RZ/A2M Group

USB Host Human Interface Device Class Driver (HHID)

Introduction

This application note describes USB Host Human Interface Device Class Driver (HHID). The USB HHID module, when used in combination with the USB-BASIC-FW module, operates as a USB host human interface device class driver (HHID).

Target Device

RZ/A2M Group

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

Contents

1. Overview

The USB HHID module, when used in combination with the USB-BASIC-FW 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 4 HID devices to 1 USB module by using USB Hub.

The names of API etc are different between port 0 and port 1.
In this document, API names etc. of port 0 are described as an example.

Table 1-1 Peripheral device used

Peripheral device	Usage
Host PC	output messages from the sample code

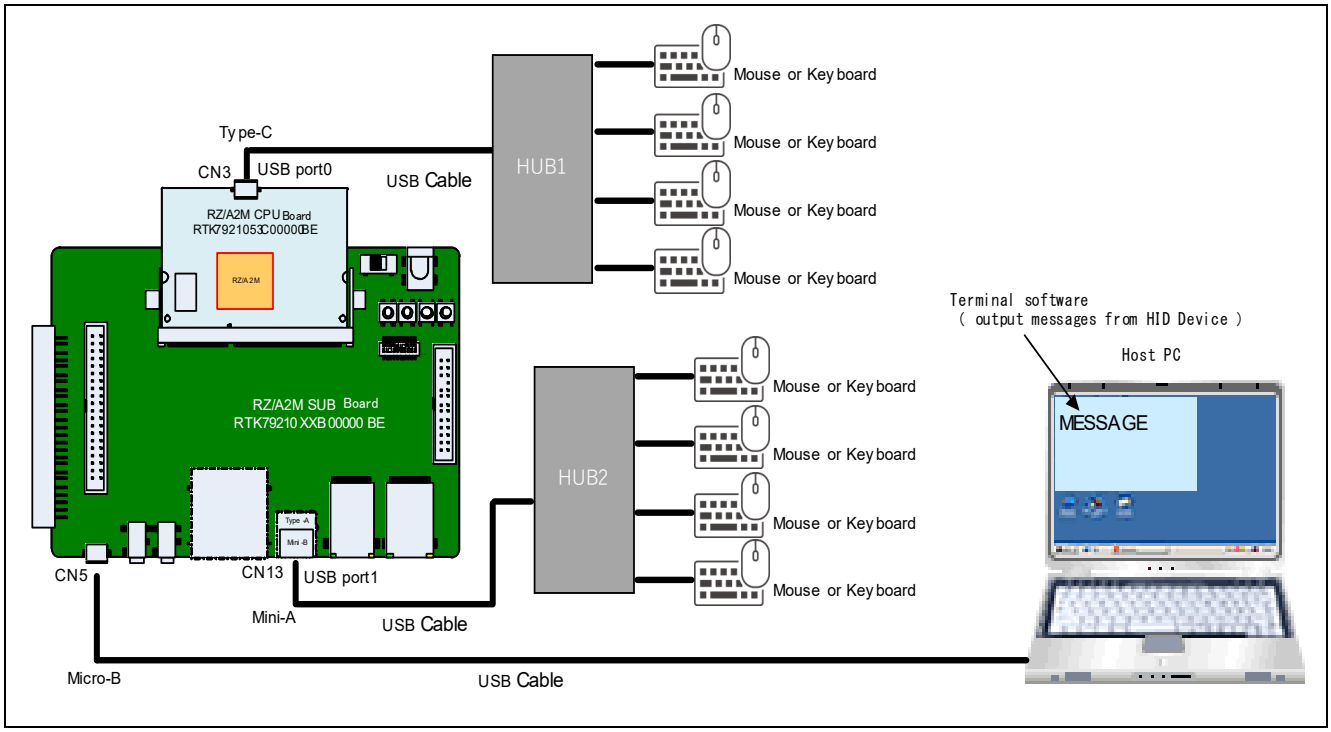


Figure 1-1 Operation check conditions

1.1 Limitations

The following limitations apply to the HHID.

1. This driver does not acquire and analyze the report descriptor, but determines the report format according to the interface protocol code. This HID driver determines the report format from the interface protocol (Keyboard/Mouse) alone.
2. One USB Hub is used for each USB module and a maximum of 4 HID devices can be connected.

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 Terms and Abbreviations

Abbreviation	Full Form
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
MGR	Peripheral Device State Manager for HCD
Non-OS	USB Driver for OS-less
RTOS	USB Driver for the real-time OS
USB	Universal Serial Bus
USB-BASIC-FW	USB Basic Host and Peripheral Driver

2. Operation Confirmation Conditions

Table 2-1 Operation Confirmation Conditions(1/2)

item	Contents
Microcomputer used	RZ/A2M
Operating frequency (Note)	CPU Clock (Iϕ) : 528MHz Image processing clock (Gϕ) : 264MHz Internal Bus Clock (Bϕ) : 132MHz Peripheral Clock 1 (P1ϕ) : 66MHz Peripheral Clock 0 (P0ϕ) : 33MHz QSPI0_SPCLK : 66MHz CKIO : 132MHz
Operating voltage	Power supply voltage (I/O): 3.3 V Power supply voltage (either 1.8V or 3.3V I/O (PVcc SPI)) : 3.3V Power supply voltage (internal): 1.2 V
Integrated development environment	e2 studio V7.8.0
C compiler	"GNU Arm Embedded Tool chain 6.3.1" compiler options(except directory path) Release: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0

Note: The operating frequency used in clock mode 1 (Clock input of 24MHz from EXTAL pin)

Table 2-2 Operation Confirmation Conditions(2/2)

Operation mode	Boot mode 3 (Serial Flash boot 3.3V)
Terminal software communication settings	• Communication speed: 115200bps • Data length: 8 bits • Parity: None • Stop bits: 1 bit • Flow control: None
Board to be used	RZ/A2M CPU board RTK7921053C00000BE RZ/A2M SUB board RTK79210XXB00000BE
Device (functionality to be used on the board)	• Serial flash memory allocated to SPI multi-I/O bus space (channel 0) Manufacturer : Macronix Inc. Model Name : MX25L51245GXD • RL78/G1C (Convert between USB communication and serial communication to communicate with the host PC.) • LED1

3. Reference Application Notes

・USB Basic Host Driver Application Note (Document number. R01AN4715EJ0140)

4. Software 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 4-1 shows the HHID software block diagram. Table 4-1 lists the modules and an overview of each.

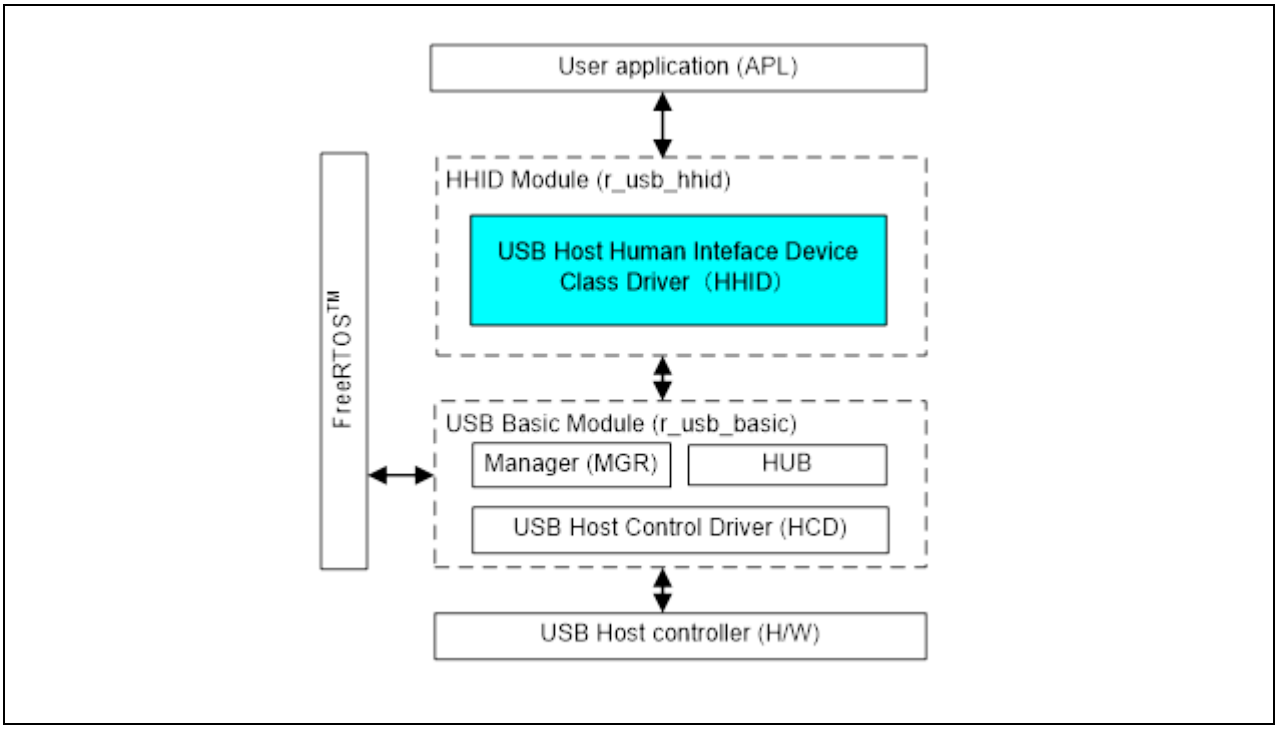


Figure 4-1 Software Block Diagram

Table 4-1 Module Function Descriptions

Module Name	Description
APL	User application program. The information received from the HID device. The PC Terminal S/W 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
FreeRTOS™	FreeRTOS™

5. API Information

This Driver API follows the Renesas API naming standards.

5.1 Hardware Requirements

This driver requires your MCU support the following features:

- USB

5.2 Software Requirements

This driver is dependent upon the following packages:

- r_usb_basic

5.3 Usage of Interrupt Vector

Table 5-1 shows the interrupt vector which this driver uses.

Table 5-1 List of Usage Interrupt Vectors

Port	Contents
0	USBIH0 Interrupt (Vector number: 63)
1	USBIH1 Interrupt (Vector number: 68)

5.4 Header Files

All API calls and their supporting interface definitions are located in r_usbh0_basic_if.h and r_usbh0_hhid_if.h.

5.5 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.

5.6 Compile Setting

For compile settings, refer to chapter 10, Configurationin this document and chapter "Configuration" in the document (Document number: R01AN4015EJ0140) for USB Basic Host Driver Application Note.

5.7 Argument

For the structure used in the argument of API function, refer to chapter " Structure Definitions" in the document (Document number: R01AN4015EJ0140) for USB Basic Host Driver Application Note.

6. Target Peripheral List (TPL)

For the structure used in the argument of API function, refer to chapter "Target Peripheral List (TPL)" in the document (Document number: R01AN4015EJ0140) for USB Basic Host Driver Application Note.

7. Human Interface Device Class (HID)

7.1 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

7.2 Class Requests (Host to Device Requests)

This driver supports the following class requests.

For the class request processing, refer to chapter "Control Transfer" in the document (Document number: R01AN4015EJ0140) for USB Basic Host Driver Application Note.

Table 7-1 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 a HID descriptor

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

7.2.1 GetReport Request Format

Table 7-2 shows the GetReport request format.
Receives a report from the device in a control transfer.

Table 7-2 GetReport Format

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

7.2.2 SetReport Request Format

Table 7-3 shows the SetReport request format.
Sends report data to the device in a control transfer.

Table 7-3 SetReport Format

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

7.2.3 GetIdle Request Format

Table 7-4 shows the GetIdle request format.
Acquires the interval time of the report notification (interrupt transfer). Idle rate is indicated in 4 ms units.

Table 7-4 GetIdle Format

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

7.2.4 SetIdle Request Format

Table 7-5 shows the SetIdle request format.
Sets the interval time of the report notification (interrupt transfer). Duration time is indicated in 4 ms units.

Table 7-5 SetIdle Format

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

7.2.5 GetProtocol Request Format

Table 7-6 shows the GetProtocol request format.
Acquires current protocol (boot protocol or report protocol) settings.

Table 7-6 GetProtocol Format

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

7.2.6 SetProtocol Request Format

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

Table 7-7 SetProtocol Format

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

7.3 HID Report Format

7.3.1 Receive Report Format

Table 7-8 shows the receive report format example used for notifications from the HID device. Reports are received in interrupt IN transfers or class request GetReport.

Table 7-8 Receive Report Format Example

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	-

7.3.2 Transmit Report Format

Table 7-9 shows the format example of the transmit report sent to the HID device. Reports are sent in the class request SetReport.

Table 7-9 Transmit Report Format Example

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	-	-

7.3.3 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.

8. API Functions

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

API	Port	Description
R_USBH0_HhidGetType() R_USBH1_HhidGetType()	0 1	Obtains type information for the HID device.
R_USBH0_HhidGetMxps() R_USBH1_HhidGetMxps()	0 1	Obtains the max packet size for the HID device.

Note:

Refer to chapter "API" in the document (Document number: R01AN4015EJ0140) for USB Basic Host Driver Application Note. when using the other API.

8.1 R_USBH0_HhidGetType

Obtains type information for the HID device.

Format

usbh0_err_t R_USBH0_HhidGetType(uint8_t address, uint8_t *p_type)

Arguments

address	Device address
p_drive	Pointer to the area to store the type information

Return Value

USBH0_SUCCESS	Successfully completed
USBH0_ERR_PARA	Parameter error
USBH0_ERR_NG	Other error

Description

Based on the information assigned to the 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 8-1.

Table 8-1 Type Information

Type Information	Description
USBH0_HID_KEYBOARD	Keyboard
USBH0_HID_MOUSE	Mouse
USBH0_HID_OTHER	HID device other than keyboard and mouse

Reentrant

This API is reentrant.

Note

1. Before calling this API, assign the device address of the HID device, to the address. If there is a problem with what is assigned to the address, then USBH0_ERR_PARA will be the return value.
2. If USB_NULL is assigned to the argument (p_type), then USBH0_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, USBH0_ERR_NG is returned.

Example

```
void usb_application( void )
{
  switch ( usbh0_hid_event_get(g_usbh0_hhid_devaddr) )
  {
       :
    case USBH0_STS_CONFIGURED:
       :
       R_USBH0_HhidGetType( g_usbh0_hhid_devaddr, &type );
       if( USBH0_HID_KEYBOARD == type )
       {
          :
       }
          :
    break;
       :
  }
}
```

8.2 R_USBH0_HhidGetMxps

Obtains the max packet size for the HID device.

Format

uint16_t R_USBH0_HhidGetMxps(uint16_t pipe_id)

Arguments

pipe_id PIPE number

Return Value

Max Packet Size

Description

Based on the information assigned to pipe_id, obtains max packet size for the connected HID device.

Reentrant

This API is reentrant.

Note

1. Please set the PIPE number to the pipe_id before calling this API.If you set an unused PIPE number, 0 will be returned.

Example

```
void usb_application( void )
{
  switch ( usbh0_hid_event_get(g_usbh0_hhid_devaddr) )
  {
       :
    case USBH0_STS_CONFIGURED:
       :
       pipe_id = R_USBH0_HstdGetPipeID(
                    g_usbh0_hhid_devaddr, USBH0_EP_INT, USBH0_EP_IN, 0xFF);
       R_USBH0_HhidGetMxps( pipe_id );
       if( USBH0_HID_KEYBOARD == type )
       {
           :
       }
           :
    break;
       :
  }
}
```

8.3 R_USBH0_HhidTask

Get string descriptor (String lang IDs / String iProduct) of connected HID device.

Format

void	R_USBH0_HhidTask(void)

Arguments

—	—

Return Value

—	—

Description

Get String lang IDs of connected HID device.

Get String iProduct of connected HID device.

Note

1. Please call in the loop that performs scheduler processing.

Example

```
void usbh_main(void)
{
    while(1)
    {
        // Confirming the event and getting (Note 1)
        R_USBH0_CstdScheduler();
        if( USBH0_FLGSET == R_USBH0_CstdCheckSchedule() )
        {
            R_USBH0_HstdMgrTask();          /* MGR task */
            R_USBH0_HhubTask();             /* HUB task */
            R_USBH0_HhidTask();             /* HHID Task */
        }
        usbh0_hid_main();
    }
} /* End of function usbh_main() */
```

9. Sample Application

9.1 Application Specifications

The following application programs are provided:

9.1.1 Normal Mode Application (r_usbh0_hhid_main_normal.c)

Transfers data to and from a HID device (mouse or keyboard) connected to the Board. Data received from the HID device is read and discarded.

[Note]

Up to 4 HID devices can be connected to a single USB module by using a USB hub.

9.1.2 Demo Mode Application (r_usbh0_hhid_main_demo.c)

Transfers data to and from a HID device (mouse or keyboard) connected to the Board. Data received from the HID device is displayed on PC Terminal S/W. I

[Note]

Up to 4 HID devices can be connected to a single USB module by using a USB hub.

9.2 Application Processing

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

9.2.1 Initial Settings

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

9.2.2 Setting of user-defined function (r_usbh0/1_pa_to_va)

In this driver, the following two functions must be implemented in the application.

```
uint32_t r_usbh0_pa_to_va(uint32_t paddr);
uint32_t r_usbh1_pa_to_va(uint32_t paddr);
```

In the sample application, it is defined in the following file.
rza2m_usbh_hid_sample_freertos_gcc\src\renesas\application\r_usb_mmu_pa_to_va.c

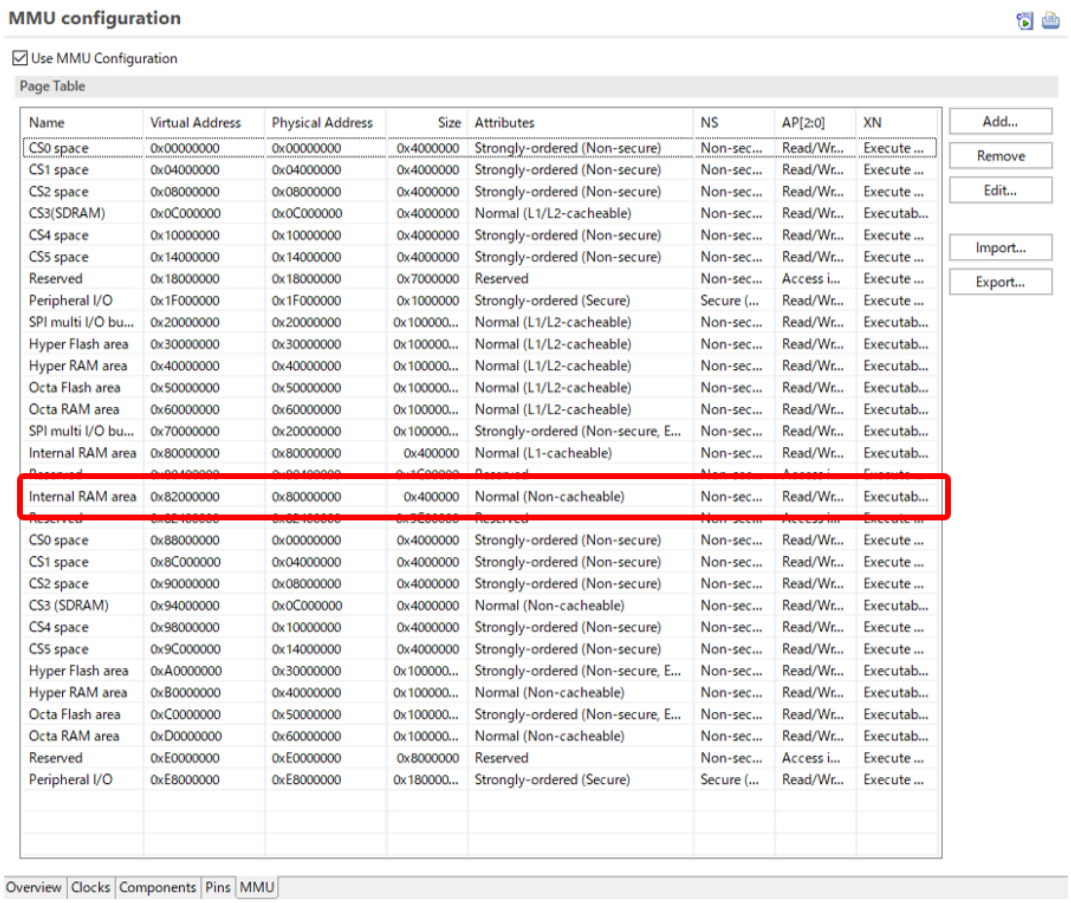
Set the values of the following four macros defined in r_usb_mmu_pa_to_va.c according to the usage environment. The setting values can be checked from the MMU tab of Smart Configurator.

Attribute: Allocate a total of 4MB area of Normal (Non-cacheable) Internal RAM Area to the following macros by 1MB.
* Macro settings can be used by default if the initial settings of the MMU have not been changed.

- Setting example (initial setting)

Set the virtual addresses corresponding to the physical addresses 0x80000000 to 0x80300000 in the following macros in 1MB increments. (See the figure below in the red frame)
Even if they are consecutive, it is necessary to define 4 macros each 1MB.

```
#define  USB_MMU_VIRTUAL_PAGE_0  (0x82000000)  //Virtual address corresponding to
                                                 0x80000000
#define  USB_MMU_VIRTUAL_PAGE_1  (0x82100000)  //Virtual address corresponding to
                                                 0x80100000
#define  USB_MMU_VIRTUAL_PAGE_2  (0x82200000)  //Virtual address corresponding to
                                                 0x80200000
#define  USB_MMU_VIRTUAL_PAGE_3  (0x82300000)  //Virtual address corresponding to
                                                 0x80300000
```



9.2.3 Main Loop (r_usb_hhid_apl.c)

After the USB driver initial settings, call the scheduler (R_USBH0_CstdScheduler()) from the main routine of the application. Calling R_USBH0_CstdScheduler() from the main routine causes a check for events. If there is an event, a flag is set to inform the scheduler that an event has occurred. After calling R_USBH0_CstdScheduler(), call R_USBH0_CstdCheckSchedule() to check for events. Also, it is necessary to run processing at regular intervals to get events and perform the appropriate processing.(Note 1)

In order to use the 3 Non OS functions called in the Non OS main loop as the OS task, call R_USBH0_Init () with BSP_CFG_RTOS_USED = 1.
Therefore, when BSP_CFG_RTOS_USED = 1, it is not necessary to call four Non OS functions in the main loop.
The scheduler:R_USBH0_CstdScheduler() sends messages to each task function regardless of Non OS or OS and controls task processing.

```
void usbh_main(void)
{
    while(1)
    {
        // Confirming the event and getting (Note 1)
        R_USBH0_CstdScheduler();
#if(BSP_CFG_RTOS_USED == 0)
        if( USBH0_FLGSET == R_USBH0_CstdCheckSchedule() )
        {
            R_USBH0_HstdMgrTask();          /* MGR task */
            R_USBH0_HhubTask();             /* HUB task  (Note 3) */    } (Note 2)
            R_USBH0_HhidTask();             /* HHID Task */
        }
#endif /* (BSP_CFG_RTOS_USED == 0) */
        usbh0_hid_main();
    }
} /* End of function usbh_main() */
```

[Note]

1. If, after getting an event with R_USBH0_CstdScheduler () and before running the corresponding processing, R_USBH0_CstdScheduler () is called again and gets another event, the first event is discarded. After getting an event, always call the corresponding task to perform the appropriate processing.
2. Be sure to describe these processes in the main loop for the application program.
3. It is only necessary to call this function when the HUB will be used.

9.2.4 APL

APL is managed by the state transition.

Table 9-1 shows list of states.

Table 9-1 List of States

State	Description
USBH0_STS_CONFIGURED	Configured
USBH0_STS_REQUEST_COMPLETE	Class Request Complete
USBH0_STS_READ_COMPLETE	Read Complete

Figure 9-1 shows the process flowchart of APL.

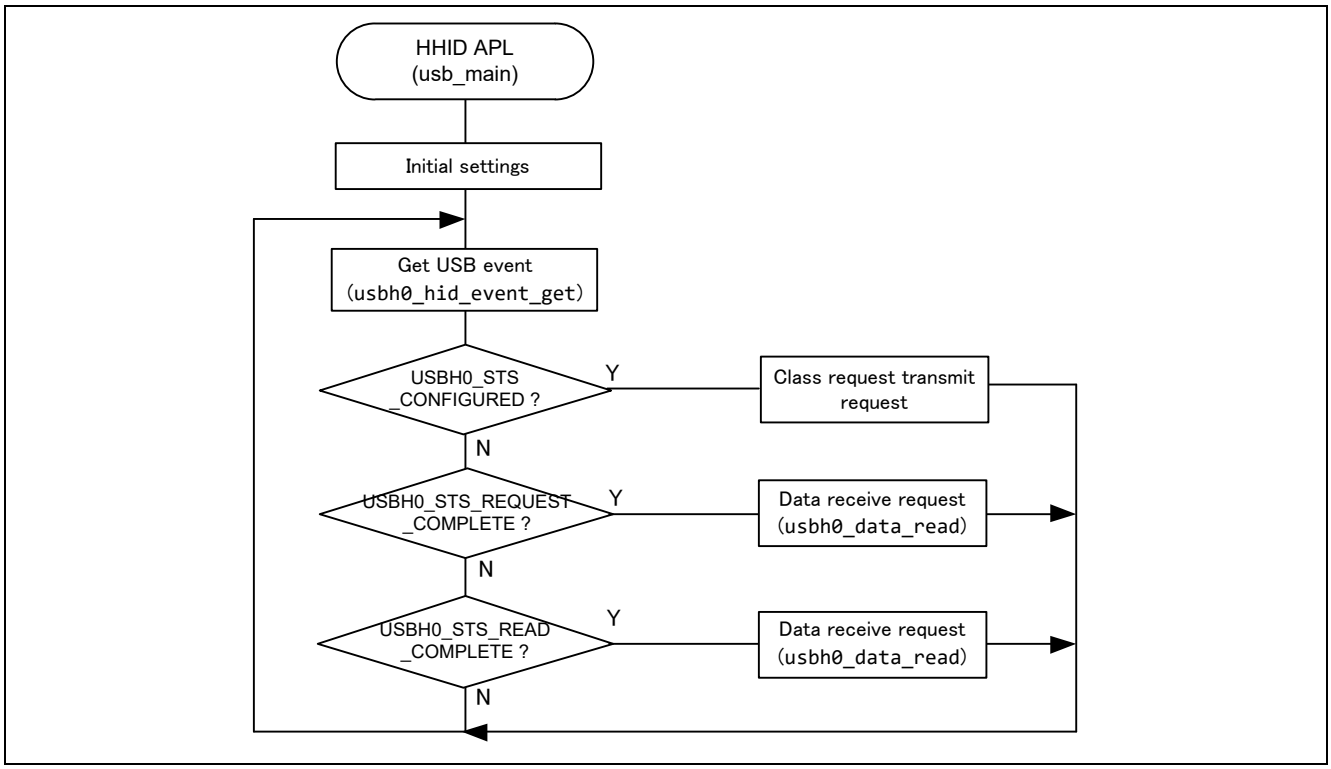


Figure 9-1 Main Loop flowchart

9.2.5 State Management

An overview of the processing associated with each state is provided below.

1) Device Configured (USBH0_STS_CONFIGURED)

== Outline ==

This state indicates that the HID device is connected and has become device configured.

In this state, a Set Protocol request is sent to the device.

== Description ==

1. A Set Protocol request is sent to the device.
2. When the Set Protocol request is complete, change the state to USBH0_STS_REQUEST_COMPLETE.

2) Class Request Complete (USBH0_STS_REQUEST_COMPLETE)

== Outline ==

This state indicates that the class request for the HID device is complete.

Make a data read request.

== Description ==

1. Make a data read request.
2. After data read is complete, change the state to USBH0_STS_READ_COMPLETE.

3) Read Complete (USBH0_STS_READ_COMPLETE)

== Outline ==

This state indicates that data has been received to the HID device.

Next, make a data read request to the HID device.

== Description ==

1. Make a data read request.
2. After data read is complete, change the state to USBH0_STS_READ_COMPLETE.

10. Configuration

Make the following settings.

Make settings for the r_usbh0_basic_config.h and r_usbh1_basic_config.h files.
For r_usbh0_basic_config.h and r_usbh1_basic_config.h, refer to the “Configuration” chapter in the USB Basic Host Driver application note (Document No. R01AN4715EJ0120).

10.1 r_usb_hhid_apl.h

Make settings for the following definitions.

1. USBHx_APL_MODE definition

Specify one of the following for the USBHx_APL_MODE definition:

```
#Define USBHx_APL_MODE    HID_NORMAL          // NORMAL mode
#Define USBHx_APL_MODE    HID_DEMO            // DEMO mode
```

11. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware
The latest version can be downloaded from the Renesas Electronics website.

RTK7921053C00000BE (RZ/A2M CPU board) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

RTK79210XXB00000BE (RZ/A2M SUB board) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
The latest version can be downloaded from the ARM website.

ARM Cortex™-A9 Technical Reference Manual Revision: r4p1
The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification - Architecture version 2.0
The latest version can be downloaded from the ARM website.

ARM CoreLink™ Level 2 Cache Controller L2C-310 Technical Reference Manual Revision: r3p3
The latest version can be downloaded from the ARM website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

Integrated development environment e2studio User’s Manual can be downloaded from the Renesas Electronics website.
The latest version can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.15.19	26	First edition issued
1.10	May.17.19	26	Table 2.1 Operation Confirmation Conditions(1/2) Remove compiler option "-mthumb-interwork"
1.20	Mar.31.20	19	Add user defined function
1.30	June.30.20		Basic verup

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 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.

(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.4.0-1 November 2017)

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/.