RX Family

R01AN2028EJ0144
Rev.1.44
Mar 01, 2025

USB Host Human Interface Device Class Driver (HHID) using Firmware Integration Technology

Introduction

This application note describes USB Host Human Interface Device Class Driver (HHID), which utilizes Firmware Integration Technology (FIT). This module performs hardware control of USB communication. It is referred to below as the USB-BASIC-FW FIT module.

Target Device

RX65N/RX651 Group
RX64M Group
RX71M Group
RX66T Group
RX72T Group
RX72M Group
RX66N Group
RX72N Group
RX671 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. RX64M Group User's Manual: Hardware (Document number .R01UH0377)
5. RX71M Group User's Manual: Hardware (Document number .R01UH0493)
6. RX65N/RX651 Group User's Manual: Hardware (Document number .R01UH0590)
7. RX65N/RX651-2M Group User's Manual: Hardware (Document number .R01UH0659)
8. RX66T User's Manual: Hardware (Document number. R01UH0749)
9. RX72T User's Manual: Hardware (Document number. R01UH0803)
10. RX72M User's Manual: Hardware (Document number. R01UH0804)
11. RX66N User's Manual: Hardware (Document number. R01UH0825)
12. RX72N User's Manual: Hardware (Document number. R01UH0824)
13. RX671 User's Manual: Hardware (Document number. R01UH0899)
14. USB Basic Host and Peripheral Driver using Firmware Integration Technology Application Note (Document number. R01AN2025)

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

Contents

1. Overview

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

This module 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: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology 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

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.

1.3 Limitations

The following limitations apply to the HHID.

1. The HID driver does not analyze the report descriptor. This driver determines the report format from the interface protocol.
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.

1.4 Terms and Abbreviations

Terms and abbreviations used in this document are listed below.

APL	:	Application program
HCD	:	Host Control Driver for USB-BASIC-FW
HDCD	:	Host Device Class Driver (Device Driver and USB Class Driver)
HHID	:	Host Human Interface Device
HID	:	Human Interface Device Class
HUBCD	:	Hub Class Driver
IDE	:	Integrated Development Environment
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

1.5 USB HHID FIT

User needs to integrate this module to the project using r_usb_basic. User can control USB H/W by using this module API after integrating to the project.

2. Module Configuration

The HHID comprises the HID class driver and device drivers for mouse and keyboard.

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.

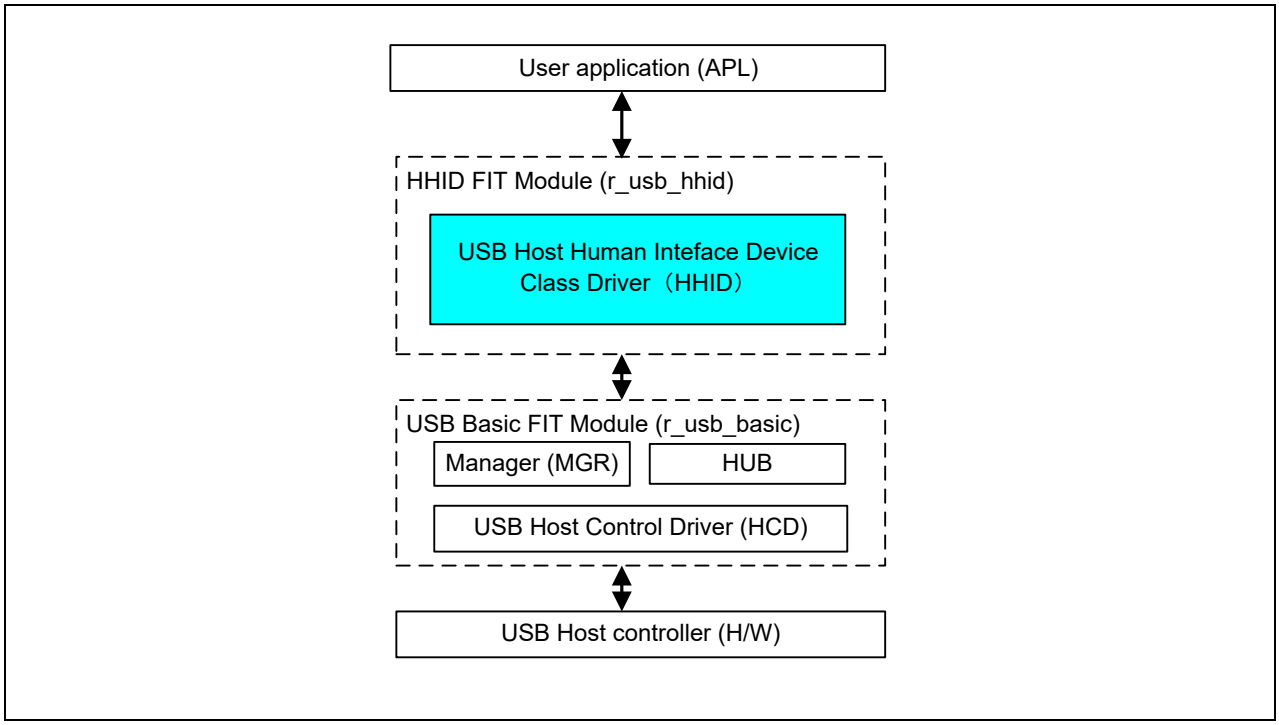

Figure 2-1 Software Module Structure

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/MGR	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 Software Requirements

This driver is dependent upon the following packages:

- r_bsp
- r_usb_basic

3.3 Operating Confirmation Environment

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

Table 3-1 Operating Confirmation Environment

Item	Contents
C compiler	Renesas Electronics C/C++ compiler for RX Family V.3.07.00 (The option "-lang=C99" is added to the default setting of IDE)
	GCC for Renesas RX 8.3.0.202411 (The option "-std=gnu99" is added to the default setting of IDE)
	IAR C/C++ Compiler for Renesas RX version 5.10.1
Real-Time OS	FreeRTOS V.10.0.0 RI600V4
Endian	Little Endian, Big Endian
USB Driver Revision Number	Rev.1.44
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

3.4 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
RX64M RX71M	USBI0 Interrupt (Vector number: 189, Interrupt source number : 62, Software Configurable Interrupt B) USB D0FIFO0 Interrupt (Vector number: 34) / USB D1FIFO0 Interrupt (Vector number: 35) USBR0 Interrupt (Vector number:90)
	USBAR Interrupt (Vector number: 94) USB D0FIFO2 Interrupt (Vector number: 32) / USB D1FIFO2 Interrupt (Vector number: 33)
RX65N RX651 RX72M RX72N RX66N	USBI0 Interrupt (Vector number: 185, Interrupt source number : 62, Software Configurable Interrupt B) USB D0FIFO0 Interrupt (Vector number: 34) / USB D1FIFO0 Interrupt (Vector number: 35) USBR0 Interrupt (Vector number:90)
RX66T RX72T	USBI0 Interrupt (Vector number: 174) / USBR0 Interrupt (Vector number: 90) USB D0FIFO0 Interrupt (Vector number: 34) / USB D1FIFO0 Interrupt (Vector number: 35)
RX671	USBI0 Interrupt (Vector number: 185, Interrupt source number : 62, Software Configurable Interrupt B) USB D0FIFO0 Interrupt (Vector number: 34) / USB D1FIFO0 Interrupt (Vector number: 35) USBR0 Interrupt (Vector number:90)
	USBI1 Interrupt (Vector number: 182, Interrupt source number : 63, Software Configurable Interrupt B) USB D0FIFO1 Interrupt (Vector number: 36) / USB D1FIFO1 Interrupt (Vector number: 37)

3.5 Header Files

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

3.6 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*.

3.7 Compile Setting

For compile settings, refer to chapter **7, Configuration (r_usb_hhid_config.h)** in this document and chapter "Configuration" in the document (Document number: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology Application Note.*.

3.8 ROM / RAM Size

The follows show ROM/RAM size of this driver.

1. CC-RX (Optimization Level: Default)

(1). Non-OS

	Checks arguments	Does not check arguments
ROM size	38.6K bytes (Note 3)	38.1K bytes (Note 4)
RAM size	14.4K bytes	14.4K bytes

(2). RTOS

a. FreeRTOS

	Checks arguments	Does not check arguments

ROM size	49.3K bytes (Note 3)	49.3K bytes (Note 4)
RAM size	35.9K bytes	35.9K bytes

b. RI600V4

	Checks arguments	Does not check arguments
ROM size	51.3K bytes (Note 3)	51.3K bytes (Note 4)
RAM size	18.0K bytes	18.0K bytes

2. GCC (Optimization Level: -O2)

	Checks arguments	Does not check arguments
ROM size	44.8K bytes (Note 3)	44.2K bytes (Note 4)
RAM size	14.2K bytes	14.2K bytes

3. IAR (Optimization Level: Medium)

	Checks arguments	Does not check arguments
ROM size	38.7K bytes (Note 3)	38.1K bytes (Note 4)
RAM size	12.9K bytes	12.9K bytes

[Note]

1. ROM/RAM size for BSP and USB Basic Driver is included in the above size.
2. The above is the size when specifying RX V2 core 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.
5. The result of RTOS includes the ROM/RAM size of the real-time OS.

3.9 Argument

For the structure used in the argument of API function, refer to chapter "**Structures**" in the document (Document number: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology Application Note*.

3.10 “for”, “while” and “do while” statements

In FIT module, when using “for”, “while” and “do while” statements (loop processing) in register reflection waiting processing, etc., write comments with “WAIT_LOOP” as a keyword for these loop processing. Also, write in the FIT documentation that “WAIT_LOOP” is written as a comment in these loop processes.

3.11 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using “Smart Configurator” on e^2 studio

By using the Smart Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e^2 studio

By using the FIT Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the Smart Configurator in CS+

By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(4) Adding the FIT module to your project on CS+

In CS+, please manually add the FIT module to your project. Refer to “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

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: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology 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: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology 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: R01AN2025) for USB Basic Host and Peripheral Driver using Firmware Integration Technology 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 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 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_hhid_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: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology Application Note*.

1. Setting pipe to be used

Set the pipe number (PIPE6 to PIPE9) to use for Interrupt IN transfer and Interrupt OUT. 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)
#define	USB_CFG_HHID_INT_OUT	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*, *USB_CFG_HHID_INT_IN3* and *USB_CFG_HHID_INT_OUT*, then set *USB_NULL* as these definitions.

8. Configuration File (When using RI600V4)

It is necessary to register the OS resource used by HHID USB driver to RI600V4 when using RI600V4. Please add the following definition in the configuration file. For how to create the configuration file, refer to the chapter, "**RI600V4(Configuration File Creation)**" in the document (Document number: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology Application Note*.

8.1 Mailbox Definition

name	:	ID_USB_RTOS_HHID_MBX
wait_queue	:	TA_FIFO
message_queue	:	TA_MFIFO

9. Creating an Application

Refer to the chapter "**Creating an Application Program**" in the document (Document number: R01AN2025) for *USB Basic Host and Peripheral Driver using Firmware Integration Technology Application Note*.

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	Apr 1, 2014	—	First edition issued
1.10	Dec 26, 2014	—	1. RX71M is added as new device. 2. The multiple connecting of HID device is supported. 3. The argument “ipno” is added to the following APIs. R_usb_hhid_GetReportLength, R_usb_hhid_get_hid_protocol
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(Document No.R01AN3293EJ)
1.21	Mar 31, 2017	—	1. When the return value of *R_USB_GetEvent* function is *USB_STS_READ_COMPLETE* or *USB_STS_WRITE_COMPLETE*, the USB driver has been changed so that *USB_HHID* is set for the member *type* of *usb_ctrl_t* structure. 2. The API other than the chapter ***API Functions*** is moved to the document (Document number: R01AN2025) of *USB Basic Host and Peripheral Driver Firmware Integration Technology*.
1.22	Sep 30, 2017	—	Supporting RX65N/RX651-2M
1.23	Mar 31, 2018	—	1. Supporting the Smart Configurator. 2. Adding *R_USB_HhidGetMxps* function.
1.24	Dec 28, 2018	—	Supporting RTOS.
1.25	Apr 16, 2019	—	Added RX66T/RX72T in Target Device.
1.26	May 31, 2019	—	1. Support GCC compiler and IAR compiler. 2. Remove RX63N from Target Device.
1.27	Jul 31, 2019	—	RX72M is added in 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.
1.42	Sep 29, 2023	—	Change argument type in *usb_hhid_task* function to *rtos_task_arg_t* type.
1.44	Mar 01, 2025	—	Change Disclaimer

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.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

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/.