APPLICATION NOTE

RX Family

R01AN2236EJ0131
Rev.1.31
Mar 1, 2021

Sample Program using USB Host Human Interface Device Class Driver (HHID) 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 Host and Peripheral Driver Firmware Integration Technology Application Note” (Document number: R01AN2025) together with the user’s manual for each MCU (Hardware). In addition, also refer to the " USB Host Human Interface Device Class Driver (HHID) Firmware Integration Technology Application Note” (Document number: R01AN2028), if necessary. “USB Basic Host and Peripheral Driver Firmware Integration Technology Application Note” (Document number: R01AN2025) is located in the "reference_documents" folder within the package.

Target Device

RX65N/RX651 Group
RX64M Group
RX71M Group
RX66T Group
RX72T Group
RX72M Group
RX66N Group
RX72N Group
RX671 Group

The operation of this program has been confirmed using the Renesas Starter Kits (RSK).

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
RX Family Board Support Package Module Firmware Integration Technology	r_bsp
RX Family USB Basic Host and Peripheral Driver Firmware Integration Technology	r_usb_basic
RX Family USB Host Human Interface Devices Class Driver(HHID) Firmware Integration Technology	r_usb_hhid

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 following is the operating confirmation environment of this program.

Table 1-2 Operating Confirmation Environment

Item	Contents
C compiler	Renesas Electronics C/C++ compiler for RX Family V.3.03.00
	Compile Option：-lang = c99
Real-Time OS	FreeRTOS V.10.0.0 RI600V4
Endian	Little Endian, Big Endian
USB Driver Revision Number	Rev.1.31
Using Board	Renesas Starter Kits for RX64M Renesas Starter Kits for RX71M Renesas Starter Kits for RX65N, Renesas Starter Kits for RX65N-2MB Renesas Starter Kits for RX72T Renesas Starter Kits for RX72M Renesas Starter Kits for RX72N Renesas Starter Kits for RX671

1.5 Terms and Abbreviations

APL	:	Application program
HCD	:	Host Control Driver for USB-BASIC-FW
HHID	:	Host Human Interface Device
HID	:	Human Interface Device Class
MGR	:	Peripheral Device State Manager for HCD
Non-OS	:	USB Driver for OS-less
RSK	:	Renesas Starter Kits
RTOS	:	USB Driver for the real-time OS
USB-BASIC-FW	:	USB Basic Host and Peripheral Driver

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.

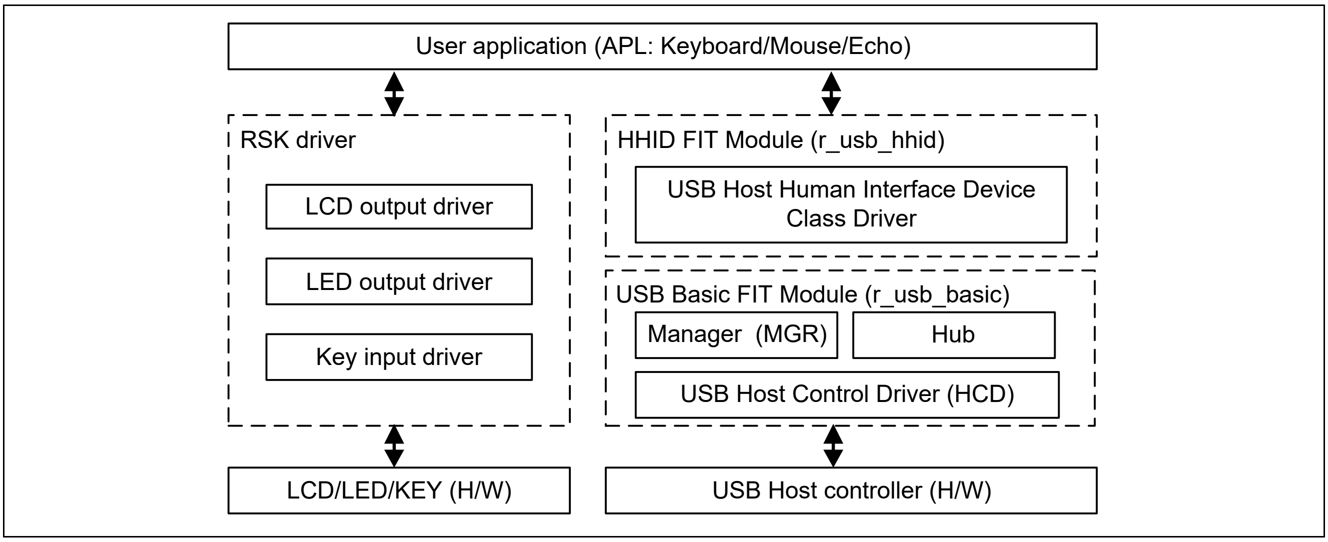


Figure 2-1 Moudle Configuration

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)	HID class driver ・ Interprets requests from the HID device. ・ Reports switch manipulation information from the APL to the HID device via the HCD.
HCD/MGR/Hub (r_usb_basic)	USB Host hardware 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.

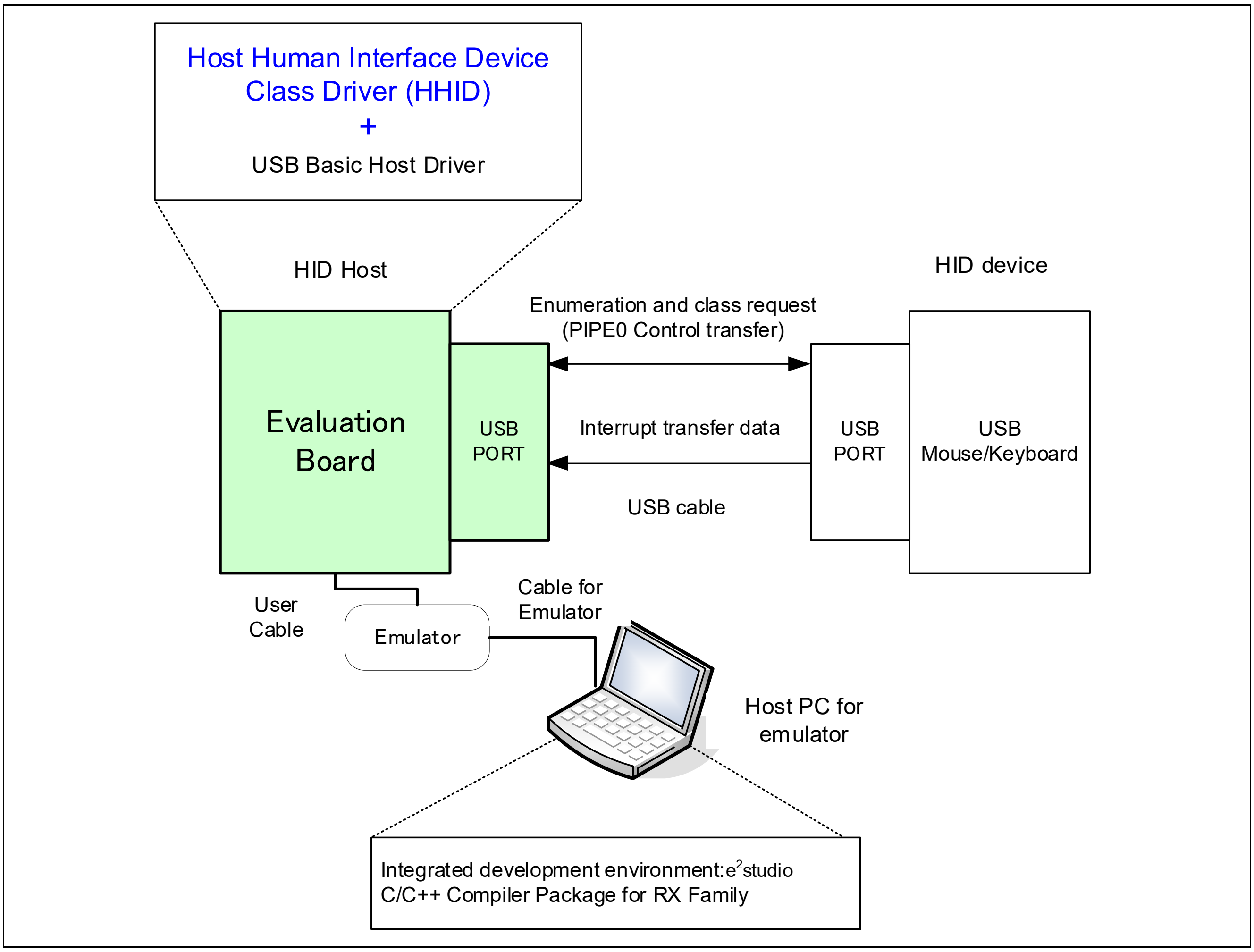


Figure 3-1 Example Operating Environment

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
RX65N	RSK+RX65N, RSK+RX65N-2MB
RX64M	RSK+RX64M
RX71M	RSK+RX71M
RX72T	RSKRX72T
RX72M	RSK+RX72M
RX72N	RSK+RX72N
RX671	RSK+RX671

3.1.2 RSK Setting

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

Table 3-2 **RSK Setting**

RSK	Jumper Setting
RSK+RX65N	J8: Shorted Pin1-2
RSK+RX65N_2MB	J7: Shorted Pin1-2 J16: Shorted Pin2-3
RSK+RX64M (USB0)	J2: Shorted Pin1-2 J6: Shorted Pin2-3
RSK+RX64M (USBH)	J7: Shorted Pin1-2 J9: Shorted Pin2-3
RSK+RX71M (USB0)	J1: Shorted Pin1-2 J3: Shorted Pin2-3
RSK+RX71M (USBA)	J4: Shorted Pin1-2 J7: Shorted Pin2-3
RSKRX72T	J13: Shorted Pin1-2
RSK+RX72M	J8: Shorted Pin2-3 J10: Shorted Pin2-3
RSK+RX72N	J7: Shorted Pin2-3 J8: Shorted Pin2-3
RSK+RX671	J8: Shorted Pin2-3 J13: Shorted Pin2-3

Note:

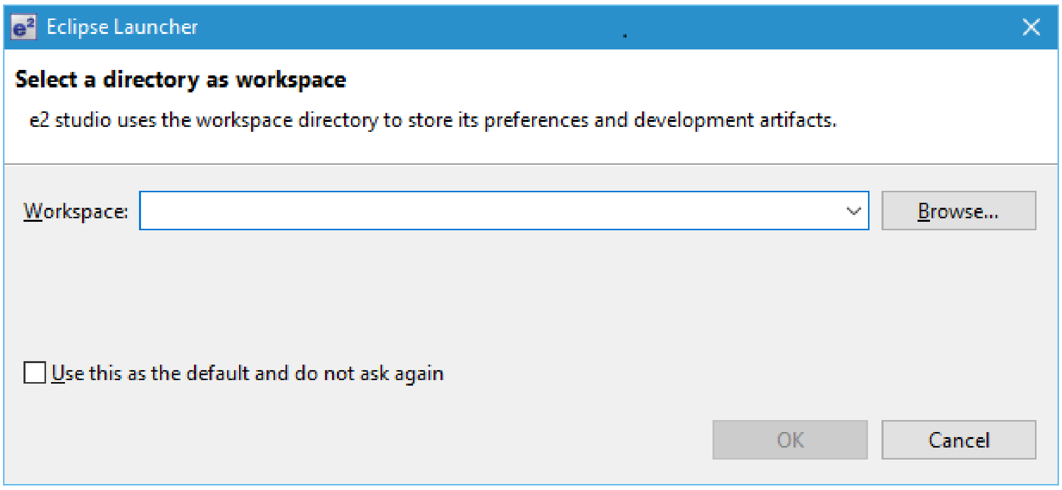
For the detail of RSK setting, refer to the user's manual of RSK.

3.2 Software

1) Setup e2 studio

a) Start e2 studio

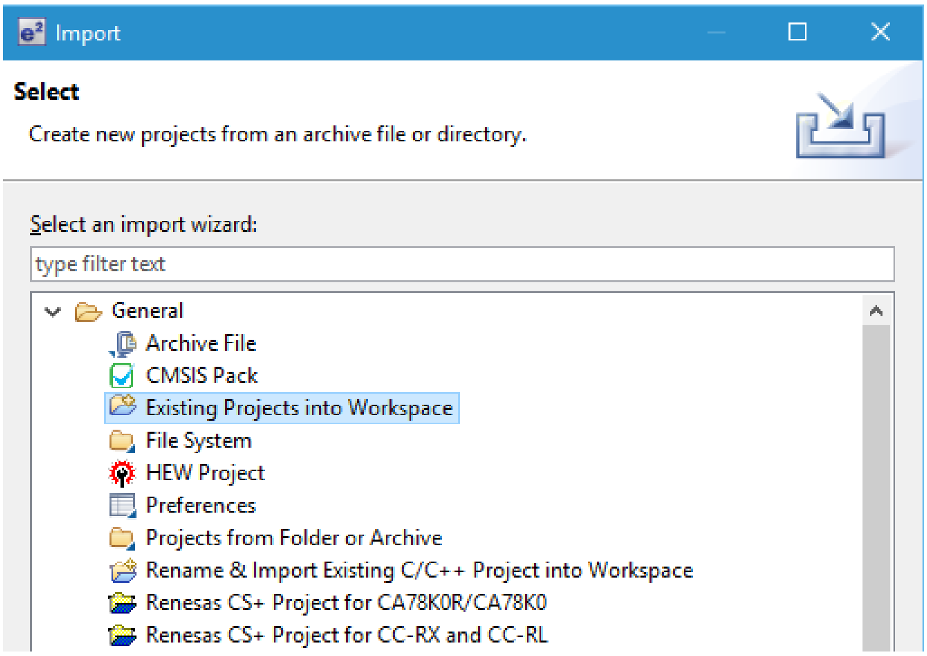
b) If you start up e2 studio at first, the following dialog is displayed. Specify the folder to store the project in this dialog.



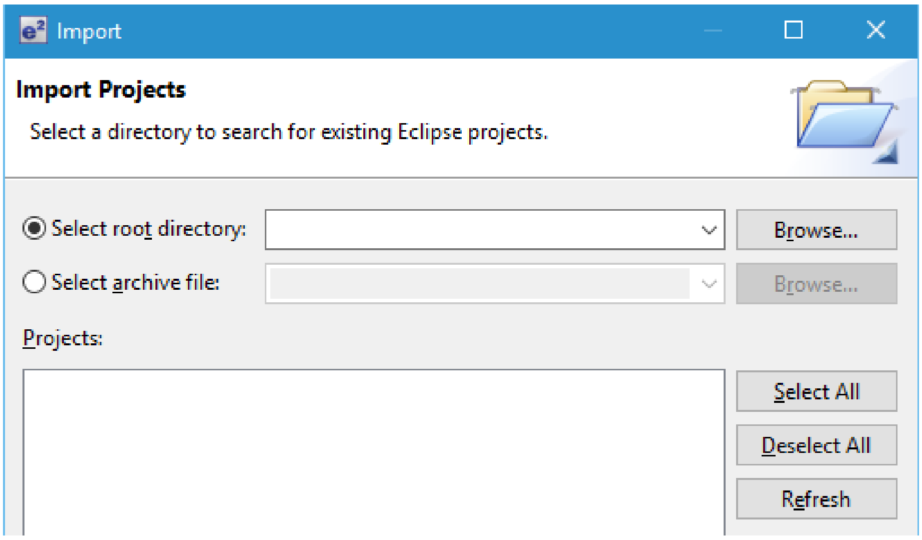
2) Import the project to the workspace

a) Select [File] > [Import]

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



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. Data received from the HID device is read and discarded.

[Note]

Up to three HID devices can be connected to a single USB module by using a USB hub.

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. 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.

[Note]

Up to three HID devices can be connected to a single USB module by using a USB hub.
(In this mode, up to two HID devices can be connected when using RX63N/RX631.)

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 is sent unmodified back to the HID device.

[Note]

1. Loopback processing is possible only when an HID device that supports interrupt out transfer is connected.
2. In this mode only one HID device can be connected to a single USB module.

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) 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:

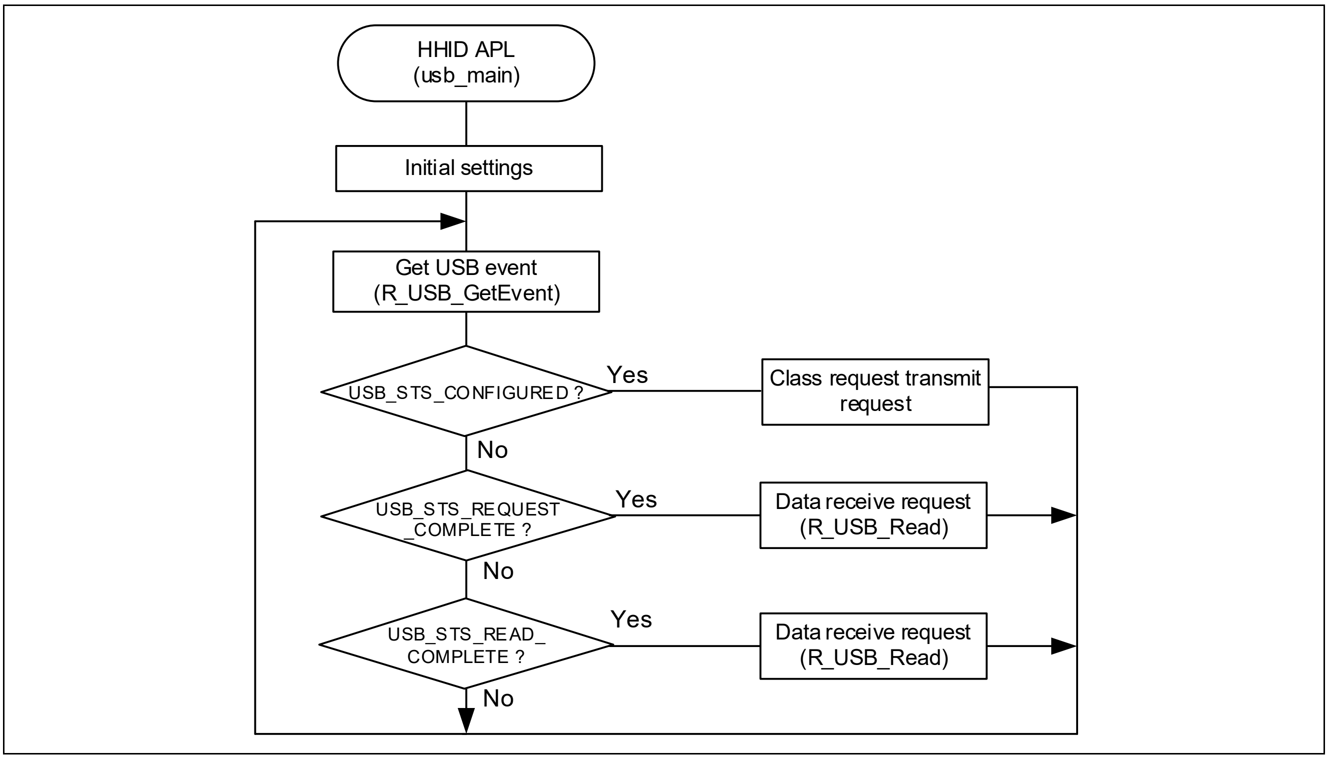


Figure 4-1 Main Loop (Normal mode)

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 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.**

4.2.4 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:

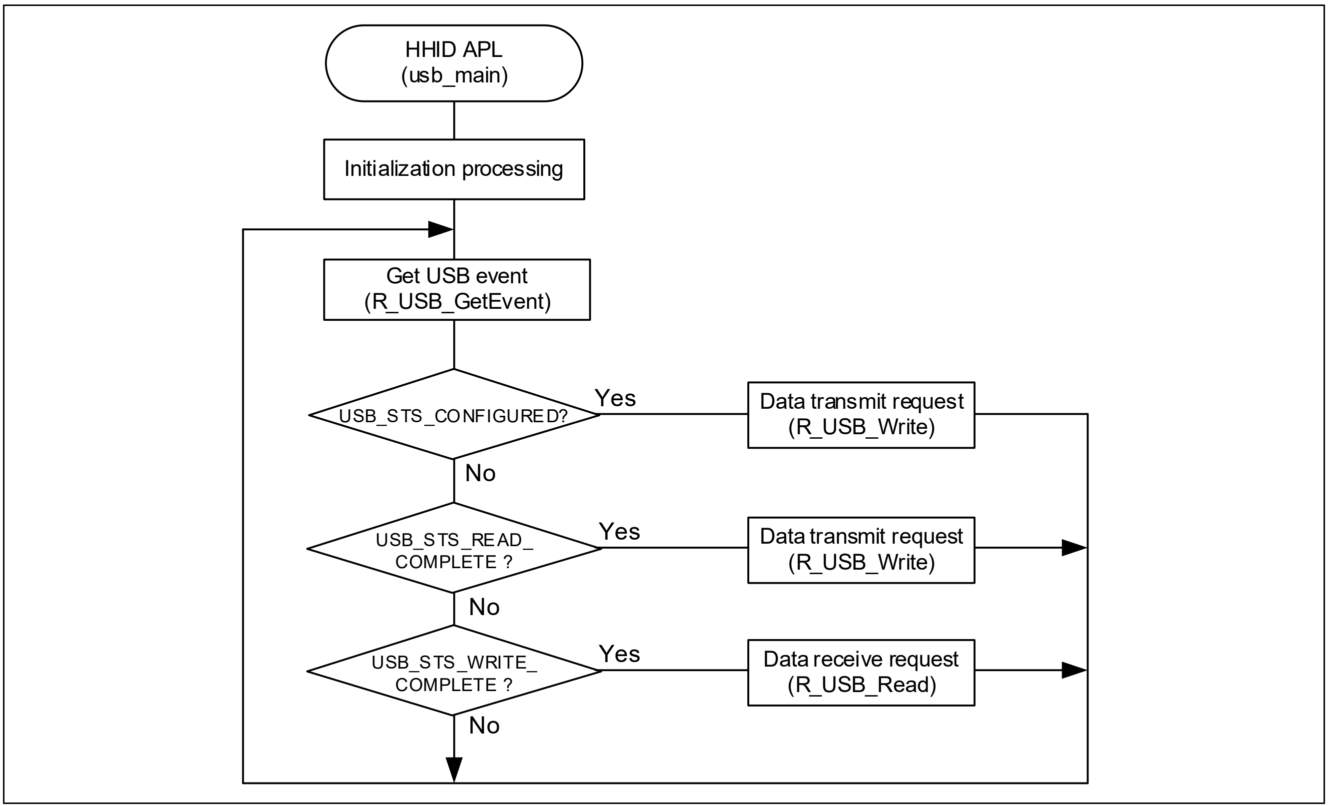


Figure 4-2 Main Loop (Echo mode)

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.

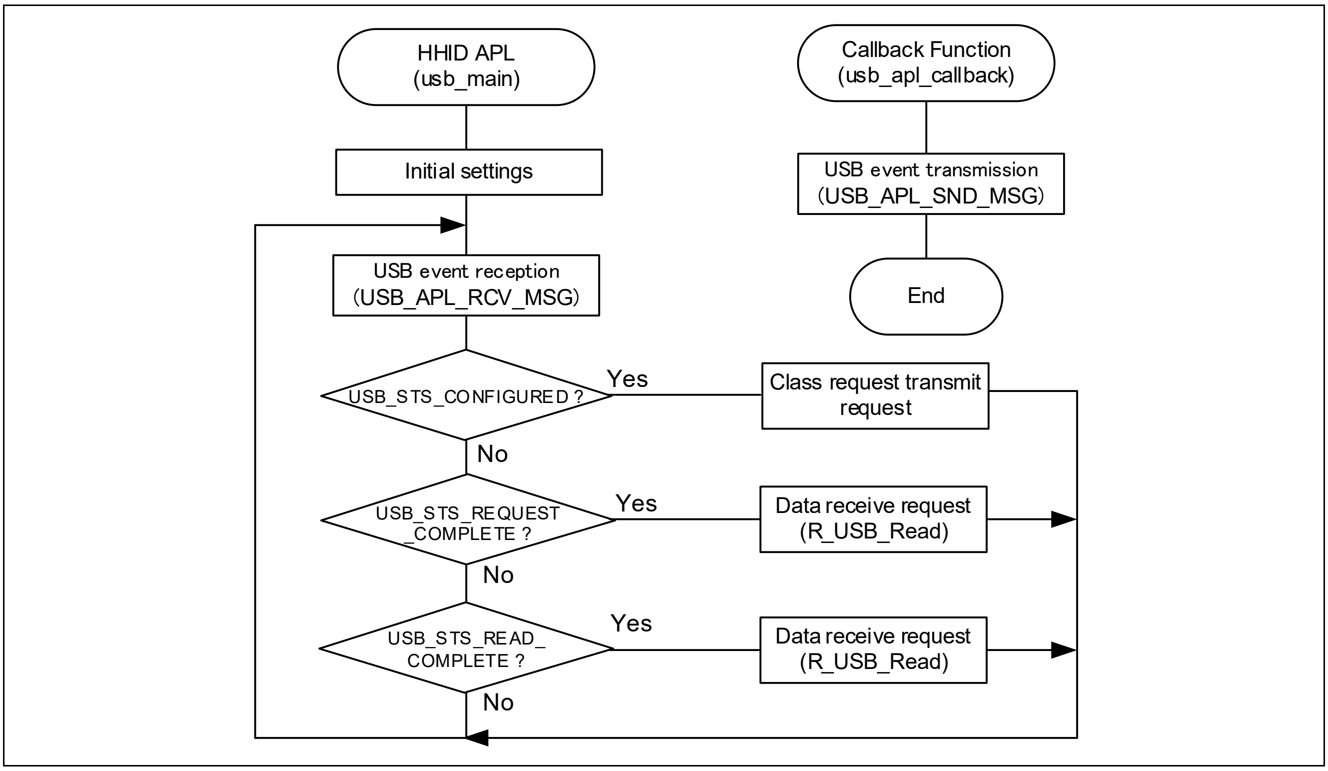


Figure 4-3 Main Loop (Normal mode)

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,**

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:

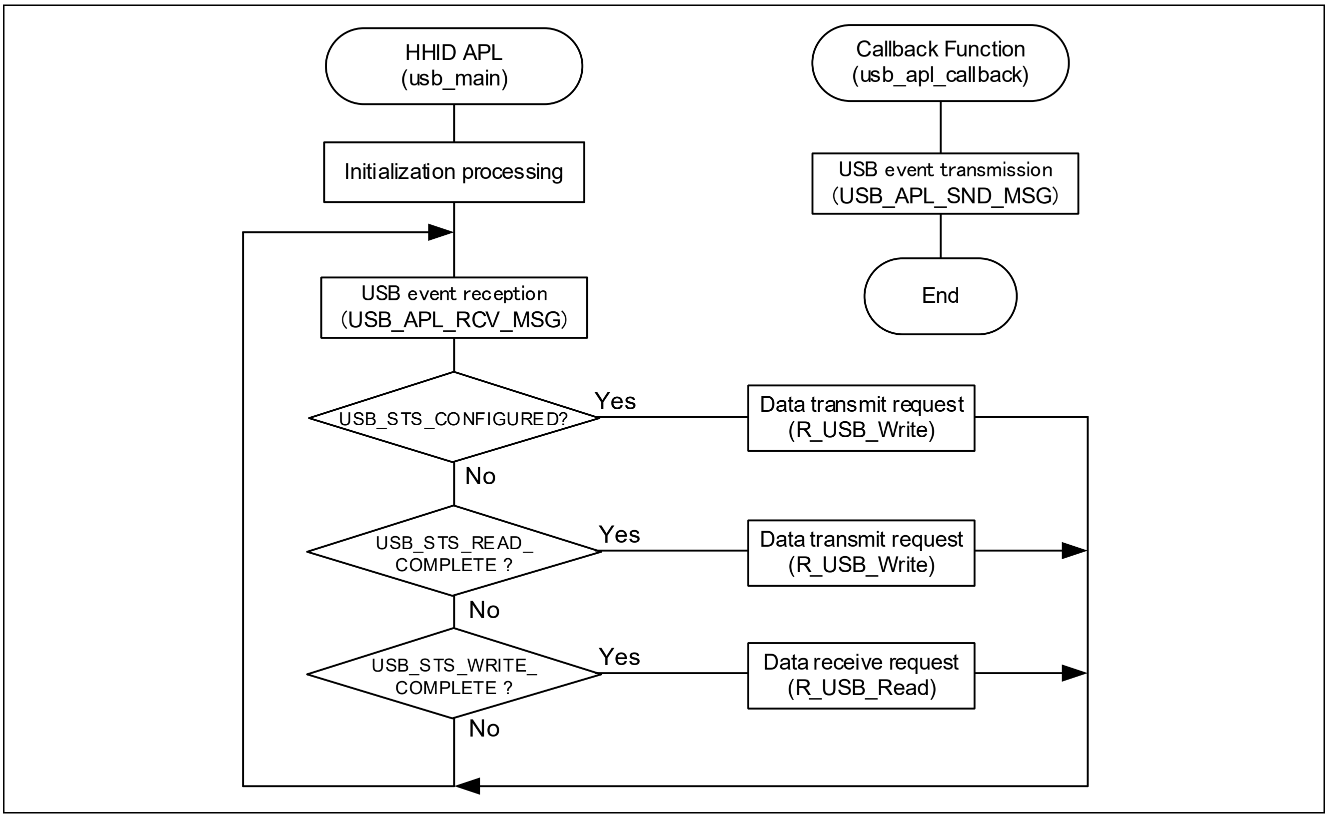


Figure 4-4 Main Loop (Echo mode)

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/SW3 transitions the HID device to the suspend state.
2. When the HID device is in the suspend state, pressing SW2/SW3 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
	Switch3(SW3)	Changes the status of the HID device connected to the USB1 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, refer to the instruction manual of the RSK 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. USE_USBIP Definition

Specify the module number of the USB module you are using.

```
#define   USE_USBIP   USE_USBIP0                // Specify USB_IP0.
#define   USE_USBIP   USE_USBIP1                // Specify USB_IP1.
#define   USE_USBIP   (USE_USBIP1|USE_USBIP0)   // Specify USB_IP1 and USB_IP0
```

[Note]

You can specify *USE_USBIP1* when using RX64M or RX71M. Specify *USE_USBIP0* when using the MCU other than RX64M and RX71M.

2. 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
```

3. USB_SUPPORT_MULTI Definition

Please specify *USB_APL_ENABLE* to *USB_SUPPORT_MULTI* definition when connecting the multiple HID devices at the same time by using USB Hub etc.

```
#define    USB_SUPPORT_MULTI    USB_APL_DISABLE    // Do not conntect the multiple HID devices
#define    USB_SUPPORT_MULTI    USB_APL_ENABLE     // Conntect the multiple HID devices
```

4. 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
```

5. 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 Host and Peripheral Driver Firmware Integration Technology* (Document number: R01AN2025EJ).

4.6 Connecting Multiple HID Devices

Refer to the following sample programs for reference when developing application programs that connect with multiple HID devices using a USB hub, etc.

1. Normal mode

   r_usb_hhid_apl_multi.c

2. Demo mode

   r_usb_hhid_demo_multi.c

3. Echo mode (Supporting only RX64M and RX71M)

   r_usb_hhid_echo_apl_multi.c

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]

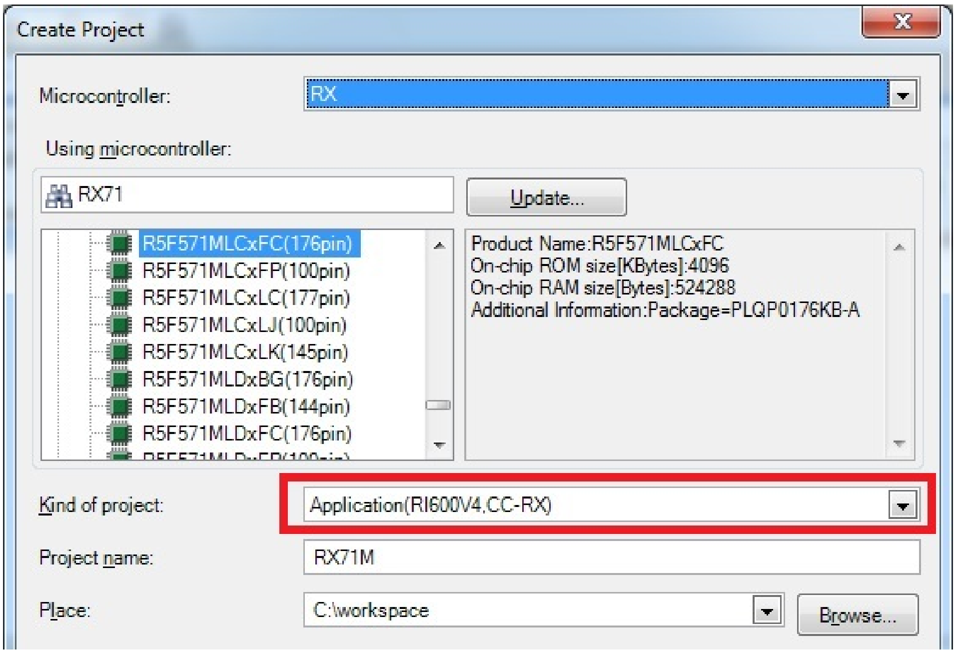
The data format used for report protocol data transfers must conform to the report descriptor. This driver does not acquire and analyze the report descriptor; it determines the report format according to whether the interface protocol code is “keyboard” or “mouse.” The user should make changes as necessary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



6.2 Launch Smart Configurator

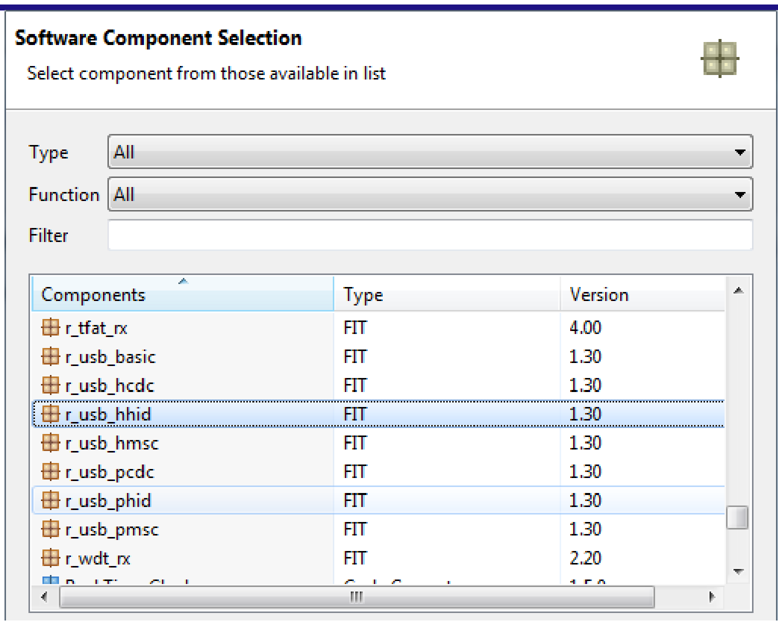
1. Clock Setting (Select "Clocks" tab)

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

2. Component Setting (Select "Components" tab)

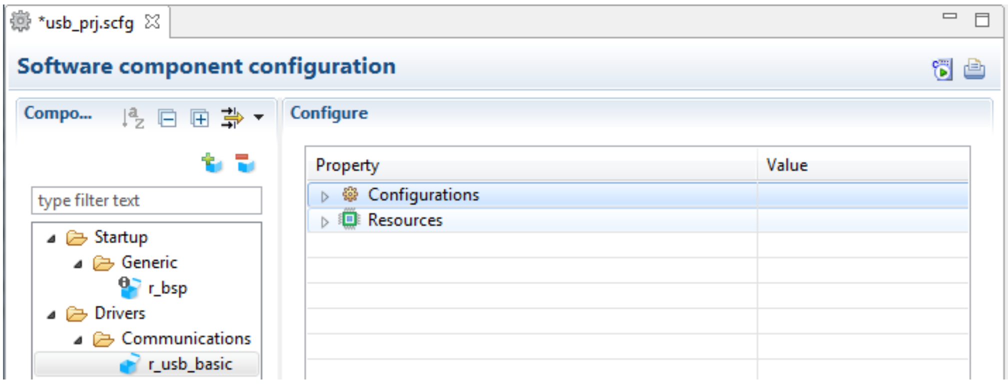
(1). Import the USB FIT module

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



(2). Configuration Setting

a. r_usb_basic



(a). Configurations

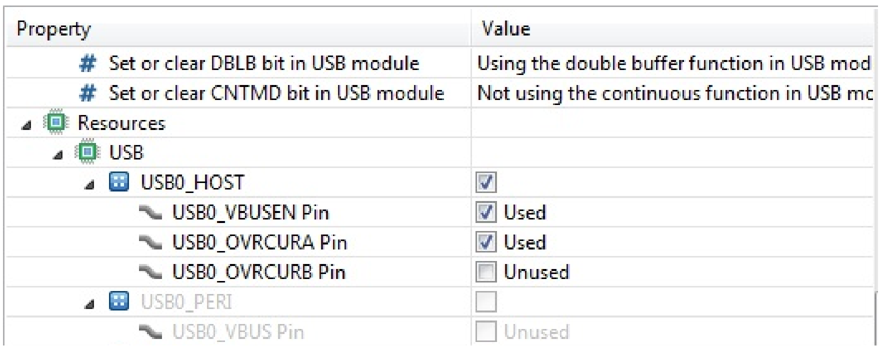
Set each item according to the user system.
For the detail of each item, refer to chapter "Configuration" in *USB Basic Host and Peripheral Driver Firmware Integration Technology* application note (Document number: R01AN2025).

(b). Resources

Check the following check box.

i. *USBx_VBUSEN* pin

ii. *USBx_OVRCURA* pin or *USBx_OVRCURB* pin

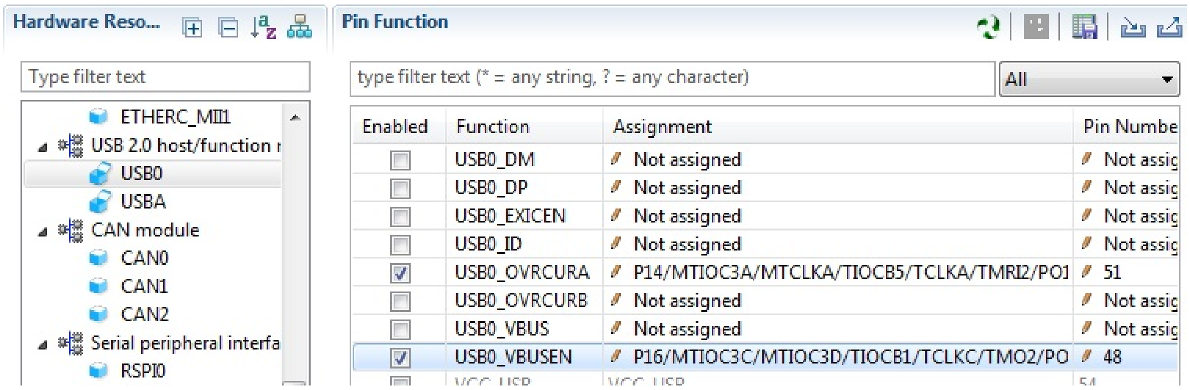


b. r_usb_hhid

Refer to chapter "Configuration" in *USB Host Human Interface Devices Class Driver (HHID) Firmware Integration Technology* application note (Document number: R01AN2028).

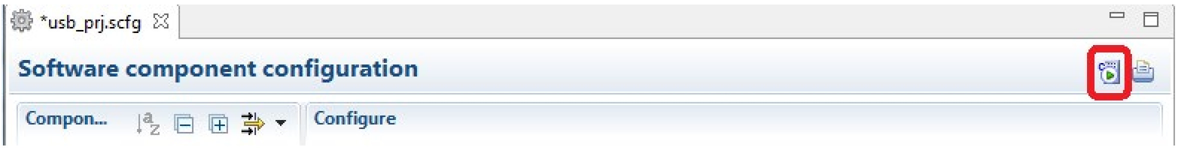
3. Pin Setting (Select "Pins" tab)

Select the port for USB pin match the user system.



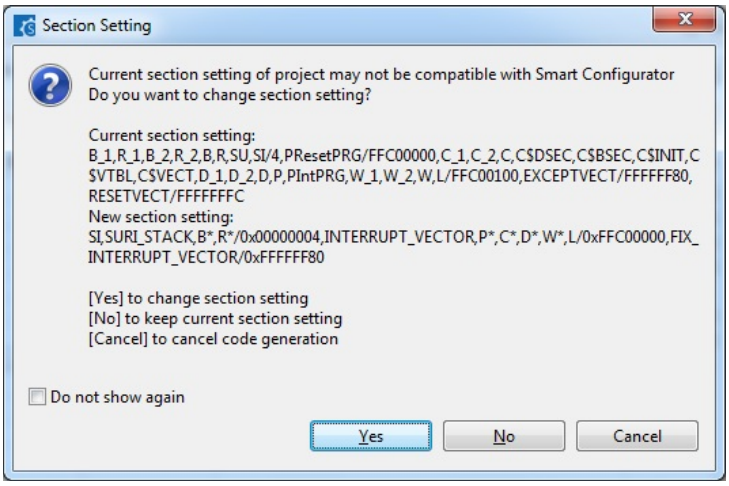
4. Generate Code

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:

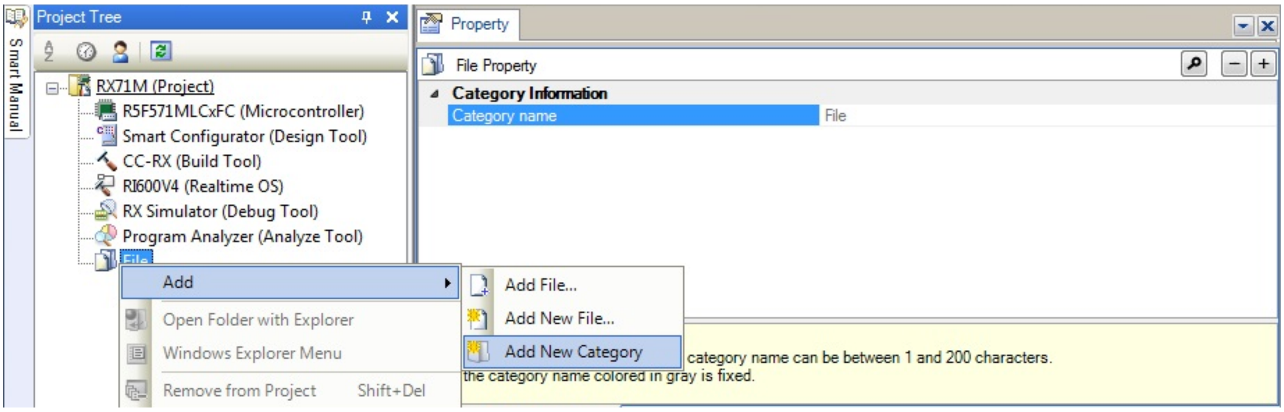
Select "Yes" if the following dialog box is displayed.



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.



Note:

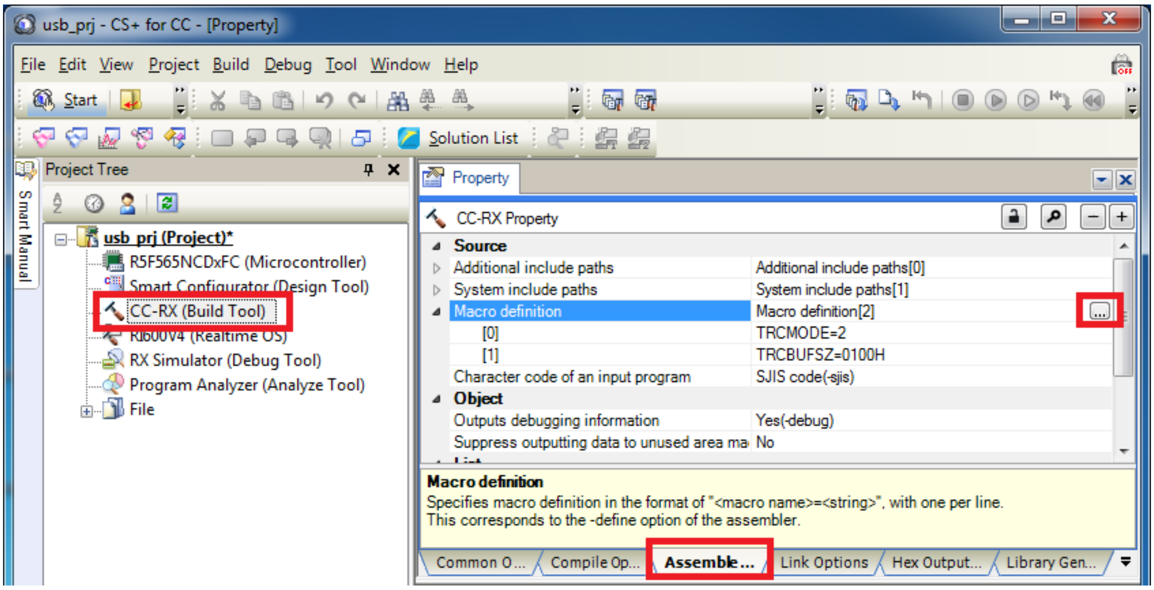
Remove the "task.c" file and "sample.cfg" created in "<*ProjectDir*>" folder by CS+.

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



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+**.

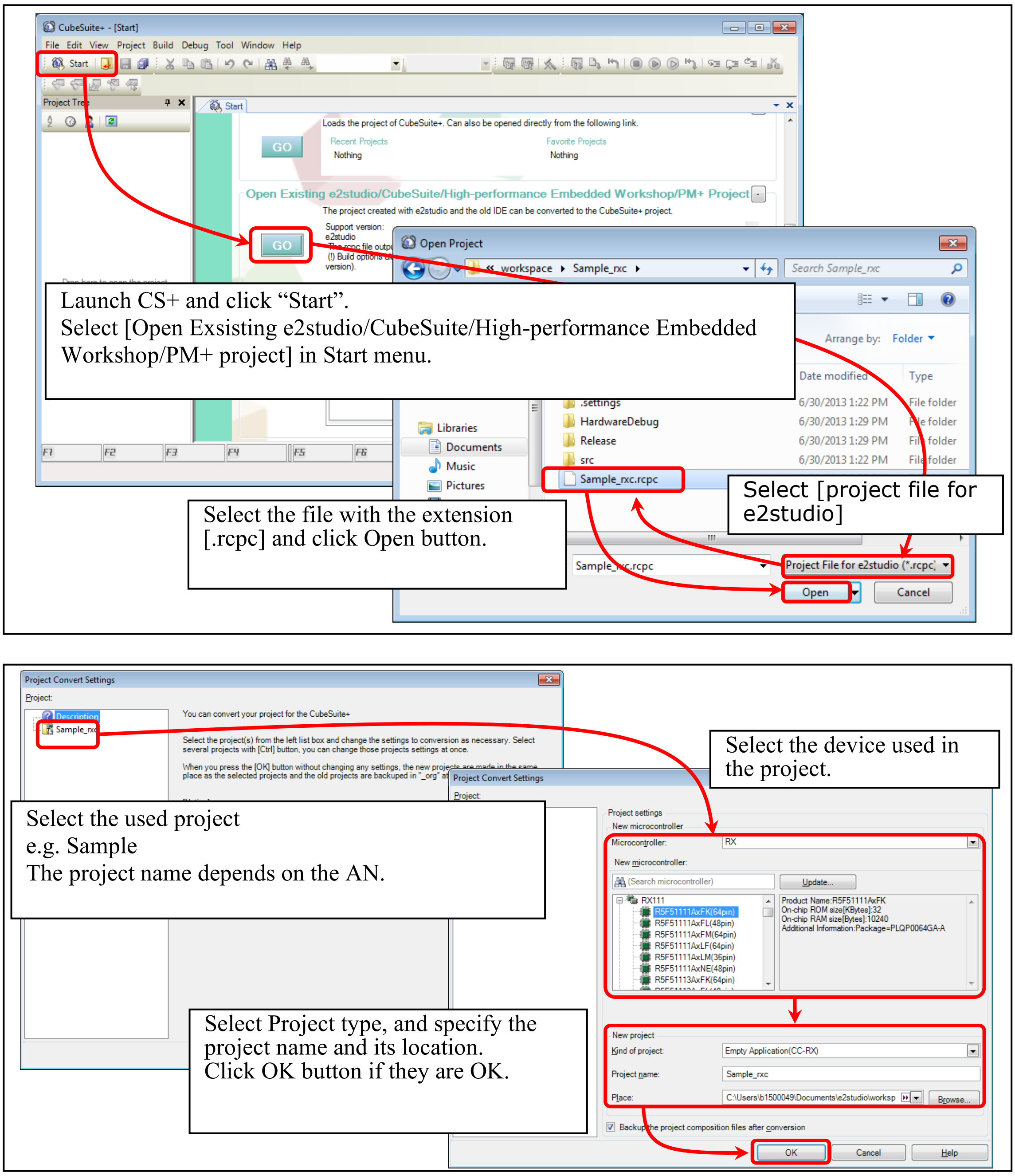


Figure 7-1 Using the e^2 studio project with CS+

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	Oct 16, 2014	—	First Edition Issued.
1.10	Dec 26, 2014	—	RX71M is added in Target Device.
1.11	Sep 30, 2015	—	RX63N and RX631 are added in Target Device.
1.20	Sep 30, 2016	—	1. RX65N and RX651 are added in Target Device. 2. Supporting USB Host and Peripheral Interface Driver application note.
1.21	Mar 31, 2017	—	The revision of USB Basic driver has been updated.
1.22	Sep 30, 2017	—	Supporting RX65N/RX651-2M
1.23	Mar 31, 2018	—	The revision of USB Basic driver has been updated.
1.24	Dec 28, 2018	—	Supported the real-time OS.
1.25	Apr 16, 2019	—	Added RX66T/RX72T in Target Device.
1.27	Jul 31, 2019	—	1. RX72M is added in Target Device. 2. RX63N is removed from Target Device.
1.30	Mar 1, 2020	—	1. Supported the real time OS (uITRON:RI600V4). 2. Added RX72N/RX66N in Target Device.
1.31	Mar 1, 2021	—	Added RX671 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/.