RX Family

R01AN2169EJ0130
Rev.1.30
Jul 31, 2024

USB Host Mass Storage Class Driver for USB Mini Firmware Using Firmware Integration Technology

Introduction

This application note describes USB Host Mass Storage Class Driver(HMSC), which utilizes Firmware Integration Technology (FIT). This module operates in combination with the USB Basic Mini Host and Peripheral Driver. It is referred to below as the USB HMSC FIT module.

Target Device

RX111 Group
RX113 Group
RX231 Group
RX23W Group
RX261 Group

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

Related Documents

1. Universal Serial Bus Revision 2.0 specification
2. USB Mass Storage Class Specification Overview Revision 1.1
3. USB Mass Storage Class Bulk-Only Transport Revision 1.0
   http://www.usb.org/developers/docs/
4. RX111 Group User’s Manual: Hardware (Document number .R01UH0365)
5. RX113 Group User’s Manual: Hardware (Document number.R01UH0448)
6. RX231 Group User’s Manual: Hardware (Document number .R01UH0496)
7. RX23W Group User’s Manual: Hardware (Document number .R01UH0823)
8. RX261 Group User’s Manual: Hardware (Document number .R01UH1045)
9. RX Family M3S-TFAT-Tiny: FAT file system software (Document number: R20AN0038EJ)
10. RX Family M3S-TFAT-Tiny: Memory Driver Interface Module (Document number: R20AN0335EJ)
11. USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note (Document number.R01AN2166)

Renesas Electronics Website

http://www.renesas.com/

USB Device Page

http://www.renesas.com/prod/usb/

Contents

1. Overview

The USB HMSC FIT module, when used in combination with the USB-BASIC-F/W FIT module, operates as a USB host mass storage class driver (HMSC).
The HMSC comprises a USB mass storage class bulk-only transport (BOT) protocol. When combined with a file system and storage device driver, it enables communication with a BOT-compatible USB storage device.
Note that please use the M3S-TFAT-Tiny (Document number: R20AN0038) and Memory driver interface module (Document numver: R20AN0335) in combination when using this driver.

This module supports the following functions.

1. Checking of connected USB storage devices (to determine whether or not operation is supported).
2. Storage command communication using the BOT protocol.
3. Support for SFF-8070i (ATAPI) USB mass storage subclass.

1.1 Please be sure to read

Please refer to the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) 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 Limitation

1. Some MSC devices may be unable to be connected (because they are not recognized as storage devices).
2. MSC devices that return values of 1 or higher in response to the GetMaxLun command (mass storage class command) are not supported.
3. USB storage devices with a sector size of 512 bytes can be connected.
4. A device that does not respond to the READ_CAPACITY command operates as a device with a sector size of 512 bytes.

1.4 Terms and Abbreviations

APL	:	Application program
BOT	:	Mass storage class Bulk Only Transport
FSL	:	FAT File System Library
HCD	:	Host Control Driver of
HDCD	:	Host Device Class Driver (device driver and USB class driver)
MGR	:	Peripheral device state manager of HCD
MSC	:	Mass Storage Class
RSK	:	Renesas Starter Kits
RTOS	:	USB Driver for the real-time OS
TFAT	:	Tiny FAT file system software for microcontrollers (M3S-TFAT-Tiny-RX)
USB-BASIC-FW	:	USB Basic Mini Host and Peripheral Driver
USB	:	Universal Serial Bus

1.5 USB HMSC FIT Module

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

2. Software Configuration

HDCD (Host Device Class Driver) is the all-inclusive term for HMSDD (Host Mass Storage Device Driver) and HMSCD (USB Host Mass Storage Class Driver).

Figure 2-1 shows the HMSC software block diagram, with HDCD as the centerpiece. Table 2-1 describes each module.

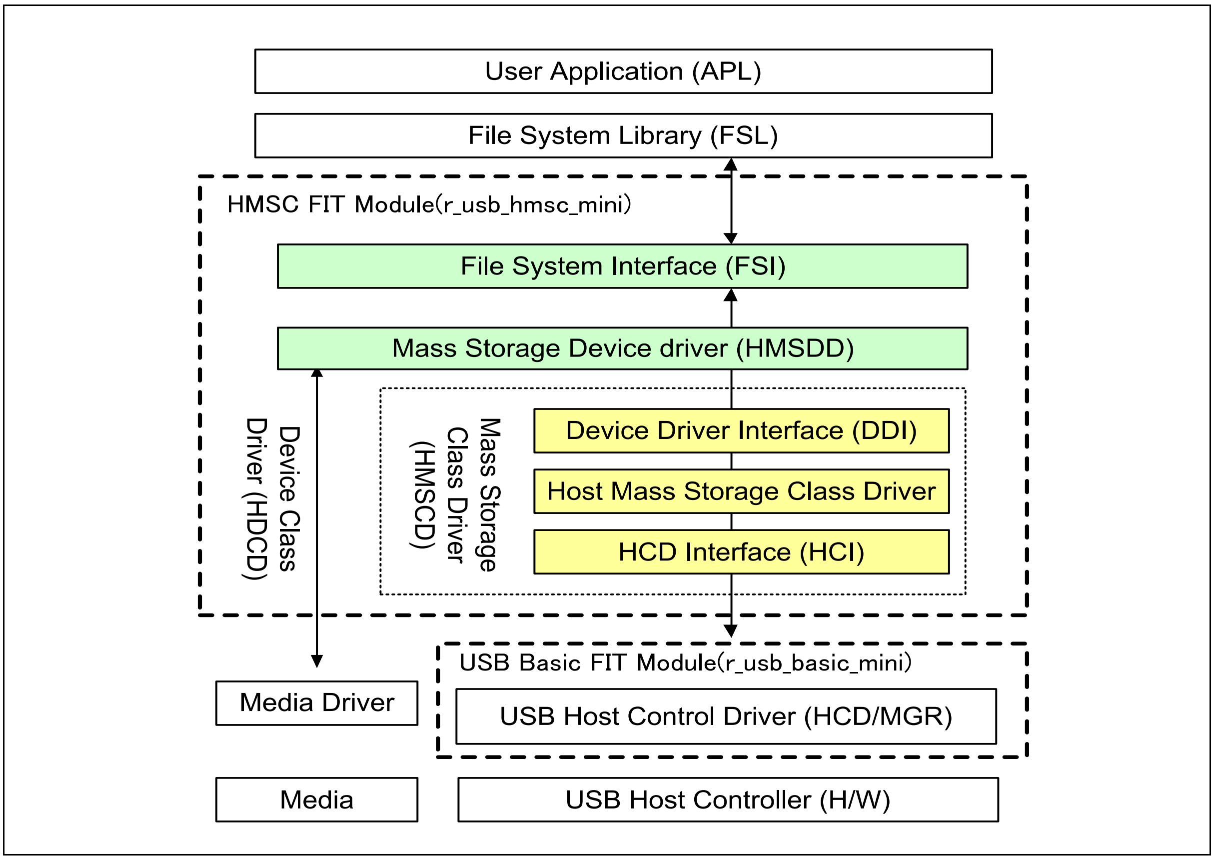

Figure 2-1 Software Module Structure

Table 2-1 Module

Module	Description
FSI	FSL-HMSDD interface functions. They should be modified to match FSL.
HMSDD	To be created (modified) by the customer to match the storage media.
DDI	HMSDD-HMSCD interface functions. They should be modified to match the storage media interface of HMSDD.
HMSCD	The USB host mass storage class driver. It appends BOT protocol information to storage commands and sends requests to HCD. It also manages the BOT sequence. The storage commands should be added (modified) by the customer to match the system specifications. SFF-8070i (ATAPI) is supported in the example code.
HCI	HMSCD-HCD interface functions.
MGR	Enumerates the connected devices and starts HMSCD. Also performs device state management.
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 Software Requirements

This driver is dependent upon the following packages:

- r_bsp
- r_usb_basic_mini

3.3 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.3.06.00 (The option "-lang=C99" is added to the default setting of IDE)
	GCC for Renesas RX 8.3.0.202311 (The option "-std=gnu99" is added to the default setting of IDE)
	IAR C/C++ Compiler for Renesas RX version 4.20.3
Real-Time OS	FreeRTOS V.10.4.3 RI600V4 V.1.06
Endian	Little Endian, Big Endian
USB Driver Revision Number	Rev.1.30
Using Board	Renesas Starter Kit for RX111 Renesas Starter Kit for RX113 Renesas Starter Kit for RX231 Renesas Solution Starter Kit for RX23W Evaluation Kit for RX261

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
RX111 RX113 RX231 RX23W RX261	USBI0 Interrupt (Vector number: 36) / USBR0 Interrupt (Vector number: 90) USB D0FIFO0 Interrupt (Vector number: 36) / USB D1FIFO0 Interrupt (Vector number: 37)

3.5 Header Files

All API calls and their supporting interface definitions are located in *r_usb_basic_mini_if.h* and *r_usb_hmsc_mini_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 **8, Configuration (r_usb_hmsc_mini_config.h)** in this document and chapter "Configuration" in the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) 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	21.5K bytes (Note 4)	21.2K bytes (Note 5)
RAM size	5.0K bytes	5.0K bytes

(2). RI600V4

	Checks arguments	Does not check arguments
ROM size	38.2K bytes (Note 4)	37.9K bytes (Note 5)
RAM size	5.4K bytes	5.4K bytes

(3). FreeRTOS

	Checks arguments	Does not check arguments
ROM size	34.2K bytes (Note 4)	33.9K bytes (Note 5)
RAM size	17.2K bytes	17.2K bytes

2. GCC (Optimization Level: -O2)

	Checks arguments	Does not check arguments
ROM size	25.5K bytes (Note 4)	25.2K bytes (Note 5)
RAM size	4.2K bytes	4.2K bytes

3. IAR (Optimization Level: Medium)

	Checks arguments	Does not check arguments
ROM size	17.7K bytes (Note 4)	17.5K bytes (Note 5)
RAM size	3.3K bytes	3.3K bytes

[Note]

1. ROM/RAM size for BSP and USB Basic Driver is included in the above size.
2. ROM/RAM size for FAT is not included in the above size.
3. The above is the size when specifying RX V2 core option.
4. 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_mini_config.h* file.
5. 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_mini_config.h* file.

3.9 Argument

For the structure used in the argument of API function, refer to chapter "**Structures**" in the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) 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: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note.*

5. Class Driver

5.1 Class Requet

This driver supports the following class request.

Table 5-1 Class Request

Request	Description
GetMaxLun	Gets the maximum number of units that are supported.
MassStrageReset	Cancels a protocol error.

5.2 Storage Command

This driver supports the following storage command.

1. TEST_UNIT_READY
2. REQUEST_SENSE
3. MODE_SELECT10
4. MODE_SENSE10
5. PREVENT_ALLOW
6. READ_FORMAT_CAPACITY
7. READ10
8. WRITE10

6. API Functions

The following are Host Mass Storage Class specific API functions

API	Desription
R_USB_HmscStrgCmd()	Issues a Mass Storage command.
R_USB_HmscGetSem()	Gets a semaphore (Only RTOS)
R_USB_HmscRelSem()	Releases a semaphore (Only RTOS)

Note:

1. Uses the FAT (File Allocation Table) API to access storage media.
2. Refer to chapter "API" in the document(Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note* when using other API.

6.1 R_USB_HmscStrgCmd

Issues a Mass Storage command

Format

usb_err_t R_USB_HmscStrgCmd(uint8_t *p_buf, uint16_t command)

Arguments

p_buf	Pointer to data area
command	Mass storage command

Return Value

USB_SUCCESS	Successfully completed
USB_ERR_PARA	Parameter error
USB_ERR_NG	Other error

Description

The Mass Storage command assigned to the argument *(command)* is issued to the MSC device. An application program can check the completion of the Mass Storage command with the *USB_STS_MSC_CMD_COMPLETE* return value of the *R_USB_GetEvent* function.

If a Mass Storage command with response data is issued, after checking *USB_STS_MSC_CMD_COMPLETE* return value of the *R_USB_GetEvent* function, an application program can obtain the response data from the area indicated by the second argument (*p_buf*). Check the member (*size*) of the usb_crtl_t structure to get the size of the response data that was received.

Assign the following to the argument (*command*).

Table 6-1 Mass Storage Command

MassStorage Command
USB_ATAPI_TEST_UNIT_READY
USB_ATAPI_REQUEST_SENSE
USB_ATAPI_INQUIRY
USB_ATAPI_MODE_SELECT10
USB_ATAPI_PREVENT_ALLOW
USB_ATAPI_READ_FORMAT_CAPACITY
USB_ATAPI_READ_CAPACITY
USB_ATAPI_MODE_SENSE10

Note

1. Do not assign a pointer to the auto variable (stack) area to the arguments (*p_buf*).
2. Assign *USB_NULL* to the argument (*p_buf*) when issuing the mass storage command without the response data.
3. If a command other than the Mass Storage commands listed in Table 6-1 is assigned to the argument (*command*), then *USB_ERR_PARA* will be the return value.
4. When calling FAT API and this API after issuing the Mass storage command by this API, be sure to call these APIs after checking the return value (*USB_STS_CMD_COMPLETE*) of *R_USB_GetEvent* function.
5. Refer to chapter "**7. Return Value (USB_STS_MSC_CMD_COMPLETED) of R_USB_GetEvent Function**" about CSW.
6. The CSW information is set to the member (*status*) of the *usb_ctrl_t* structure. If the value of the member (*status*) is *USB_CSW_FAIL*, issue the "*Requeset Sense*" command to the MSC device using this API.
7. Set the page code (1 Byte) of the "*Mode Sense10*" command in the start address to the area indicated by the 2nd argument (*p_buf*).

8. Set the parameter data for the "*Mode Select10*" command to the area indicated by the 2nd argument (*p_buf*) based on the specification for USB Mass Storage Subclass (SFF-8070i etc).

9. 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

1. Non-OS

```
void    usb_application( void )
{
    usb_ctrl_t ctrl;
    usb_err_t err;
                        :
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
        {
                        :
            case USB_STS_CONFIGURED:
                        :
                g_buf[0] = 0x3F;    /* Page Code */
                R_USB_HmscStrgCmd( &g_buf, USB_ATAPI_MODE_SENSE10 );
                        :
            break;
            case USB_STS_MSC_CMD_COMPLETE:
                if( ctrl.status == USB_CSW_FAIL )
                {
                    R_USB_HmscStrgCmd(&ctrl, &g_buf, USB_ATAPI_REQUEST_SENSE);
                }
                        :

            break;
                        :
        }
    }
}
```

2. RTOS

```
/* Callback function */
void usb_apl_callback (usb_ctrl_t *p_ctr, rtos_task_id_t task_id, uint8_t is_request)
{
    USB_APL_SND_MSG(USB_APL_MBX, (usb_msg_t *)p_ctrl);
}

void        usb_application_task( void )
{
    usb_ctrl_t      ctrl;
    usb_ctrl_t      *p_mess;
            :
    while(1)
    {
        USB_APL_RCV_MSG(USB_APL_MBX, (usb_msg_t **)&p_mess);
        ctrl = *p_mess;
        switch (ctrl.event)
        {
                    :
            case USB_STS_CONFIGURED:
                    :
                g_buf[0] = 0x3F             /* Page Code */
                R_USB_HmscStrgCmd(&g_buf, USB_ATAPI_MODE_SENSE10);
                    :
            break;
            case USB_STS_MSC_CMD_COMPLETE:
                if (ctrl.status == USB_CSW_FAIL)
                {
                    R_USB_HmscStrgCmd(&g_buf, USB_ATAPI_REQUEST_SENSE);
                }
                    :
            break;
        }
    }
}
```

6.2 R_USB_HmscGetSem

Gets a semaphore (Only RTOS)

Format

void R_USB_HmscGetSem(void)

Arguments

none

Return Value

none

Description

Gets a specific semaphore which is used in HMSC driver.

Note

1. Be sure to call this API before calling the FAT file open function (e.g *R_tfat_f_open*).
2. If this API is called when a semapohre counter value is zero, the user task which calls this API shift to a semaphore waiting status.
3. The creation processing of a semaphore which this API uses is performed in USB driver.

Example

```
/* Callback function */
void usb_apl_callback (usb_ctrl_t *p_ctr, rtos_task_id_t task_id, uint8_t is_request)
{
    USB_APL_SND_MSG(USB_APL_MBX, (usb_msg_t *)p_ctrl);
}
void        usb_application_task( void )
{
    usb_ctrl_t      ctrl;
    usb_ctrl_t      *p_mess;
            :
    while(1)
    {
        USB_APL_RCV_MSG(USB_APL_MBX, (usb_msg_t **)&p_mess);
        ctrl = *p_mess;
        switch (ctrl.event)
        {
                        :
            case USB_STS_CONFIGURED:
                        :
                R_USB_HmscGetSem();
                R_tfat_f_open(&file, (const char *) &g_msc_file[drvno][0],
                                (TFAT_FA_CREATE_ALWAYS | TFAT_FA_WRITE));
                R_tfat_f_write(&file, g_file_data, sizeof(g_file_data), &file_size);
                R_tfat_f_close(&file);
                R_USB_HmscRelSem();
                        :
            break;
                        :
        }
    }
}
```

6.3 R_USB_HmscRelSem

Releases a semaphore (Only RTOS)

Format

void R_USB_HmscRelSem(void)

Arguments

none

Return Value

none

Description

Releases a specific semaphore which is used in HMSC driver.

Note

1. Be sure to call this API after calling the FAT file close function (e.g *R_tfat_f_close*).
2. An application task during a semaphore waiting status by *R_USB_HmscGetSem* function is released the semaphore waiting status by this API.
3. The creation processing of a semaphore which this API uses is performed in USB driver.

Example

```
/* Callback function */
void usb_apl_callback (usb_ctrl_t *p_ctr, rtos_task_id_t task_id, uint8_t is_request)
{
    USB_APL_SND_MSG(USB_APL_MBX, (usb_msg_t *)p_ctrl);
}
void        usb_application_task( void )
{
    usb_ctrl_t      ctrl;
    usb_ctrl_t      *p_mess;
            :
    while(1)
    {
        USB_APL_RCV_MSG(USB_APL_MBX, (usb_msg_t **)&p_mess);
        ctrl = *p_mess;
        switch (ctrl.event)
        {
                    :
            case USB_STS_CONFIGURED:
                    :
                R_USB_HmscGetSem();
                R_tfat_f_open(&file, (const char *) &g_msc_file[drvno][0],
                            (TFAT_FA_CREATE_ALWAYS | TFAT_FA_WRITE));
                R_tfat_f_write(&file, g_file_data, sizeof(g_file_data), &file_size);
                R_tfat_f_close(&file);
                R_USB_HmscRelSem();
                    :
            break;
                    :
        }
    }
}
```

7. Return Value (USB_STS_MSC_CMD_COMPLETED) of R_USB_GetEvent Function

(1). Non-OS

After the completion of a Mass Storage command is checked with the *R_USB_HmscStrgCmd* function, if the *R_USB_GetEvent* function is called, then *USB_STS_MSC_CMD_COMPLETE* will be the return value.

(2). RTOS

When a Mass Storage command completes, the callback function that has been registered using the *R_USB_Callback* function will be called by the USB driver. At this time, *USB_STS_MSC_CMD_COMPLETE* will be set to the member (*event*) in the argument (the pointer to the *usb_ctrl_t* structure) of this callback function.

The following shows the information which is set to the member in the *usb_ctrl_t* structure when completing Mass Storage command.

size	:	Size of response data
status	:	CSW information

Note:

1. The member (*size*) has the size of the response data sent from MSC device.
2. The member (*status*) has bCSWStatus of the CSW (Command Status Wrapper):

USB_CSW_SUCCESS	(Value: 00H)	:	Successful
USB_CSW_FAIL	(Value: 01H)	:	Failed
USB_CSW_PHASE	(Value: 02H)	:	Phase error

8. Configuration (r_usb_hmsc_mini_config.h)

Please set the following according to your system.

Note:

Be sure to set *r_usb_basic_mini_config.h* file as well. For *r_usb_basic_mini_config.h* file, refer to chapter "**Configuration**" in the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note.*

1. Setting pipe to be used

   Set the pipe number (PIPE1 to PIPE5) to use for Bulk IN/OUT transfer. Do not set the same pipe number for the definitions of *USB_CFG_HMSC_BULK_IN* and *USB_CFG_HMSC_BULK_OUT*.

```
#define    USB_CFG_HMSC_BULK_IN     Pipe number (USB_PIPE1 to USB_PIPE5)
#define    USB_CFG_HMSC_BULK_OUT    Pipe number (USB_PIPE1 to USB_PIPE5)
```

9. Configuration File (When using RI600V4)

It is necessary to register the OS resource used by HMSC 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: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note*.

9.1 Mailbox Definition

1. Mailbox 1

name	:	ID_USB_RTOS_HMSC_MBX
wait_queue	:	TA_FIFO
message_queue	:	TA_MFIFO

2. Mailbox 2

name	:	ID_USB_RTOS_HMSC_REQ_MBX
wait_queue	:	TA_FIFO
message_queue	:	TA_MFIFO

9.2 Semaphore Definition

name	:	ID_USB_RTOS_HMSC_SEM
max_count	:	1
initial_count	:	1
wait_queue	:	TA_FIFO

9.3 Mutex Definition

name	:	ID_USB_RTOS_TFAT_MTX
ceilpri	:	1

10. Creating an Application

Refer to the chapter "**Creating an Application Program**" in the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) 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	Summery
1.00	Dec 1, 2014	—	First eddition issued.
1.01	Jun 1, 2015	—	RX231 is added in the target device.
1.02	Dec 28, 2015	—	Upgrading of this USB driver by upgrading of "USB Basic Mini Firmware (R01AN2166)".
1.10	Nov 30,2018	—	1. Supporting Smart Configurator. 2. The following chapter is added. (1). 5. Class Driver (2). 6. API Functions (3). 7. Return Value (USB_STS_MSC_CMD_COMPLETED of R_USB_GetEvent Function (4). 8. Configuration (r_usb_hmsc_mini_config.h) 3. The following chapters are changed. (1). 3. API Information (2). 9. Creating an Application 4. The following chapters are deleted. "How to Register Class Driver", "System Resources", "Task ID and Priority Setting","File System Interface", "Host Mass Storage Device Driver", "USB Mass Stoage Class Driver"
1.11	May 31, 2019	—	Support GCC compiler and IAR compiler.
1.12	Jun 30, 2019	—	RX23W is added in the target device.
1.20	Jum 1, 2020	—	Support the real time OS.
1.30	Jul 31, 2024	—	RX261 is added in the target device.

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

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

Contact information

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

Trademarks