RX Family

R01AN0399EJ0233
Rev.2.33
Jul 31, 2019

USB Host Human Interface Device Class Driver (HHID)

Introduction

This application note describes USB Host Human Interface Device Class Driver (HHID). This module operates in combination with the USB Basic Host Driver (USB-BASIC-F/W). It is referred to below as the HHID.

Target Device

RX62N/RX621 Group
RX63N/RX631 Group
RX63T Group

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

Related Documents

1. Universal Serial Bus Revision 2.0 specification
   http://www.usb.org/developers/docs/
2. USB Class Definitions for Human Interface Devices Version 1.1
3. HID Usage Tables Version 1.1
   http://www.usb.org/developers/docs/
4. RX62N/RX621 Group User's Manual: Hardware (Document number .R01UH0033)
5. RX63N/RX631 Group User's Manual: Hardware (Document number .R01UH0041)
6. RX63T Group User's Manual: Hardware (Document number .R01UH0238)
7. USB Basic Host and Peripheral Driver Application Note (Document number.R01AN0512)

- Renesas Electronics Website
  http://www.renesas.com/
- USB Devices Page
  http://www.renesas.com/prod/usb/

Contents

1. Overview

The HHID, when used in combination with the USB-BASIC-F/W, operates as a USB host human interface device class driver (HHID).

This driver supports the following functions.

- Data communication with a connected HID device (USB mouse, USB keyboard)
- Issuing of HID class requests to a connected HID device
- Supporting Interrupt OUT transfer.
- HHID can connect maximum 3 HID devices to 1 USB module by using USB Hub.

1.1 Please be sure to read

Please refer to the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note* when creating an application program using this driver.

This document is located in the "**reference_documents**" folder within this package.

1.2 Note

1. 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.
2. Please be sure to use the documentations (*r01an0399ej0232_usb.pdf*, *r01an0512ej0232_usb.pdf*) under the "reference_documents" folder when using RX62N/RX621/ RX63T.

1.3 Limitations

The following limitations apply to the HHID.

1. The HID driver must analyze the report descriptor to determine the report format (This HID driver determines the report format from the interface protocol alone.)
2. One USB Hub is used for each USB module and a maximum of three HID devices can be connected. If your system supports the Interrupt OUT transfer, you cannot connect more than 2 HID devices for each USB module.
3. This driver does not support DMA/DTC transfer.
4. Don't connect Low-speed HID device to HHID.

1.4 Terms and Abbreviations

Terms and abbreviations used in this document are listed below.

APL	:	Application program
HCD	:	Host control driver of
HDCD	:	Host device class driver (device driver and USB class driver)
HHID	:	Host human interface device
HID	:	Human interface device class
HUBCD	:	Hub class sample driver
MGR	:	Peripheral device state manager of HCD
RSK	:	Renesas Starter Kits
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Basic Host Driver for

2. Module Configuration

The HHID comprises the HID class driver and device drivers.

When data is received from the connected USB device, HCD notifies the application. Conversely, when the application issues a request, HCD notifies the USB device.

Figure 2.1 shows the structure of the HHID-related modules. Table 2-1 lists the modules and an overview of each.

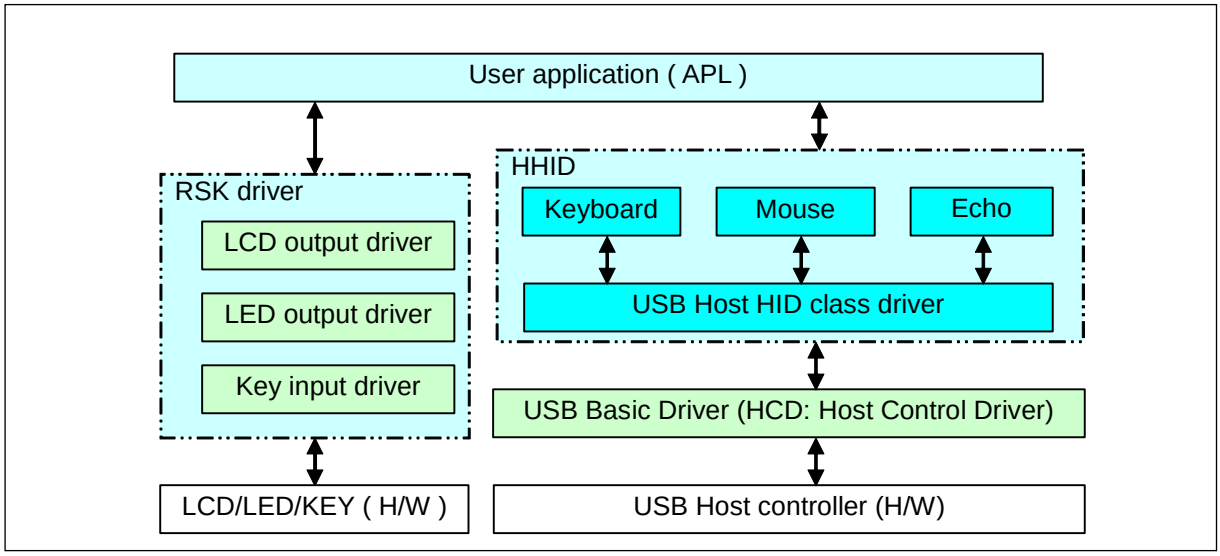


Figure 2.1 Software Configuration

Table 2-1 Module Function Descriptions

Module Name	Description
APL	User application program. Switches initiate communication with HID devices and control suspend/resume. The LCD displays the information received from the HID device.
HHID	The HHID analyzes requests from HID devices. Notifies APL key operation information to the HID host via the HCD.
HCD	USB host Hardware Control Driver

3. API Information

This Driver API follows the Renesas API naming standards.

3.1 Hardware Requirements

This driver requires your MCU support the following features:

- USB

3.2 Operating Confirmation Environment

Table 3-1 shows the operating confirmation environment of this driver.

Table 3-1 Operation Confirmation Environment

Item	Contents
C compiler	Renesas Electronics C/C++ compiler for RX Family V.3.01.00
	Compile Option：-lang = c99
Endian	Little Endian, Big Endian
Using Board	Renesas Starter Kits for RX63N

3.3 Usage of Interrupt Vector

Table 3-2 shows the interrupt vector which this driver uses.

Table 3-2 List of Usage Interrupt Vectors

Device	Contents
RX63N/RX631	USBI0 Interrupt (Vector number: 35) / USBR0 Interrupt (Vector number: 90)

3.4 Header Files

All API calls and their supporting interface definitions are located in r_usb_basic_if.h and r_usb_hhid_if.h.

3.5 Integer Types

This project uses ANSI C99 “Exact width integer types” in order to make the code clearer and more portable. These types are defined in *stdint.h*.

3.6 Compile Setting

For compile settings, refer to chapter **7, Configuration (r_usb_hcdc_config.h)** in this document and chapter "Configuration" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note*.

3.7 ROM / RAM Size

The follows show ROM/RAM size of this driver.

	Checks arguments	Does not check arguments
ROM size	34.2K bytes (Note 3)	33.5K bytes (Note 4)
RAM size	17.8K bytes	17.8K bytes

Note:

1. ROM/RAM size for USB Basic Driver is included in the above size.

2. The default option is specified in the compiler optimization option.
3. The ROM size of “Checks arguments” is the value when *USB_CFG_ENABLE* is specified to *USB_CFG_PARAM_CHECKING* definition in *r_usb_basic_config.h* file.
4. The ROM size of “Does not check arguments” is the value when *USB_CFG_DISABLE* is specified to *USB_CFG_PARAM_CHECKING* definition in *r_usb_basic_config.h* file.

3.8 Argument

For the structure used in the argument of API function, refer to chapter "**Structures**" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note*.

4. Target Peripheral List (TPL)

For the structure used in the argument of API function, refer to chapter " **How to Set the Target Peripheral List (TPL)**" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note*.

5. Human Interface Device Class (HID)

5.1 Basic Functions

This driver complies with the HID class specification. The main functions of this driver are as follows.

(1) HID device access
(2) Class request notifications to the HID device
(3) Data communication with the HID device

5.2 Class Requests (Host to Device Requests)

This driver supports the following class requests.

For the class request processing, refer to chapter "**USB Class Requests**" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note*.

Table 5-1 HID Class Requests

Symbol	Request	Code	Description
a	USB_GET_REPORT	0x01	Receives a report from the HID device
b	USB_SET_REPORT	0x09	Sends a report to the HID device
c	USB_GET_IDLE	0x02	Receives a duration (time) from the HID device
d	USB_SET_IDLE	0x0A	Sends a duration (time) to the HID device
e	USB_GET_PROTOCOL	0x03	Reads a protocol from the HID device
f	USB_SET_PROTOCOL	0x0B	Sends a protocol to the HID device
	USB_GET_REPORT_DESCRIPTOR	Standard	Transmits report descriptor
	USB_GET_HID_DESCRIPTOR	Standard	Transmits an HID descriptor

The class request data formats supported in this driver are described below.

a). GetReport Request Format

Table 5-2 shows the GetReport request format.
Receives a report from the device in a control transfer.

Table 5-2 GetReport Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	GET_REPORT (0x01)	ReportType & ReportID	Interface	ReportLength	Report

b). SetReport Request Format

Table 5-3 shows the SetReport request format.
Sends report data to the device in a control transfer.

Table 5-3 SetReport Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_REPORT (0x09)	ReportType & ReportID	Interface	ReportLength	Report

c). GetIdle Request Format

Table 5-4 shows the GetIdle request format.
Acquires the intarval time of the report notification (interrupt transfer). Idle rate is indicated in 4 ms units.

Table 5-4 GetIdle Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	GET_IDLE (0x02)	0(Zero) & ReportID	Interface	1(one)	Idle rate

d). SetIdle Request Format

Table 5-5 shows the SetIdle request format.
Sets the interval time of the report notification (interrupt transfer). Duration time is indicated in 4 ms units.

Table 5-5 SetIdle Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_IDLE (0x0A)	Duration & ReportID	Interface	0(zero)	Not applicable

e). GetProtocol Request Format

Table 5-6 shows the GetProtocol request format.
Acquires current protocol (boot protocol or report protocol) settings.

Table 5-6 GetProtocol Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	GET_PROTOCOL (0x03)	0(Zero)	Interface	1(one)	0(BootProtocol) / 1(ReportProtocol)

f). SetProtocol Request Format

Table 5-7 shows the SetProtocol request format.
Sets protocol (boot protocol or report protocol).

Table 5-7 SetProtocol Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_PROTOCOL (0x03)	0(BootProtocol) / 1(ReportProtocol)	Interface	0(zero)	Not applicable

5.3 HID-Report Format

5.3.1 Receive Report Format

Table 5-8 shows the receive report format used for notifications from the HID device.
Reports are received in interrupt IN transfers or class request GetReport.

Table 5-8 Receive Report Format

Offset	Keyboard Mode	Mouse Mode
Data length	8 Bytes	3 Bytes
0 (Top Byte)	Modifier keys	b0: Button 1 b1: Button 2 b2-7: 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	-

5.3.2 Transmit Report Format

Table 5-9 shows the format of the transmit report sent to the HID device.
Reports are sent in the class request SetReport.

Table 5-9 Transmit Report Format

Offset	Keyboard	Mouse
Data length	1 Byte	Not supported
0 (Top Byte)	b0: LED 0 (NumLock) b1: LED 1(CapsLock) b2: LED 2(ScrollLock) b3: LED 3(Compose) b4: LED 4(Kana)	-
+1 ~ +16	-	-

5.3.3 Note

The report format used by HID devices for data communication is based on the report descriptor. This HID driver does not acquire or analyze the report descriptor; rather, the report format is determined by the interface protocol code.

6. API Functions

The following is Host Human Interface Device Class specific API function.

API	Desription
R_USB_HhidGetType()	Obtains type information for the HID device.
R_USB_HhidGetMxps()	Obtains the max packet size for the HID device.

Note:

Refer to chapter "API" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note* when using the other API.

6.1 R_USB_HhidGetType

Obtains type information for the HID device.

Format

usb_err_t R_USB_HhidGetType(usb_ctrl_t *p_ctrl, uint8_t *p_type)

Arguments

p_ctrl	Pointer to usb_ctrl_t structure area
p_drive	Pointer to the area to store the type information

Return Value

USB_SUCCESS	Successfully completed
USB_ERR_PARA	Parameter error
USB_ERR_NG	Other error

Description

Based on the information assigned to the usb_crtl_t structure (the member *module* and *address*), obtains type information (mouse, keyboard, etc.) for the connected HID device. The type information is set to the area indicated by the second argument (*p_type*). For the type information to be set, see Table 6-1.

Table 6-1 Type Information

Type Information	Description
USB_HID_KEYBOARD	Keyboard
USB_HID_MOUSE	Mouse
USB_HID_OTHER	HID device other than keyboard and mouse

Note

1. Before calling this API, assign the device address of the HID device, and the USB module number (*USB_IP0* or *USB_IP1*) connected to that MSC device, to the members (*address* and *module*) of the usb_crtl_t structure. If there is a problem with what is assigned to these members, then *USB_ERR_PARA* will be the return value.
2. If *USB_NULL* is assigned to the argument (*p_ctrl*), then *USB_ERR_PARA* will be the return value.
3. This function can be called when the USB device is in the configured state. When the API is called in any other state, *USB_ERR_NG* is returned.

Example

```
void usb_application( void )
{
    usb_ctrl_t ctrl;
    uint8_t   type;
                :
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
         {
                :
          case USB_STS_CONFIGURED:
                :
             ctrl.module = USB_IP0;
             ctrl.address = adr;
             R_USB_HhidGetType( &ctrl, &type );
             if( USB_HID_KEYBOARD == type )
             {
                :
             }
                :
          break;
                :
       }
    }
}
```

6.2 R_USB_HhidGetMxps

Obtains the max packet size for the HID device.

Format

usb_err_t R_USB_HhidGetMxps(usb_ctrl_t *p_ctrl, uint16_t *p_mxps, uint8_t dir)

Arguments

p_ctrl	Pointer to usb_ctrl_t structure area
p_mxps	Pointer to the area to store the max packe size
dir	Transfer direction

Return Value

USB_SUCCESS	Successfully completed
USB_ERR_PARA	Parameter error
USB_ERR_NG	Other error

Description

Based on the information assigned to the usb_crtl_t structure (the member *module* and *address*), obtains max packet size for the connected HID device. The max packet size is set to the area indicated by the second argument (*p_type*).

Set the direction (USB_IN / USB_OUT) of the max packet size which the user want to obtain to the third argument (3rd).

Note

1. Before calling this API, assign the device address of the HID device, and the USB module number (*USB_IP0* or *USB_IP1*) connected to that MSC device, to the members (*address* and *module*) of the usb_crtl_t structure. If there is a problem with what is assigned to these members, then *USB_ERR_PARA* will be the return value.
2. If the MCU being used only supports one USB module, then do not assign *USB_IP1* to the member *(module)*. If *USB_IP1* is assigned, then *USB_ERR_PARA* will be the return value.
3. If *USB_NULL* is assigned to the argument (*p_ctrl*), then *USB_ERR_PARA* will be the return value.
4. This function returns *USB_ERR_NG* when the connected HID device does not support the transfer direction set the third argument.
5. This function can be called when the USB device is in the configured state. When the API is called in any other state, *USB_ERR_NG* is returned.

Example

```
void usb_application( void )
{
    uint16_t  mxps;
    usb_ctrl_t ctrl;
                :
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
         {
                :
          case USB_STS_CONFIGURED:
                :
             ctrl.module = USB_IP0;
             ctrl.address = adr;
             R_USB_HhidGetMxps(&ctrl, &mxps, USB_IN);
                :
          break;
                :
        }
    }
}
```

7. Configuration (r_usb_hcdc_config.h)

Please set the following according to your system.

Note:

Be sure to set *r_usb_basic_config.h* file as well. For *r_usb_basic_config.h* file, refer to chapter "**Configuration**" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note*.

1. Setting pipe to be used

Set the pipe number (PIPE6 to PIPE9) to use for Interrupt IN transfer. Do not set the same pipe number. If the USB Hub is being used, then PIPE9 cannot be set as the following definitions.

#define	USB_CFG_HHID_INT_IN	Pipe number (USB_PIPE6 to USB_PIPE9)
#define	USB_CFG_HHID_INT_IN2	Pipe number (USB_PIPE6 to USB_PIPE9)
#define	USB_CFG_HHID_INT_IN3	Pipe number (USB_PIPE6 to USB_PIPE9)

Note:

If no pipe number is required to be set for the definitions of *USB_CFG_HHID_INT_IN2* and *USB_CFG_HHID_INT_IN3*, then set *USB_NULL* as these definitions.

8. Sample Application

8.1 Application Specifications

The following three application programs are provided:

8.1.1 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.

8.1.2 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 two HID devices can be connected in this mode.

8.1.3 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.

8.2 Application Processing

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

8.2.1 Initial Settings

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

8.2.2 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 sends class request *SET_PROTOCOL* 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:

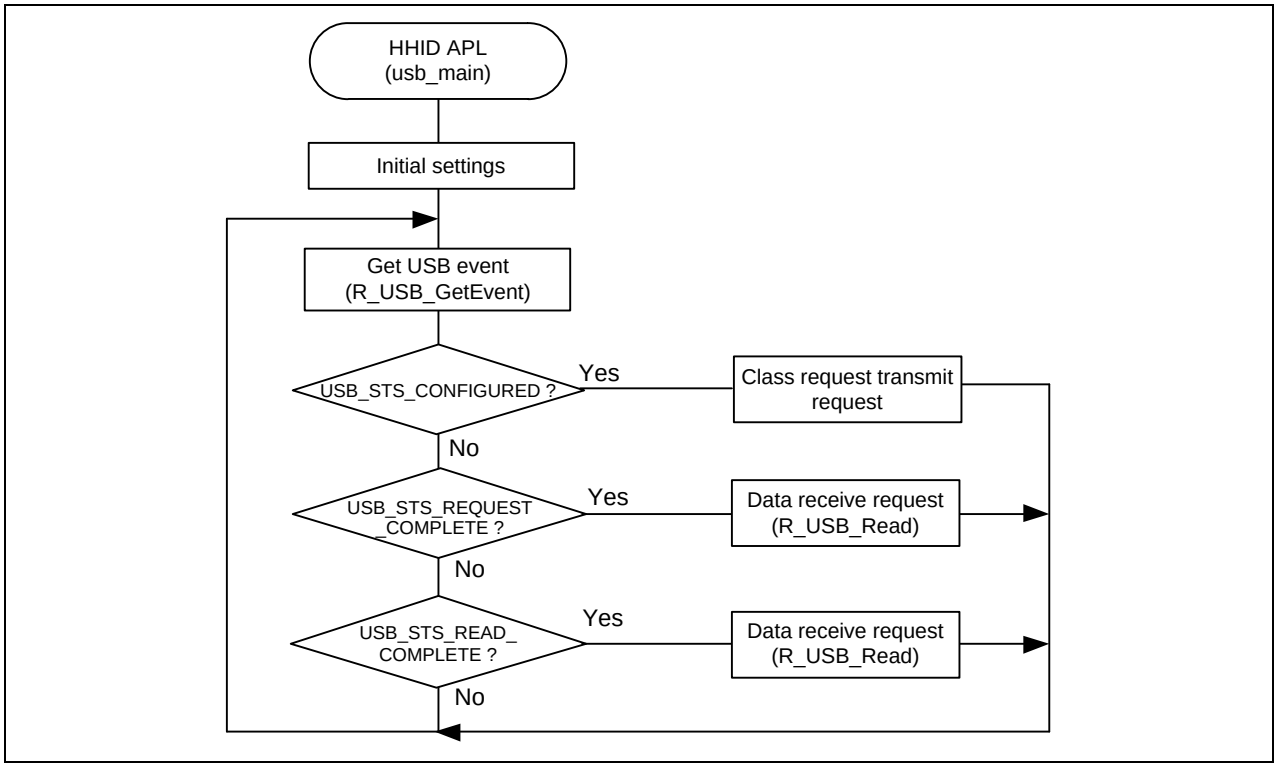


Figure 8-1 Main Loop (Normal mode)

8.2.3 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 8.2.2, 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 **8.3, Switch(Button) Operations and LCD/LED Indications in Demo Mode**.

8.2.4 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:

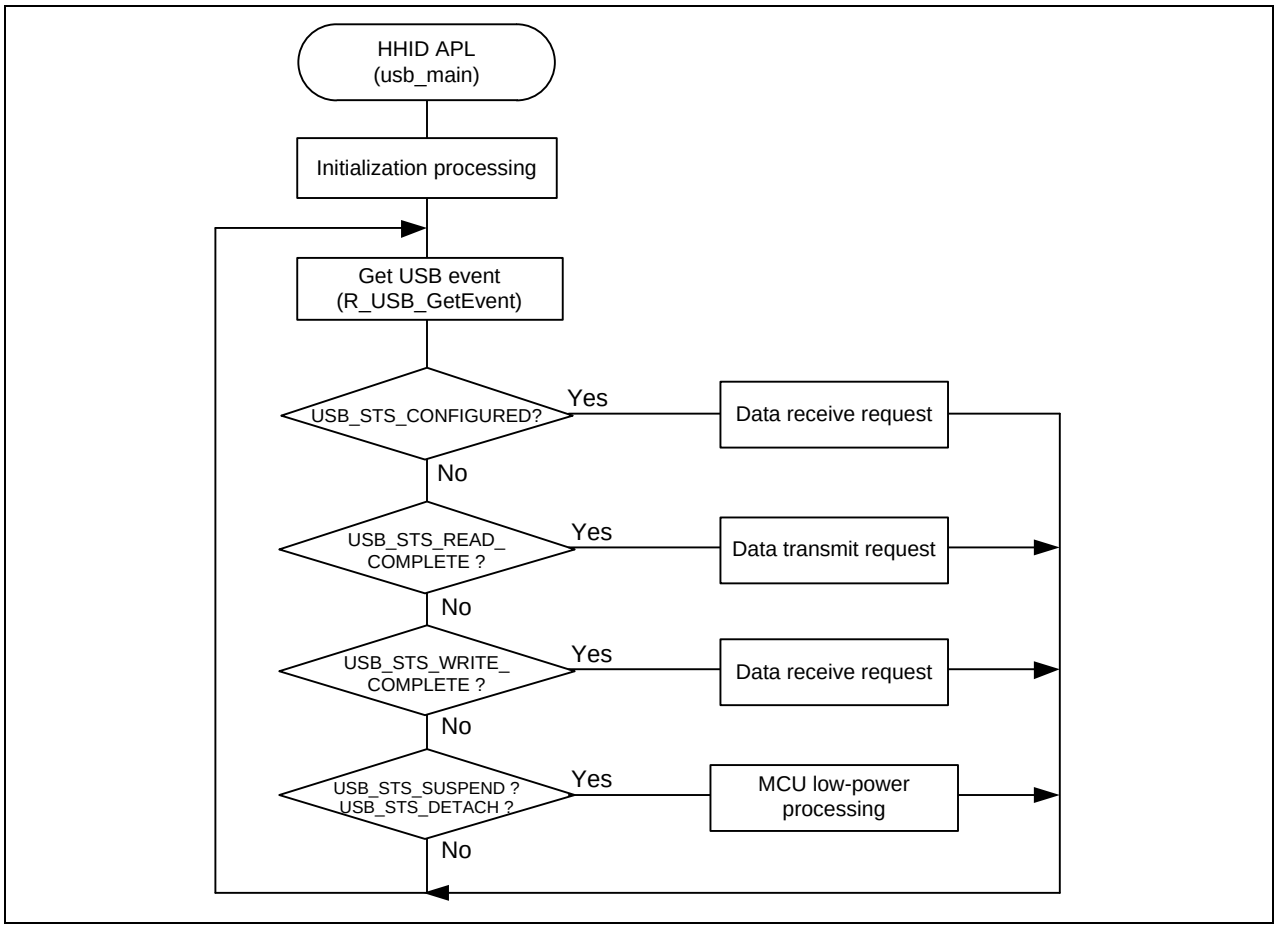


Figure 8-2 Main Loop (Normal mode)

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

8.3.1 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.

The switch specification is shown in Table 8-1.

Table 8-1 Switch Specification

Switch 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.

8.3.2 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).

8.4 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. 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 *USB Basic Host and Peripheral Driver Application Note* (Document number. R01AN0512EJ).

8.5 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

   r_usb_hhid_echo_apl_multi.c

9. Setup

9.1 Hardware

9.1.1 Example Operating Environment

Figure 9-1 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.

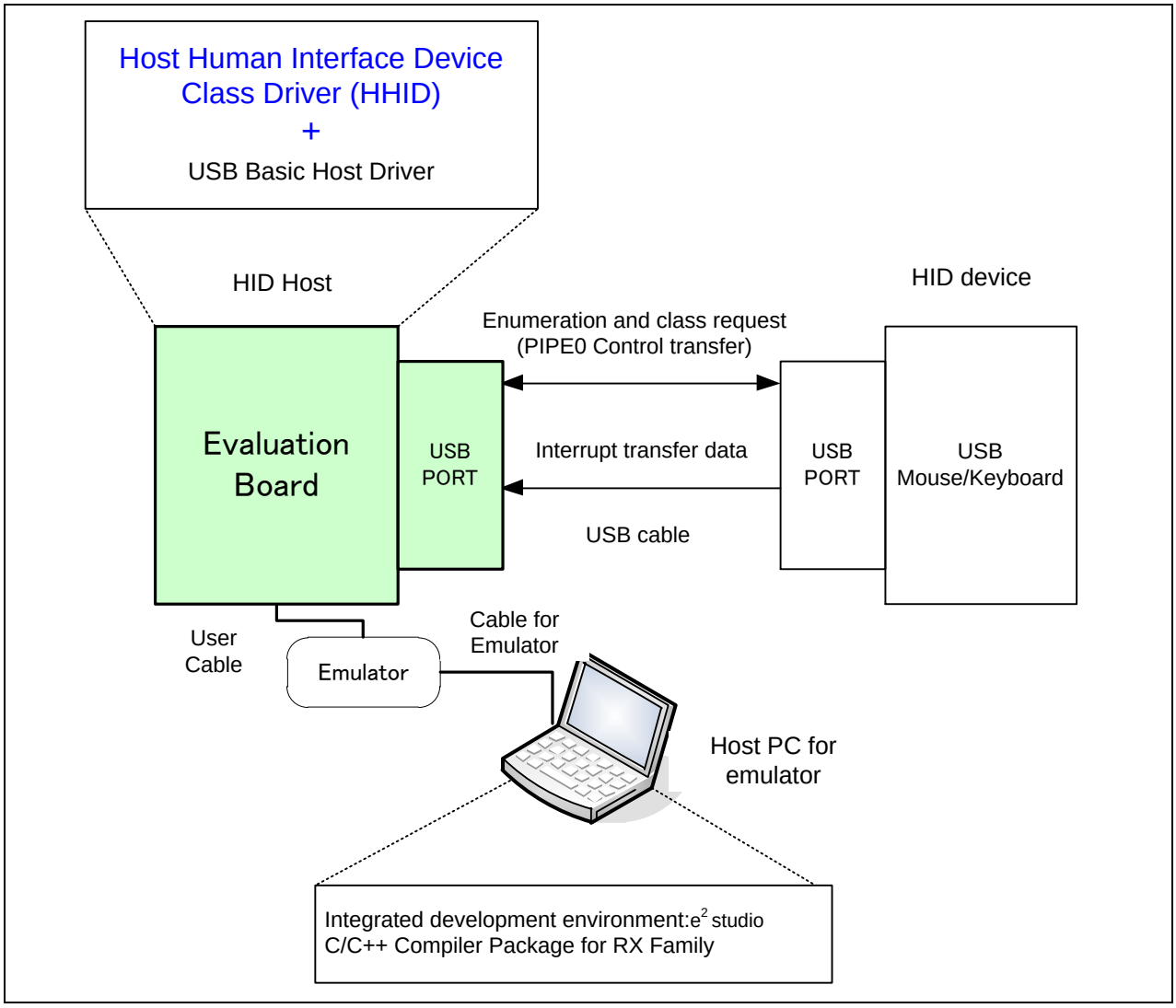


Figure 9-1 Example Operating Environment

9.1.2 RSK Setting

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

Table 9-1 RSK Setting

RSK	Jumper Setting
RSK+RX63N	J3: Shorted Pin 2-3 J4: Shorted Pin 2-3 J18:Shorted Pin1-2

Note:

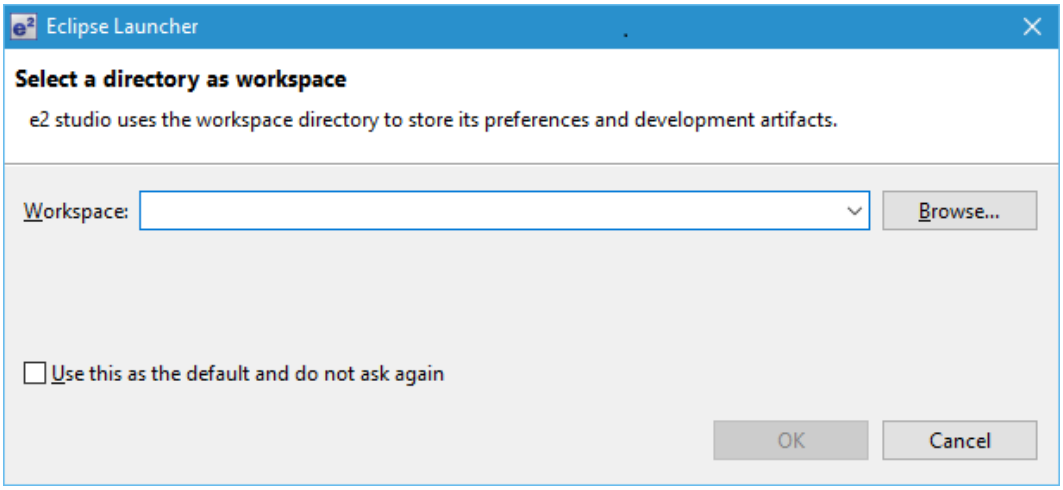
For the detail of RSK setting, refer to the user's manual of RSK.

9.2 Software

1) Setup e² studio

a) Start e² studio

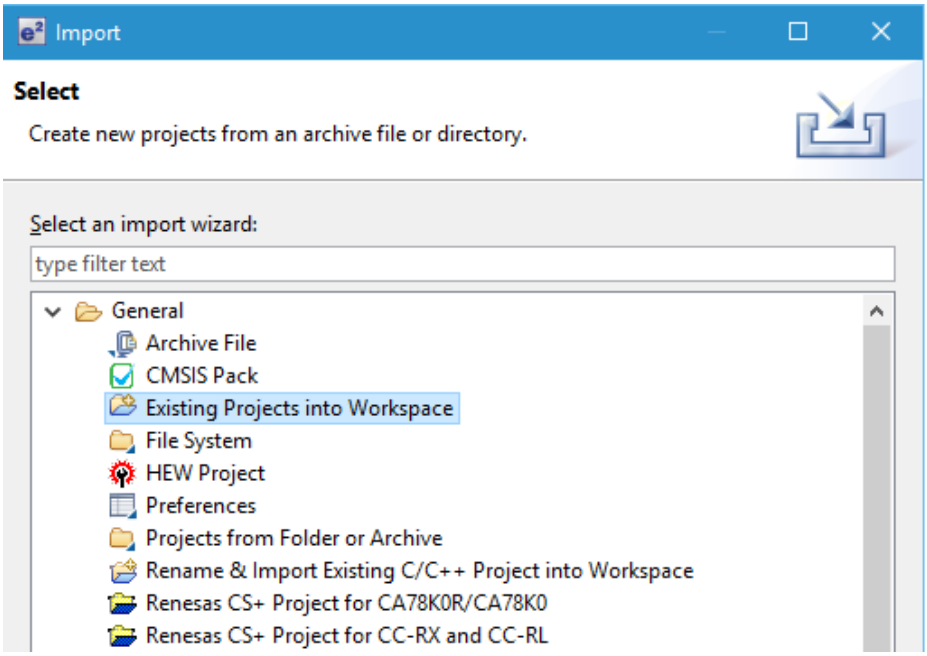
b) If you start up e² studio at first, the following dialog is displayed. Specify the folder to store the project in this dialog.



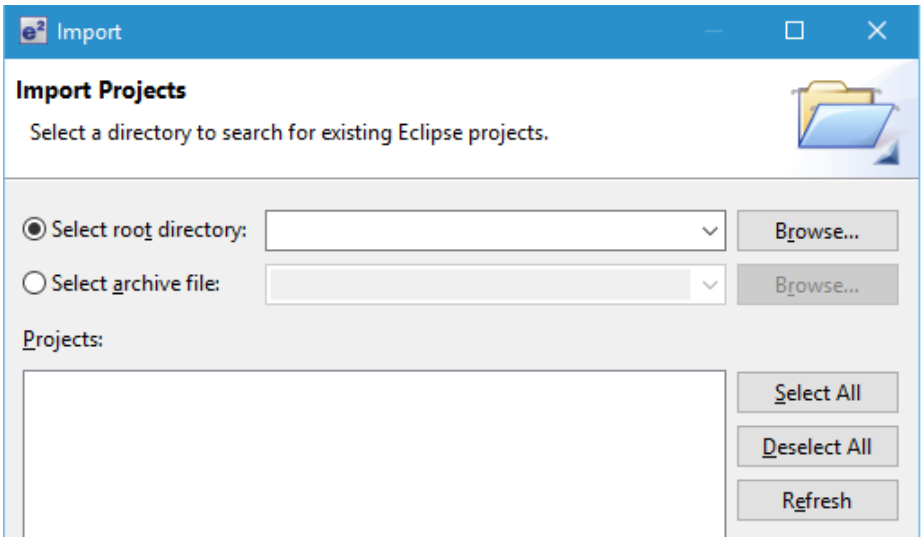
2) Import the project to the workspace

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 ".project" 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.

10. Creating an Application

Refer to the chapter "**Creating an Application Program**" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note*.

11. Using the e² studio project with CS+

The HHID contains a project only for e² studio. When you use the HHID with CS+, import the project to CS+ by following procedures.

[Note]

Uncheck the checkbox Backup the project composition files after conversion in Project Convert Settings window.

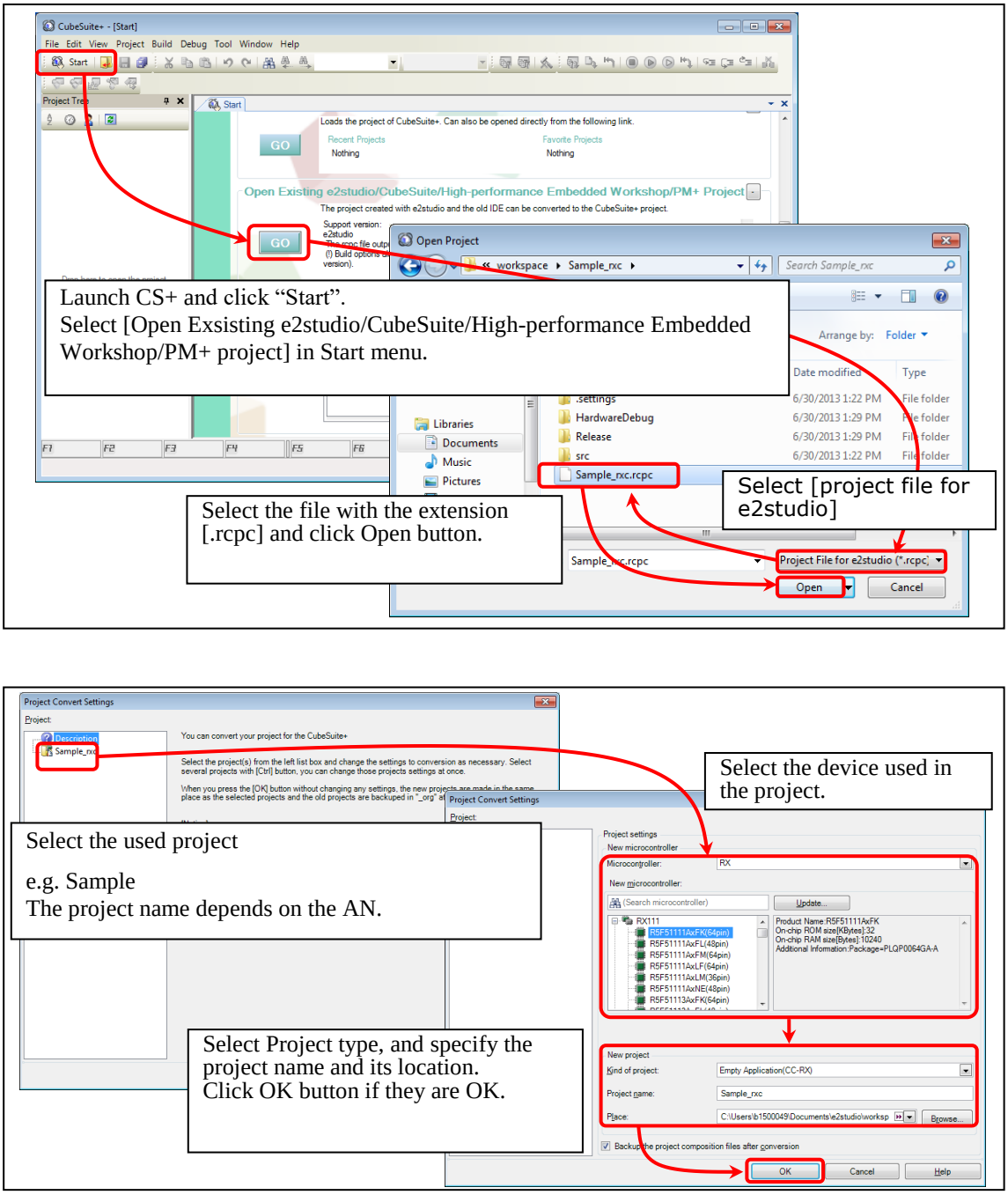


Figure 11-1 Using the e² studio project with CS+

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Mar.09.11	—	First edition issued
2.00	Mar.28.12	—	First edition issued for V.2.00
2.01	Feb.01.13	—	Description mistake and the reference error is fixed
2.10	Apr.01.13	—	First Release for V.2.10 Add Target Device RX63T.Add the information on RX63T
2.20	Sep.30.15	—	Change the application program. Change the folder structure. RX63N, RX631 and R8A66597 is deleted from Target Device. The multiple connecting of HID device is supported.
2.30	Sep 30, 2016	—	Supporting USB Host and Peripheral Interface Driver application note(Document No.R01AN3293EJ)
2.31	Sep 30, 2017	—	The contents of USB Host and Peripheral Interface Driver application note (Document number: R01AN3293EJ) is moved to this document and USB Host and Peripheral Interface Driver application note is deleted.
2.32	Mar 31, 2018	—	1. Adding *R_USB_HhidGetMxps* function. 2. The revision of USB Basic driver has been updated.
2.33	Jul 31, 2019	—	RX63N and RX631 are 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. 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 Microprocessing unit or Microcontroller unit products 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

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

(Rev.4.0-1 November 2017)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

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

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 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, 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 200333, China
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 No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 8.0