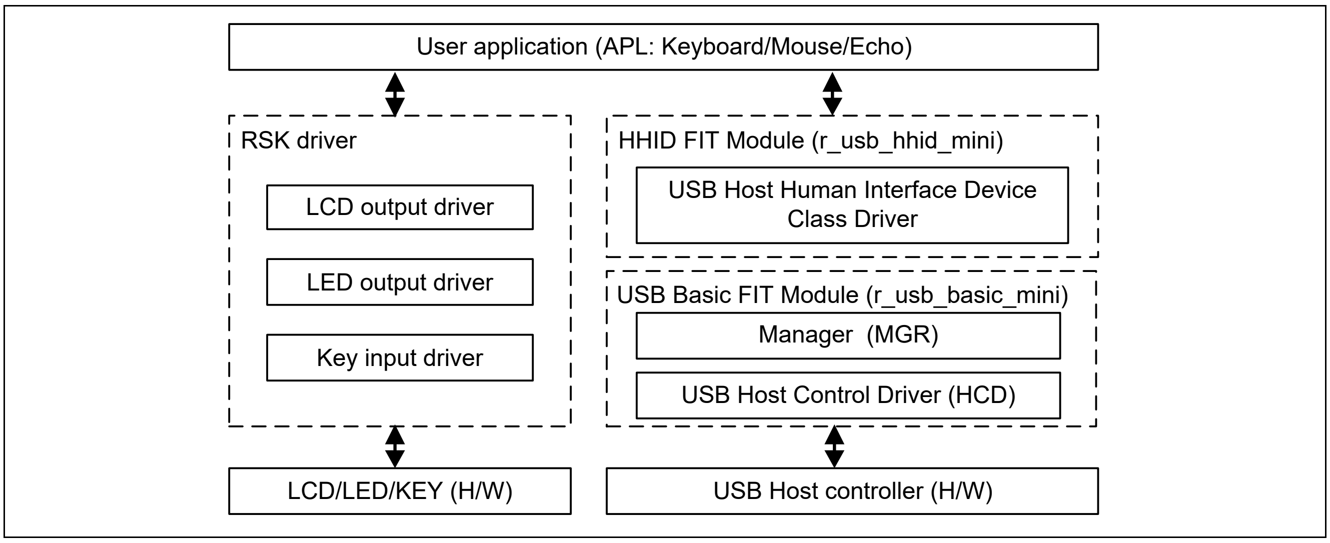

User application (APL: Keyboard/Mouse/Echo)
RSK driver
LCD output driver
LED output driver
Key input driver
HHID FIT Module (r_usb_hhid_mini)
USB Host Human Interface Device Class Driver
USB Basic FIT Module (r_usb_basic_mini)
Manager (MGR)
USB Host Control Driver (HCD)
LCD/LED/KEY (H/W)
USB Host controller (H/W)

Figure 2-1 Moudle Configuration

Table 2-1 Functions of Modules

Module Name	Function
APL	Sample application program ・ Starts communication with the HID device and controls suspending and resuming by means of switch operations. ・ Displays on the LCD report information received from the HID device.
RSK driver	Sample application for using the peripheral functions of the RSK
HHID (r_usb_hhid_mini)	HID class driver ・ Interprets requests from the HID device. ・ Reports switch manipulation information from the APL to the HID device via the HCD.
HCD (r_usb_basic_mini)	USB Host Control Driver

3. Setup

3.1 Hardware

3.1.1 Example Operating Environment

Figure 3-1 shows an example operating environment for the HHID. Refer to the associated instruction manuals for details on setting up the evaluation board and using the emulator, etc.

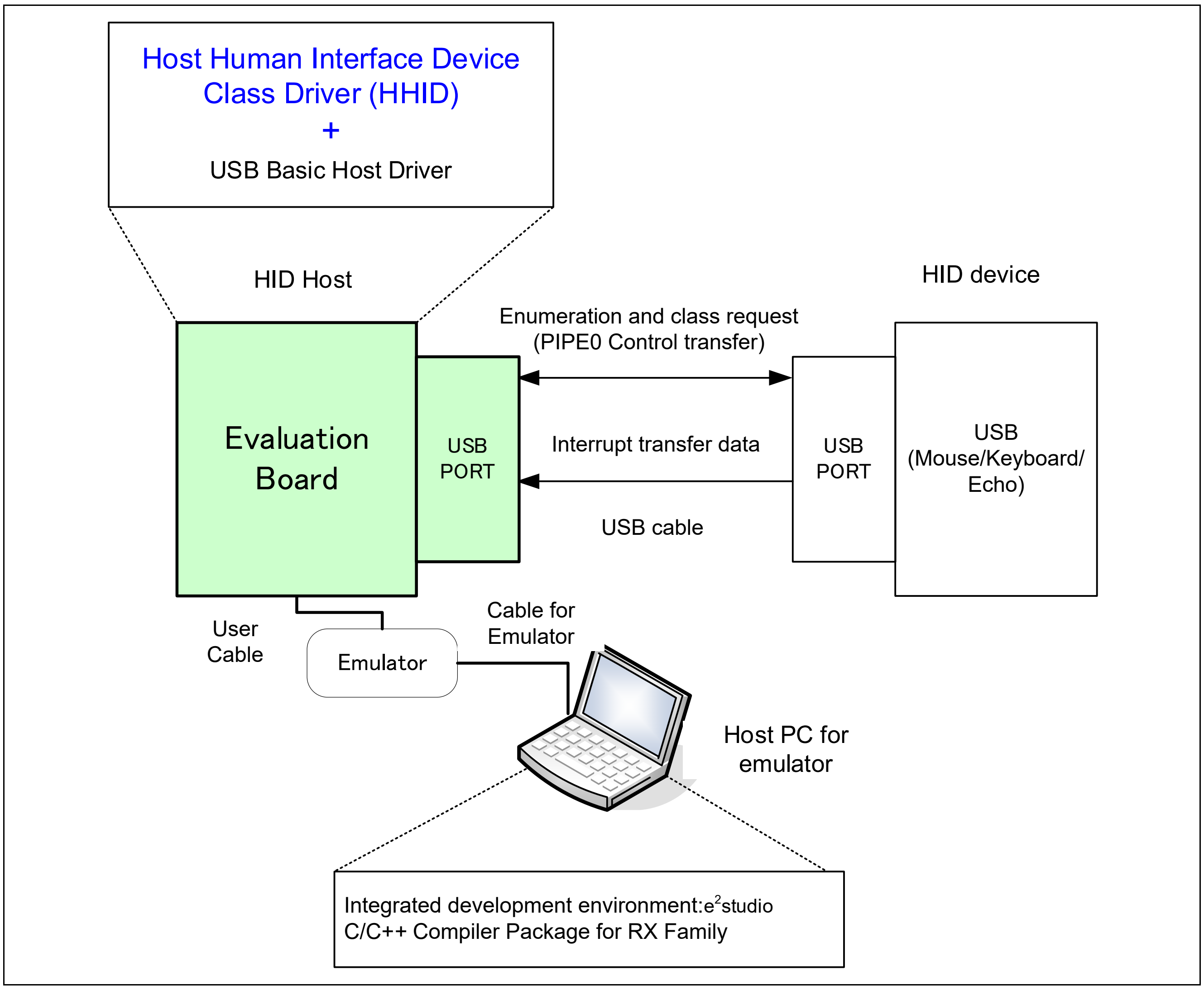

Host Human Interface Device Class Driver (HHID)
+
USB Basic Host Driver
HID Host
HID device
Enumeration and class request (PIPE0 Control transfer)
Evaluation Board
USB PORT
Interrupt transfer data
USB PORT
USB (Mouse/Keyboard/ Echo)
USB cable
User Cable
Emulator
Cable for Emulator
Host PC for emulator
Integrated development environment:e²studio
C/C++ Compiler Package for RX Family

Figure 3-1 Example Operating Environment

Table 3-1 shows the evaluation board on which operation has been confirmed.

Table 3-1 Evaluation Board on which HHID Operation Has Been Verified

MCU	Evaluation Board
RX111	RSKRX111
RX113	RSKRX113
RX231	RSKRX231
RX23W	RSSKRX23W
RX261	EK-RX261

3.1.2 RSK / RSSK / EK Setting

It is necessary to set RSK/RSSK/EK to operate in the host mode. Please refer to the following.

Table 3-2 **Jumper Setting**

RSK	Jumper Setting
RSKRX111	J12: Shorted Pin1-2
RSKRX113	J12: Shorted Pin1-2
RSKRX231	J15: Shorted Pin1-2
RSSKRX23W	J5: Shorted Pin2-3
EK-RX261	J18: Open, J19: Shorted Pin 1-2

Note:

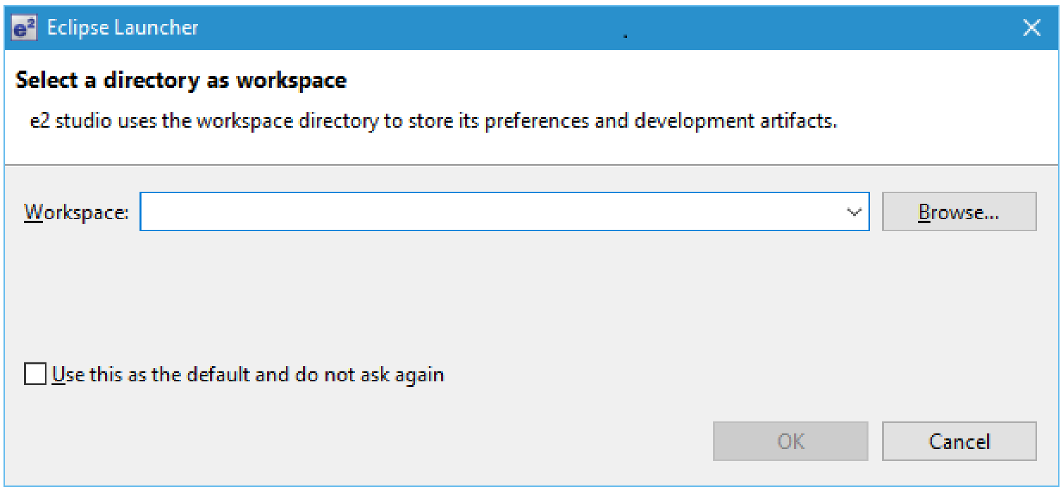
For the detail of RSK / RSSK / EK setting, refer to the user's manual of RSK / RSSK.

3.2 Software

1) Setup e^2 studio

a) Start e^2 studio

b) If you start up e^2 studio at first, the following dialog is displayed. Specify the folder to store the project in this dialog.


Eclipse Launcher

Select a directory as workspace

e2 studio uses the workspace directory to store its preferences and development artifacts.

Workspace: [ ] Browse...

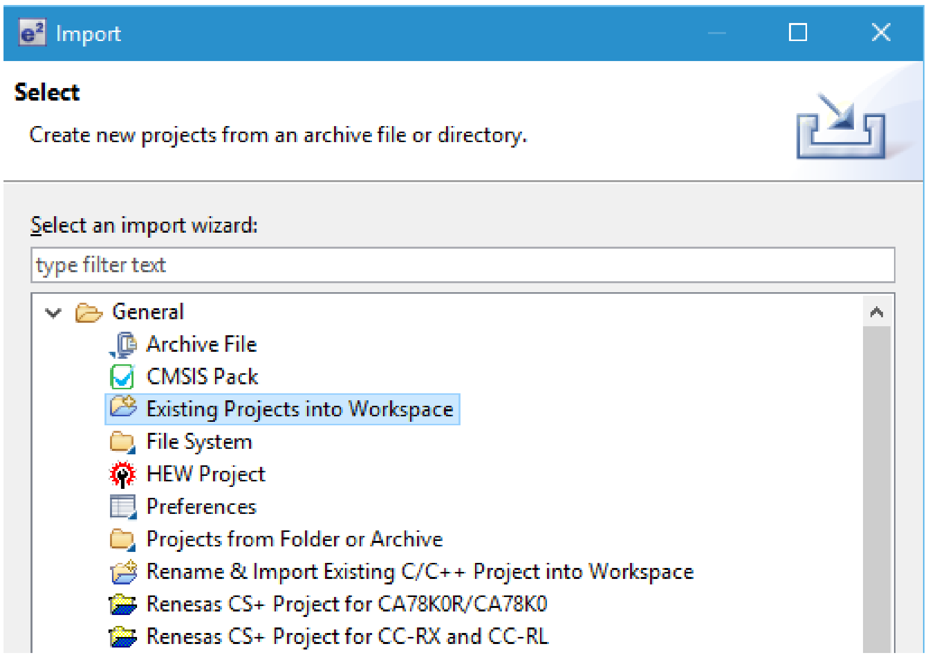
☐ Use this as the default and do not ask again

OK Cancel

2) Import the project to the workspace

a) Select [File] > [Import]

b) Select [General] => [Existing Projects into Workspace]


Import

Select

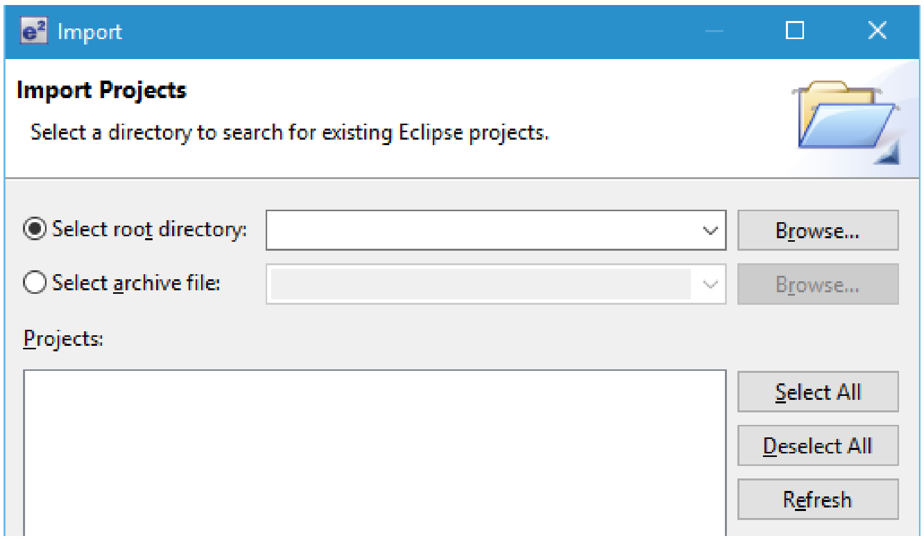
Create new projects from an archive file or directory.

Select an import wizard:

type filter text

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

c) Select the root directory of the project, that is, the folder containing the ".cproject" file.



d) Click "Finish".

You have now imported the project into the workspace. Note that you can import other projects into the same workspace.

3) Generate the binary target program by clicking the "Build" button.

4) Connect the target board to the debug tool and download the executable. The target is run by clicking the "Run" button.

4. Sample Application

4.1 Application Specifications

The following three application programs are provided:

4.1.1 Normal Mode Application (demo_src¥r_usb_hhid_apl.c)

Transfers data to and from an HID device (mouse or keyboard) connected to the RSK/RSSK. Data received from the HID device is read and discarded.

4.1.2 Demo Mode Application (demo_src¥r_usb_hhid_apl_demo.c)

Transfers data to and from an HID device (mouse or keyboard) connected to the RSK/RSSK. Data received from the HID device is displayed on an LCD. In addition, the processing is provided for sending of suspend and resume signals to the HID device.

4.1.3 Echo (Loopback) Mode Application (demo_src¥r_usb_hhid_apl_echo.c)

Performs echo(loopback) processing in which data received from an HID device connected to the RSK/RSSK is sent unmodified back to the HID device.

[Note]

Loopback processing is possible only when an HID device that supports interrupt out transfer is connected.

4.2 Application Processing (for Non-OS)

The application comprises two parts: initial settings and main loop. An overview of the processing in these two parts is provided below.

4.2.1 Initial Settings

Initial settings consist of MCU pin settings, USB driver settings, and initial settings to the USB controller.

4.2.2 Main Loop (Normal mode: demo_src¥r_usb_hhid_apl.c)

The main loop performs processing to receive data from the HID device as part of the main routine. An overview of the processing of the main loop is presented below.

1. When the *R_USB_GetEvent* function is called after an HID device attaches to the USB host (RSK/RSSK) and enumeration completes, *USB_STS_CONFIGURED* is set as the return value. When the APL confirms *USB_STS_CONFIGURED*, it calls the *R_USB_Write* function to request transmission of data to the HID device.
2. When the *R_USB_GetEvent* function is called after sending of class request *SET_PROTOCOL* to the HID device has completed, *USB_STS_REQUEST_COMPLETE* is set as the return value. When the APL confirms *USB_STS_REQUEST_COMPLETE*, it calls the *R_USB_Read* function to make a data receive request for data sent by the HID device.
3. When the *R_USB_GetEvent* function is called after reception of data from the HID device has completed, *USB_STS_READ_COMPLETE* is set as the return value. When the APL confirms *USB_STS_READ_COMPLETE*, it calls the *R_USB_Read* function to make a data receive request for data sent by the HID device.
4. The processing in step 3, above, is repeated.

An overview of the processing performed by the APL is shown below:

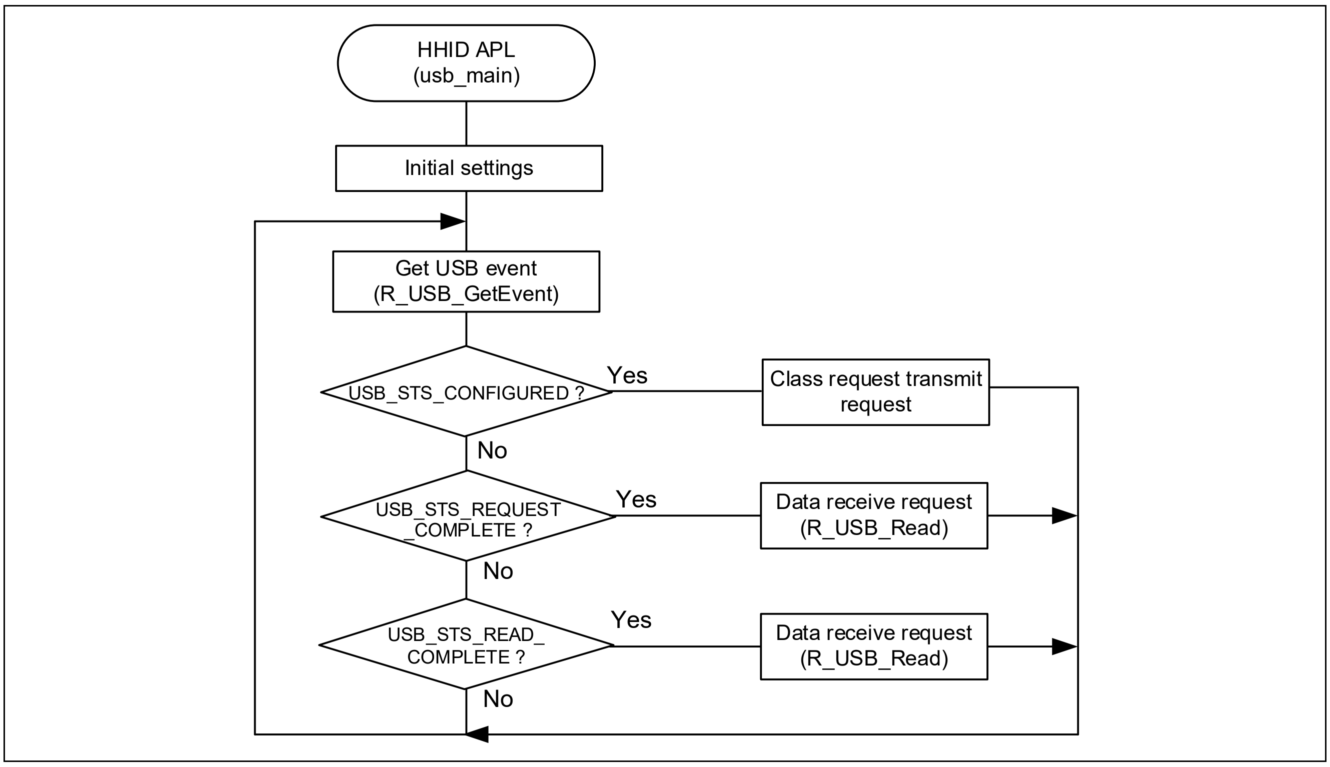

HHID APL
(usb_main)
Initial settings
Get USB event
(R_USB_GetEvent)
USB_STS_CONFIGURED ?
Yes
Class request transmit request
No
USB_STS_REQUEST _COMPLETE ?
Yes
Data receive request (R_USB_Read)
No
USB_STS_READ_ COMPLETE ?
Yes
Data receive request (R_USB_Read)
No

Figure 4-1 Main Loop (Normal mode)

4.2.3 Main Loop (Demo mode: demo_src¥r_usb_hhid_demo.c)

This application program performs the processing described below. For the data reception processing from the HID device, refer to 4.2.2, Main Loop (Normal mode: demo_src¥r_usb_hhid_apl.c).

1. The data reception processing from the HID device
2. LCD display based on data received from the HID device.
3. Processing to transmit a suspend or resume signal to the HID device.

[Note]

Pressing a switch(button) on the RSK/RSSK functions as the transmit trigger for sending a suspend or resume signal to the HID device. For LCD/LED indication of the reception data and the switch(button) specifications , refer to 4.4.1,Switch (button).

4.2.4 Main Loop (Echo(Loopback) mode: demo_src¥r_usb_hhid_echo.c)

In echo(loop-back) mode, loop-back processing in which data sent by the USB host is received and then transmitted unmodified back to the USB host takes place as part of the main routine. An overview of the processing of the main loop is presented below.

1. When the *R_USB_GetEvent* function is called after enumeration with the USB host completes, *USB_STS_CONFIGURED* is set as the return value. When the APL confirms *USB_STS_CONFIGURED*, it calls the *R_USB_Write* function to make the data transmission request to HID device.
2. When the *R_USB_GetEvent* function is called after the data transmission to HID device completes, *USB_STS_WRITE_COMPLETE* is set as the return value. When the APL confirms *USB_STS_WRITE_COMPLETE*, it calls the *R_USB_Read* function to make a data receive request for data sent by HID device.
3. When the *R_USB_GetEvent* function is called after reception of data from the USB device has completed, *USB_STS_READ_COMPLETE* is set as the return value. When the APL confirms *USB_STS_READ_COMPLETE*, it calls the *R_USB_Write* function to make a data transmit request to transmit the received data to the USB host.
4. The processing in steps 2 and 3, above, is repeated.

An overview of the processing performed by the APL is shown below:

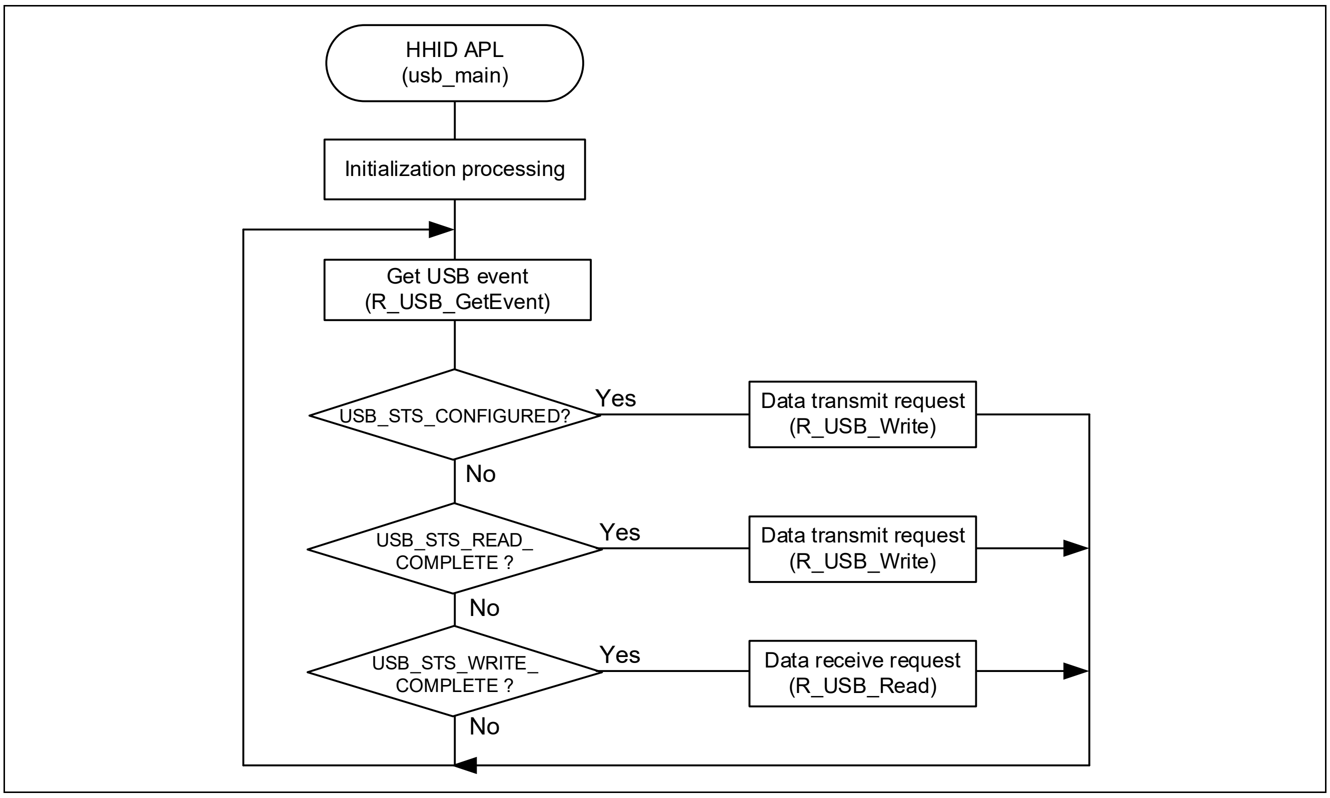


Figure 4-2 Main Loop (Echo mode)

4.3 Application Processing (for RTOS)

The application comprises two parts: initial settings and main loop. An overview of the processing in these two parts is provided below.

4.3.1 Initial Settings

Initial settings consist of MCU pin settings, USB driver settings, and initial settings to the USB controller.

4.3.2 Main Loop (Normal mode: demo_src¥r_usb_hhid_apl.c)

The loop performs processing to receive data from the HID device as part of the main routine. An overview of the processing performed by the loop is shown below.

1. When a USB-related event has completed, the USB driver calls the callback function (*usb_apl_callback*). In the callback function (*usb_apl_callback*), the application task (APL) is notified of the USB completion event using the real-time OS functionality.
2. In APL, information regarding the USB completion event was notified from the callback function is retrieved using the real-time OS functionality.
3. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_CONFIGURED*, APL sends the class request (*SET_PROTOCOL*) to the HID device.
4. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_REQUEST_COMPLETE*, APL performs a data reception request to receive data transmitted from the HID device by calling the *R_USB_Read* function.
5. The avove processing is repeated.

An overview of the processing performed by the APL is shown below:

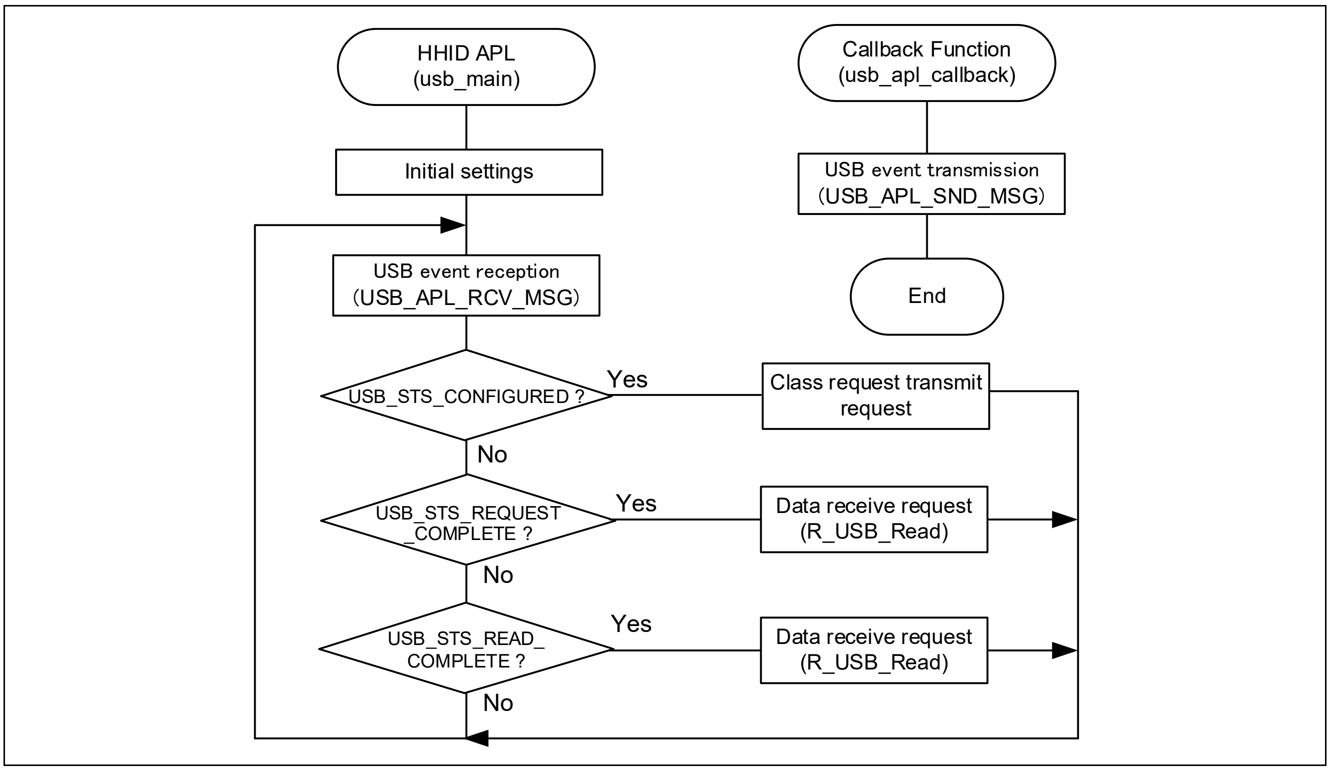


Figure 4-3 Main Loop (Normal mode)

4.3.3 Main Loop (Demo mode: demo_src¥r_usb_hhid_demo.c)

This application program performs the processing described below. For the data reception processing from the HID device, refer to **4.2.2, Main Loop (Normal mode: demo_src¥r_usb_hhid_apl.c)**.

1. The data reception processing from the HID device
2. LCD display based on data received from the HID device.
3. Processing to transmit a suspend or resume signal to the HID device.

[Note]

Pressing a switch(button) on the RSK functions as the transmit trigger for sending a suspend or resume signal to the HID device. For LCD/LED indication of the reception data and the switch(button) specifications, refer to **4.4, Switch(Button) Operations and LCD/LED Indications in Demo Mode.**

4.3.4 Main Loop (Echo(Loopback) mode: demo_src¥r_usb_hhid_echo.c)

In echo(loop-back) mode, loop-back processing in which data sent by the USB host is received and then transmitted unmodified back to the USB host takes place as part of the main routine. An overview of the processing performed by the loop is shown below.

1. When a USB-related event has completed, the USB driver calls the callback function (*usb_apl_callback*). In the callback function (*usb_apl_callback*), the application task (APL) is notified of the USB completion event using the real-time OS functionality.
2. In APL, information regarding the USB completion event was notified from the callback function is retrieved using the real-time OS functionality.
3. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_CONFIGURED*, APL performs a data transmission request to the HID device by calling *R_USB_Read* function.
4. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_WRITE_COMPLETE*, APL performs a data reception request to receive data transmitted from the HID device by calling the *R_USB_Read* function.
5. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_READ_COMPLETE*, APL performs a data transmission request to send the recieved data to HID device by calling the *R_USB_Write* function.
6. The avove processing is repeated.

An overview of the processing performed by the APL is shown below:

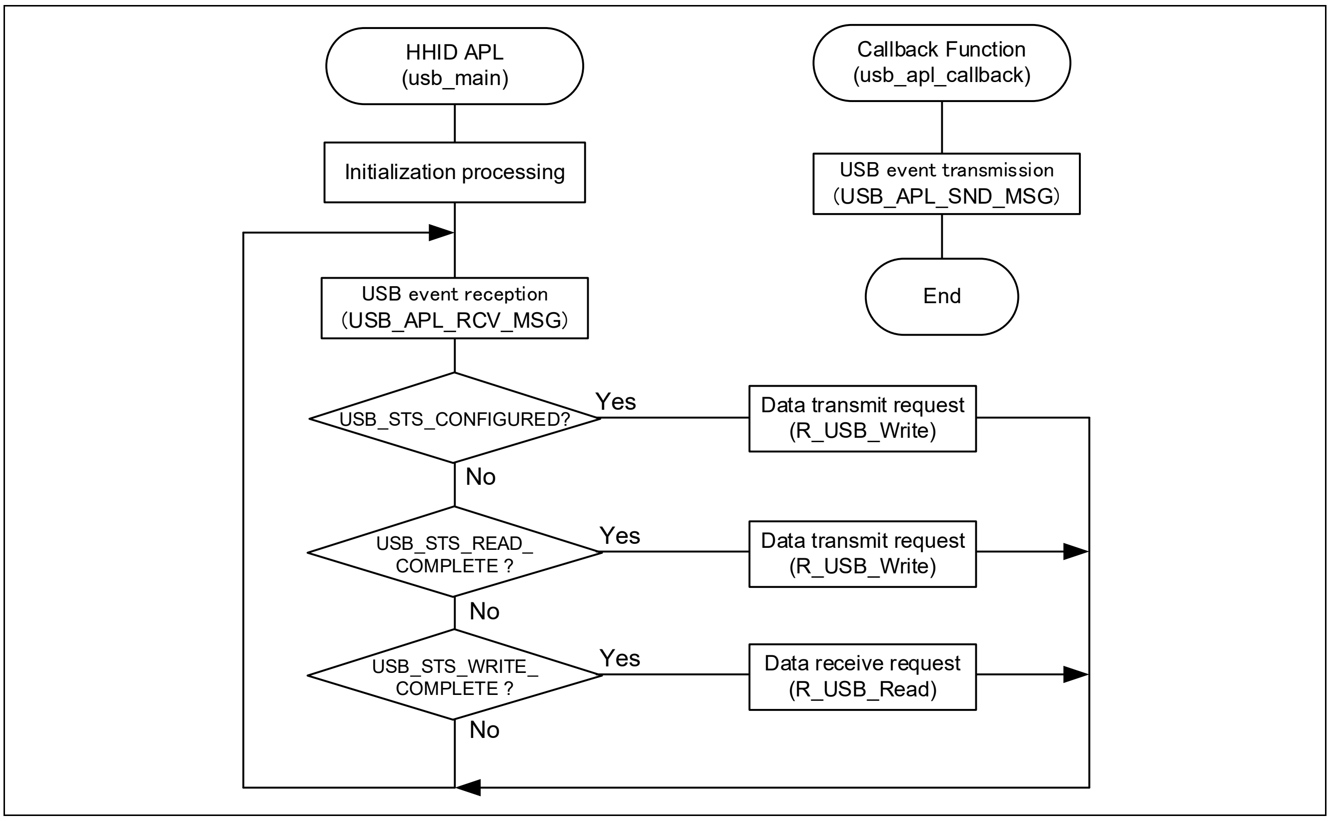


Figure 4-4 Main Loop (Echo mode)

4.4 Switch(Button) Operations and LCD/LED Indications in Demo Mode

4.4.1 Switch (button)

The APL starts data transfer after an HID device is connected. Pushing the switch while data transfer is in progress causes the following operations to occur:

1. When in the data transfer state, pressing SW2 transitions the HID device to the suspend state.
2. When the HID device is in the suspend state, pressing SW2 cancels the suspend state.

Table 4-1 shows the switch(button) specification.

Table 4-1 Switch Specification

Swich Name	Switch Number	Description
State Change Switch	Switch2(SW2)	Changes the status of the HID device connected to the USB0 module. 1. Data transfer state: Transitions to suspend state 2. Suspend state: Transitions to data transfer state

[Note]

For details of the switch and MCU pin connections on the RSK/RSSK/EK, refer to the instruction manual of the RSK/RSSK and the user's manual of the MCU.

4.4.2 Display Information

The APL displays on the LCD screen the connection state of the HID device and data received from the connected HID device.

Mouse connected	:	Displays on the LCD the amount of movement on the X and Y axes (– 127 to 127). Lighting on LED0 when right clicking, Lighting on LED1 when left clicking, Lighting on LED2 when wheel button clicking.
Keyboard connected	:	Displays on the LCD the last input key data.

The LCD indication does not change when the data received from the HID device is NULL (no key on keyboard pressed, mouse not moved on X or Y axes).

4.5 Configuration File for the application program (r_usb_hhid_apl_config.h)

Make settings for the definitions listed below.

1. OPERATION_MODE Definition

Specify one of the following settings for the *OPERATION_MODE* definition.

```
#define   OPERATION_MODE   HID_NORMAL          // Normal Mode
#define   OPERATION_MODE   HID_DEMO            // Demo Mode
#define   OPERATION_MODE   HID_ECHO            // Echo Mode
```

2. USB_SUPPORT_RTOS Definition

Please specify *USB_APL_ENABLE* to *USB_SUPPORT_RTOS* definition when using the real-time OS.

```
#define   USB_SUPPORT_RTOS   USB_APL_DISABLE   // No use the real-time OS
#define   USB_SUPPORT_RTOS   USB_APL_ENABLE    // Use the real-time OS
```

3. Note

The above configuration settings apply to the application program. USB driver configuration settings are required in addition to the above settings. For information on USB driver configuration settings, refer to the application note *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology* (Document number: R01AN2166).

5. Class Driver Overview

5.1 Class Request (Request from Host to Device)

Table 5-1 lists the class requests supported by the HHID.

Table 5-1 Supported Basic Requests and HID Class Requests

Request	Code	Description
GET_REPORT	0x01	Requests a report from the HID device.
SET_REPORT	0x09	Notifies the HID device of a report.
GET_IDLE	0x02	Requests the duration from the HID device.
SET_IDLE	0x0A	Notifies the HID device of the duration.
GET_PROTOCOL	0x03	Requests the protocol from the HID device.
SET_PROTOCOL	0x0B	Notifies the HID device of the protocol
GET_REPORT_DESCRIPTOR	Standard	Requests the report descriptor.
GET_HID_DESCRIPTOR	Standard	Requests the HID descriptor

5.2 Data Format

The boot protocol data format of data received from the keyboard or mouse through interrupt-IN transfers is shown below

Table 5-2 Receive Data Format (Boot Protocol)

offset	Keyboard (8 Bytes)	Mouse (3 Bytes)
0 (Top Byte)	Modifier keys	b0 : Button 1 b1 : Button 2 b2 : Button 3 b3-b7 : Reserved
+1	Reserved	X displacement
+2	Keycode 1	Y displacement
+3	Keycode 2	—
+4	Keycode 3	—
+5	Keycode 4	—
+6	Keycode 5	—
+7	Keycode 6	—

[Note]

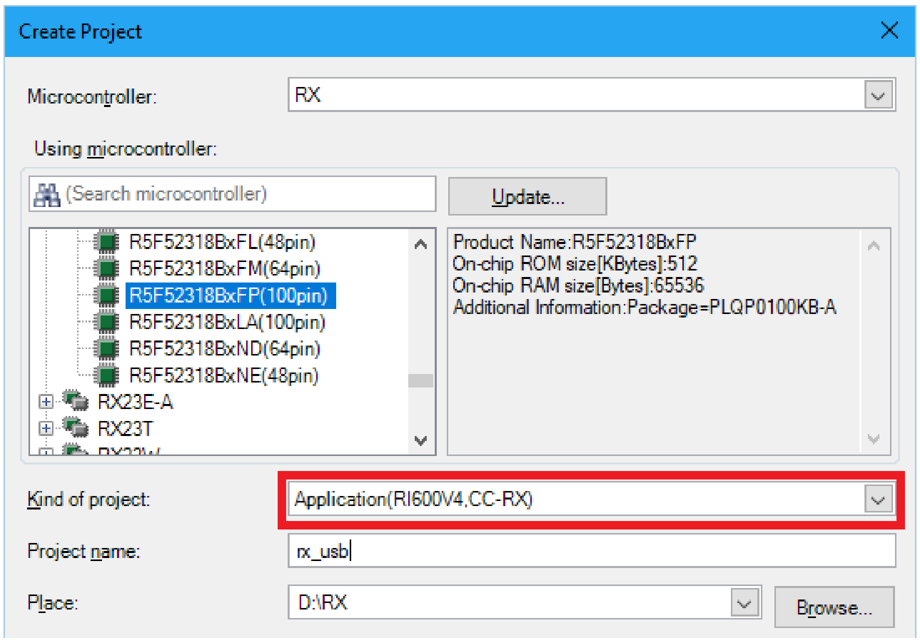
The data format used for report protocol data transfers must conform to the report descriptor. This driver does not acquire and analyze the report descriptor; it determines the report format according to whether the interface protocol code is “keyboard” or “mouse.” The user should make changes as necessary to match the system specifications.

6. Using RI600V4 project with CS+

The RI600V4 project in the package does not support CS+. The user needs to create a project for CS+ according to the following procedure when using RI600V4 project on CS+.

6.1 New Project Creation

Select "Application(RI600V4, CC-RX) for the Kind of project.



Create Project

Microcontroller: RX

Using microcontroller:

(Search microcontroller) Update...

- R5F52318BxFL(48pin)
- R5F52318BxFM(64pin)
- R5F52318BxFP(100pin)
- R5F52318BxLA(100pin)
- R5F52318BxND(64pin)
- R5F52318BxNE(48pin)
- RX23E-A
- RX23T

Product Name:R5F52318BxFP
On-chip ROM size[KBytes]:512
On-chip RAM size[Bytes]:65536
Additional Information:Package=PLQP0100KB-A

Kind of project: Application(RI600V4,CC-RX)

Project name: rx_usb

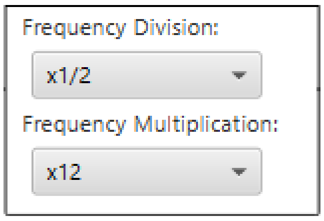
Place: D:\RX Browse...

6.2 Launch Smart Configurator

1. Clock Setting (Select "Clocks" tab)

Set the related clock so that "48MHz" is set to UCLK (USB clock).

The following is a setting example when using the oscillator(8MHz).



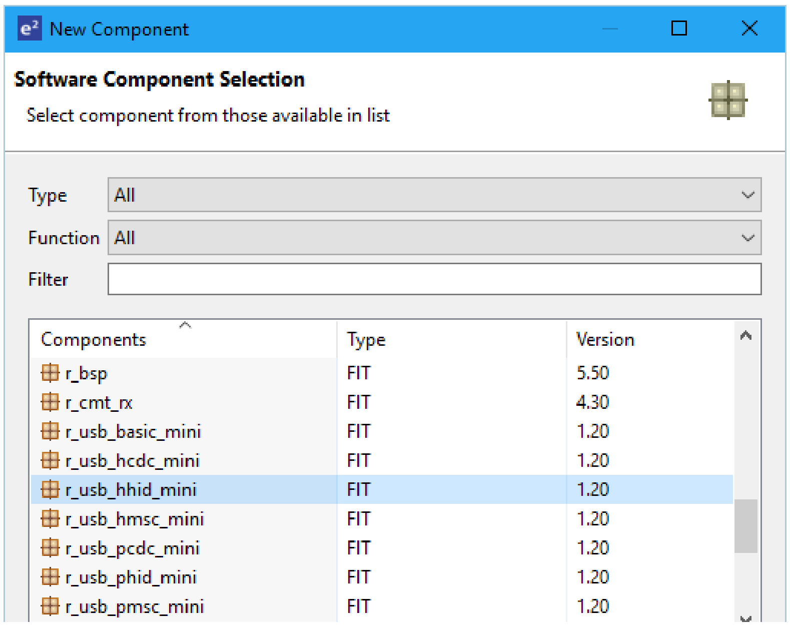
Frequency Division: x1/2

Frequency Multiplication: x12

2. Component Setting (Select "Components" tab)

(1). Import the USB FIT module

Select the *r_usb_hhid_mini* module and press the "Finish" button. The *r_usb_basic_mini* module is imported at the same time.



New Component

Software Component Selection

Select component from those available in list

Type: All

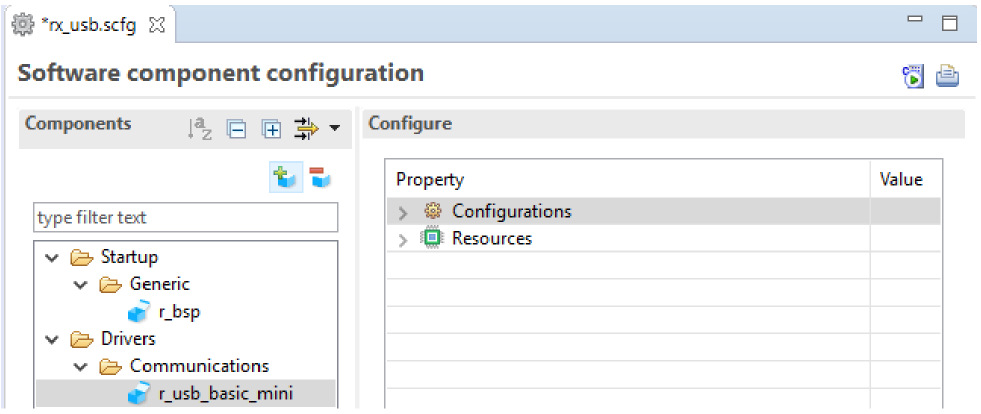
Function: All

Filter:

Components	Type	Version
r_bsp	FIT	5.50
r_cmt_rx	FIT	4.30
r_usb_basic_mini	FIT	1.20
r_usb_hcdc_mini	FIT	1.20
r_usb_hhid_mini	FIT	1.20
r_usb_hmsc_mini	FIT	1.20
r_usb_pcdc_mini	FIT	1.20
r_usb_phid_mini	FIT	1.20
r_usb_pmsc_mini	FIT	1.20

(2). Configuration Setting

a. r_usb_basic_mini



(a). Configurations

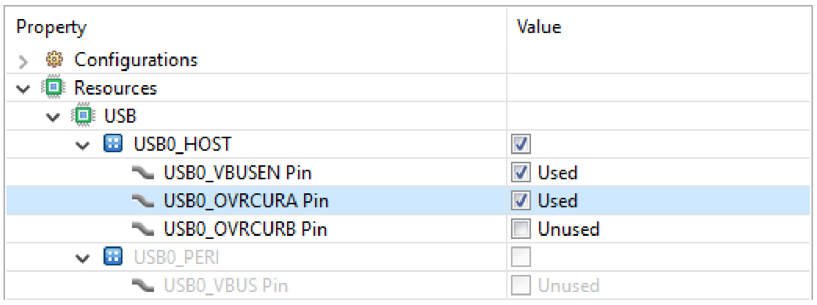
Set each item according to the user system.
For the detail of each item, refer to chapter "Configuration" in *USB Mini Basic Host and Peripheral Driver Firmware Integration Technology* application note (Document number: R01AN2166).

(b). Resources

Check the following check box.

i. *USBx_VBUSEN* pin

ii. *USBx_OVRCURA* pin or *USBx_OVRCURB* pin



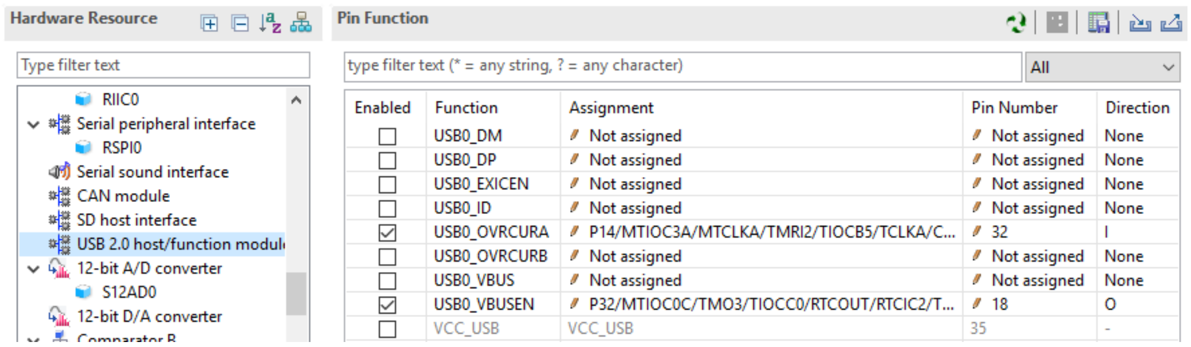
Property	Value
Configurations	
Resources	
USB	
USB0_HOST	☑
USB0_VBUSEN Pin	☑ Used
USB0_OVRCURA Pin	☑ Used
USB0_OVRCURB Pin	☐ Unused
USB0_PERI	☐
USB0_VBUS Pin	☐ Unused

b. r_usb_hhid_mini

Refer to chapter "Configuration" in *USB Host Human Interface Devices Class Driver (HHID) for USB Mini Firmware Firmware Integration Technology* application note (Document number: R01AN2168).

3. Pin Setting (Select "Pins" tab)

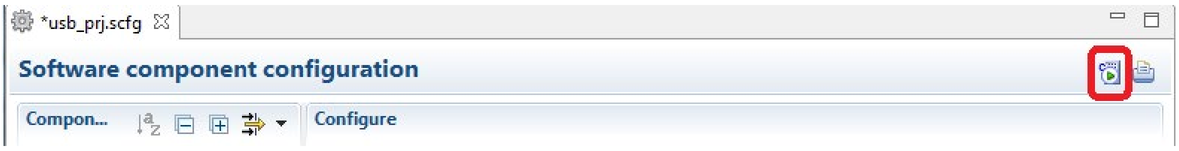
Select the port for USB pin match the user system.



Enabled	Function	Assignment	Pin Number	Direction
☐	USB0_DM	Not assigned	Not assigned	None
☐	USB0_DP	Not assigned	Not assigned	None
☐	USB0_EXICEN	Not assigned	Not assigned	None
☐	USB0_ID	Not assigned	Not assigned	None
✓	USB0_OVRCURA	P14/MTIOC3A/MTCLKA/TMRI2/TIOCB5/TCLKA/C...	32	I
☐	USB0_OVRCURB	Not assigned	Not assigned	None
☐	USB0_VBUS	Not assigned	Not assigned	None
✓	USB0_VBUSEN	P32/MTIOC0C/TMO3/TIOCC0/RTCOUT/RTCIC2/T...	18	O
☐	VCC_USB	VCC_USB	35	-

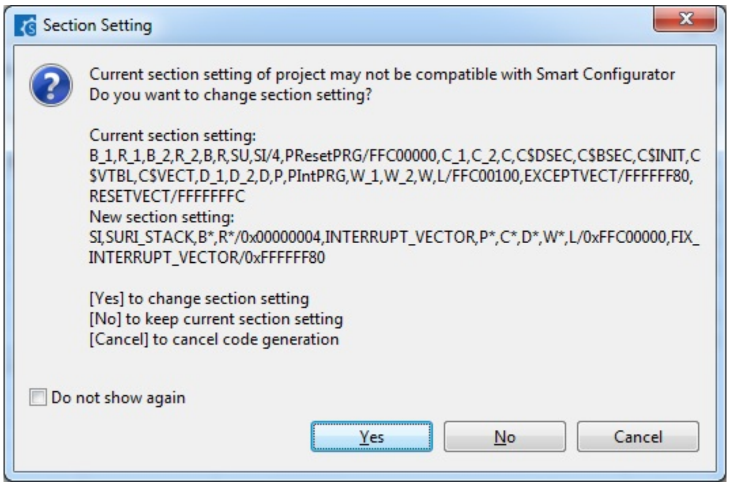
4. Generate Code

The Smart Configurator genrates source codes for USB FIT module and USB pin setting in "*<ProjectDir>¥src¥smc_gen"* folder by by clicking on the [(Generate Code)] button.



Note:

Select "Yes" if the following dialog box is displayed.



Current section setting of project may not be compatible with Smart Configurator
Do you want to change section setting?

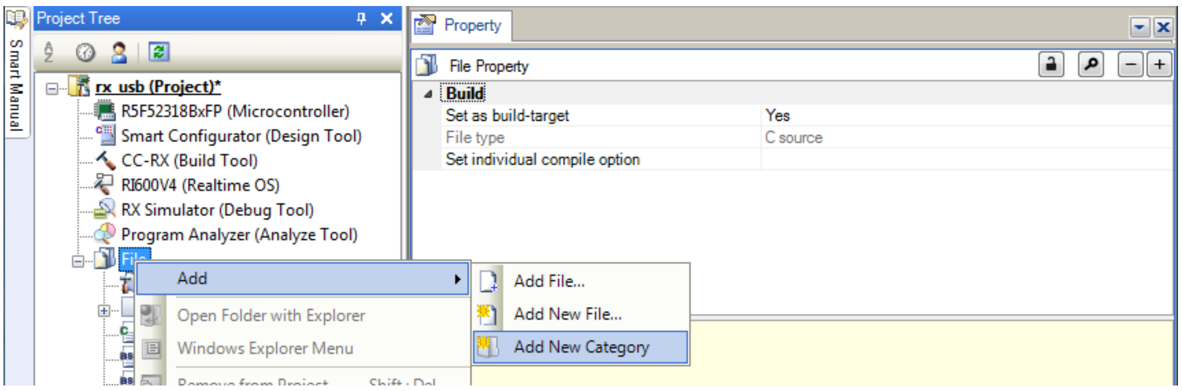
Current section setting:
B_1,R_1,B_2,R_2,B,R,SU,SI/4,PResetPRG/FFC00000,C_1,C_2,C,C$DSEC,C$BSEC,C$INIT,C$VTBL,C$VECT,D_1,D_2,D,P,PIntPRG,W_1,W_2,W,L/FFC00100,EXCEPTVECT/FFFFFF80,RESETVECT/FFFFFFFC
New section setting:
SI,SURI_STACK,B*,R*/0x00000004,INTERRUPT_VECTOR,P*,C*,D*,W*,L/0xFFC00000,FIX_INTERRUPT_VECTOR/0xFFFFFF80

[Yes] to change section setting
[No] to keep current section setting
[Cancel] to cancel code generation

☐ Do not show again

6.3 Add the application program and the configuration file

1. Copy the *demo_src* folder in this package to the *"<ProjectDir>¥src"* folder.
2. Copy the RI600V4 configuration file (.cfg file) to *"<ProjectDir>"* folder.
3. Select "File" in the "Project Tree" and click the right button. Select [Add] → [Add New Category] and create the category to store the application program. Then select [Add File] and register the application program and the configuration file which are copied at the above 2.



Build	
Set as build-target	Yes
File type	C source
Set individual compile option	

Note:

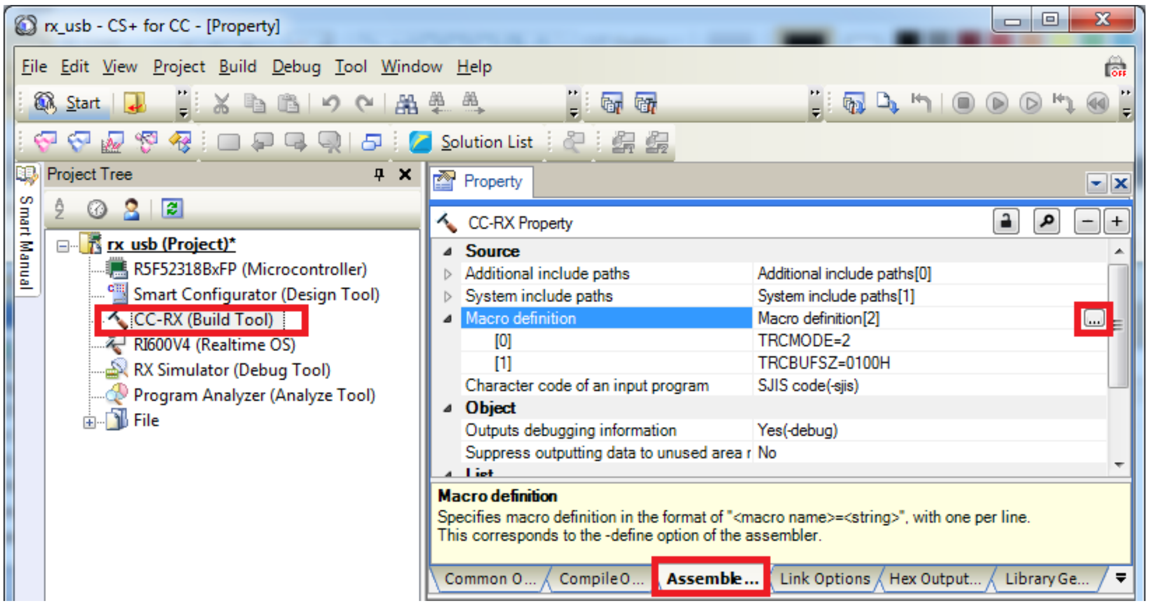
Remove the "task.c" file and "sample.cfg" created in "*<ProjectDir>*" folder by CS+.

6.4 Remote Macro Definition

Remove these macros since the following macros is defined in the new created project.

Select [CC-RX(Build Tool)] → [Assemble Options] tab, remove the following macros.

1. TRCMODE = 2
2. TRCBUFSZ = 0100H


rx_usb - CS+ for CC - [Property]
File Edit View Project Build Debug Tool Window Help
Start
Solution List
Project Tree
Smart Manual
rx_usb (Project)*
R5F52318BxFP (Microcontroller)
Smart Configurator (Design Tool)
CC-RX (Build Tool)
RI600V4 (Realtime OS)
RX Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
File
Property
CC-RX Property
Common O...
Compile O...
Assemble...
Link Options
Hex Output...
Library Ge...

Source	
Additional include paths	Additional include paths[0]
System include paths	System include paths[1]
Macro definition	Macro definition[2]
[0]	TRCMODE=2
[1]	TRCBUFSZ=0100H
Character code of an input program	SJIS code(-sjis)
Object	
Outputs debugging information	Yes(-debug)
Suppress outputting data to unused area r	No

Macro definition
Specifies macro definition in the format of "<macro name>=<string>", with one per line.
This corresponds to the -define option of the assembler.

6.5 Build Execution

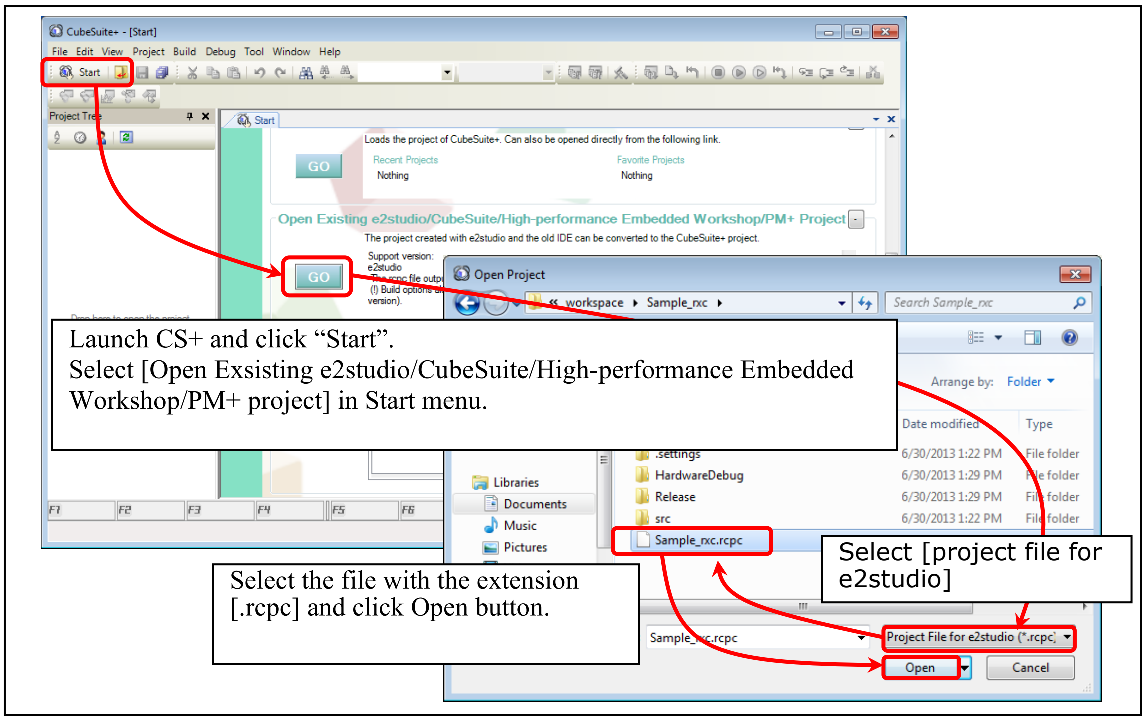
Excecute the build and generate the binary target program.

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

The HHID contains a project only for e^2 studio. When you use the HHID with CS+, import the project to CS+ by following procedures.

[Note]

1. Uncheck the checkbox Backup the project composition files after conversion in Project Convert Settings window.
2. The following method is not supported when using RI600V4. Refer to chapter **6, Using RI600V4 project with CS+**.

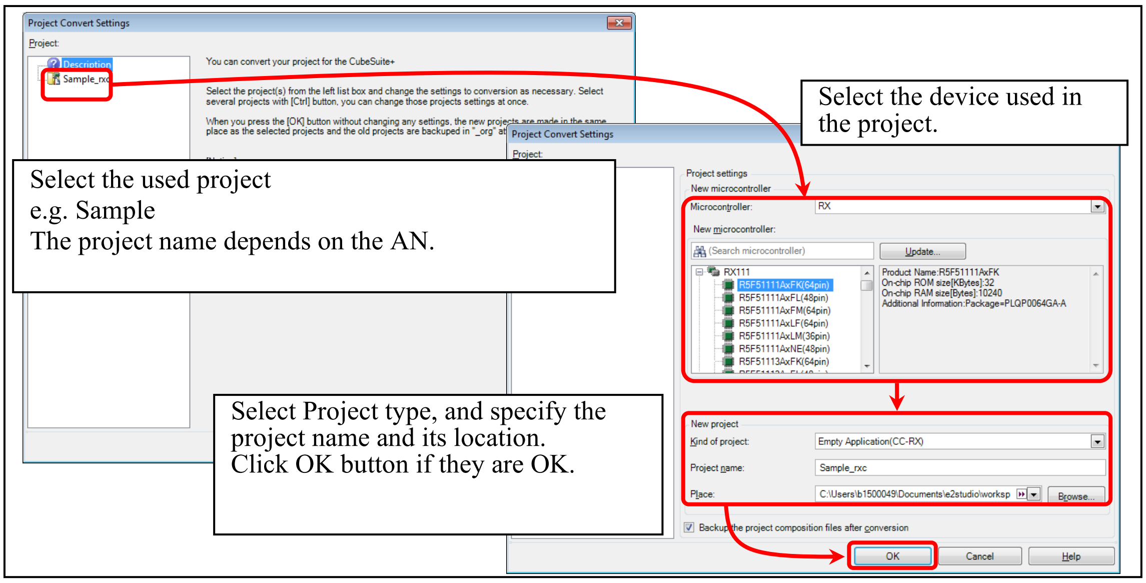


Launch CS+ and click “Start”.
Select [Open Exsisting e2studio/CubeSuite/High-performance Embedded Workshop/PM+ project] in Start menu.

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

Select [project file for e2studio]

CubeSuite+ - [Start]
File Edit View Project Build Debug Tool Window Help
Start
Project Tree
GO
Loads the project of CubeSuite+. Can also be opened directly from the following link.
Recent Projects
Nothing
Favorite Projects
Nothing
Open Existing e2studio/CubeSuite/High-performance Embedded Workshop/PM+ Project
The project created with e2studio and the old IDE can be converted to the CubeSuite+ project.
Open Project
workspace ▸ Sample_rxc ▸
Search Sample_rxc
Arrange by: Folder
Date modified
Type
.settings
HardwareDebug
Release
src
Sample_rxc.rcpc
6/30/2013 1:22 PM
6/30/2013 1:29 PM
File folder
Libraries
Documents
Music
Pictures
Sample_rxc.rcpc
Project File for e2studio (*.rcpc)
Open
Cancel



Select the device used in the project.

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

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

Project Convert Settings
Project:
Description
Sample_rxc
You can convert your project for the CubeSuite+
Select the project(s) from the left list box and change the settings to conversion as necessary. Select several projects with [Ctrl] button, you can change those projects settings at once.
Project settings
New microcontroller
Microcontroller: RX
New microcontroller:
(Search microcontroller)
Update...
RX111
R5F51111AxFK(64pin)
R5F51111AxFL(48pin)
R5F51111AxFM(64pin)
R5F51111AxLF(64pin)
R5F51111AxLM(36pin)
R5F51111AxNE(48pin)
R5F51113AxFK(64pin)
Product Name:R5F51111AxFK
On-chip ROM size[KBytes]:32
On-chip RAM size[Bytes]:10240
Additional Information:Package=PLQP0064GA-A
New project
Kind of project: Empty Application(CC-RX)
Project name: Sample_rxc
Place: C:\Users\b1500049\Documents\e2studio\worksp
Browse...
Backup the project composition files after conversion
OK
Cancel
Help

Figure 7-1 Using the e^2 studio project with CS+

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry/

Revision Record

Rev.	Date	Descriptions	
		Page	Summary
1.00	Dec 1, 2014	—	First Edition Issued.
1.01	Jun 1, 2015	—	RX231 is added in Target Device.
1.02	Dec 28, 2015	—	Upgrading of this USB driver by upgrading of "USB Basic Mini Firmware (R01AN2166)".
1.10	Nov 30, 2018	—	1. The following chapter has beed added. (1). 3.1.2 RSK/RSSK Setting 2. The following chapter has beed changed. (1). 4. Sample Application
1.12	Jun 30, 2019	—	RX23W is added in Target Device.
1.20	Jun 1, 2020	—	Supported the real-time OS.
1.30	Jul 31, 2024	—	RX261 is added in Target Device.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.