RX Family

R01AN0513EJ0233
Rev.2.33
Jul 31, 2019

USB Host Mass Storage Class Driver (HMSC)

Introduction

This application note describes USB Host Mass Storage Class Driver (HMSC). This driver operates in combination with the USB Basic Host Driver (USB-BASIC-F/W). It is referred to below as the HMSC.

Target Device

RX62N/RX621 Group
RX63N/RX631 Group
RX63T Group

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

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. RX62N/RX621 Group User's Manual: Hardware (Document number .R01UH0033)
5. RX63N/RX631 Group User's Manual: Hardware (Document number .R01UH0041)
6. RX63T Group User's Manual: Hardware (Document number .R01UH0238)
7. RX Family M3S-TFAT-Tiny: FAT file system software (Document number.R20AN0038)
8. USB Basic Host and Peripheral Driver Application Note (Document number. R01AN0512)

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

Contents

1. Overview

The HMSC, when used in combination with the USB-BASIC-F/W, 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 this software uses the M3S-TFAT-Tiny (TFAT). Therefore, it should be used in combination with RX Family M3S-TFAT-Tiny: FAT File System Software (document No. R20AN0038EJ).

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.
4. Sharing of a single pipe for IN/OUT directions or multiple devices.
5. Maximum 4 USB storage devices can be connected.

1.1 Please be sure to read

Please refer to the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note* when creating an application program using this driver.
This document is located in the "**reference_documents**" folder within this package.

1.2 Note

1. This driver is not guaranteed to provide USB communication operation. The customer should verify operation when utilizing it in a system and confirm the ability to connect to a variety of different types of devices.
2. Please be sure to use the documentations (*r01an0513ej0232_usb.pdf, r01an0512ej0232_usb.pdf*) under the "reference_documents" folder when using RX62N/RX621/ RX63T.

1.3 Limitations

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. Maximum 4 USB storage devices can be connected.
4. USB storage devices with a sector size of 512 bytes can be connected.
5. A device that does not respond to the READ_CAPACITY command operates as a device with a sector size of 512 bytes.
6. This driver does not support DMA/DTC transfer.

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
TFAT	:	Tiny FAT file system software for microcontrollers (M3S-TFAT-Tiny-RX)
	:	USB Basic Host Driver for (non-OS)
USB	:	Universal Serial Bus

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.

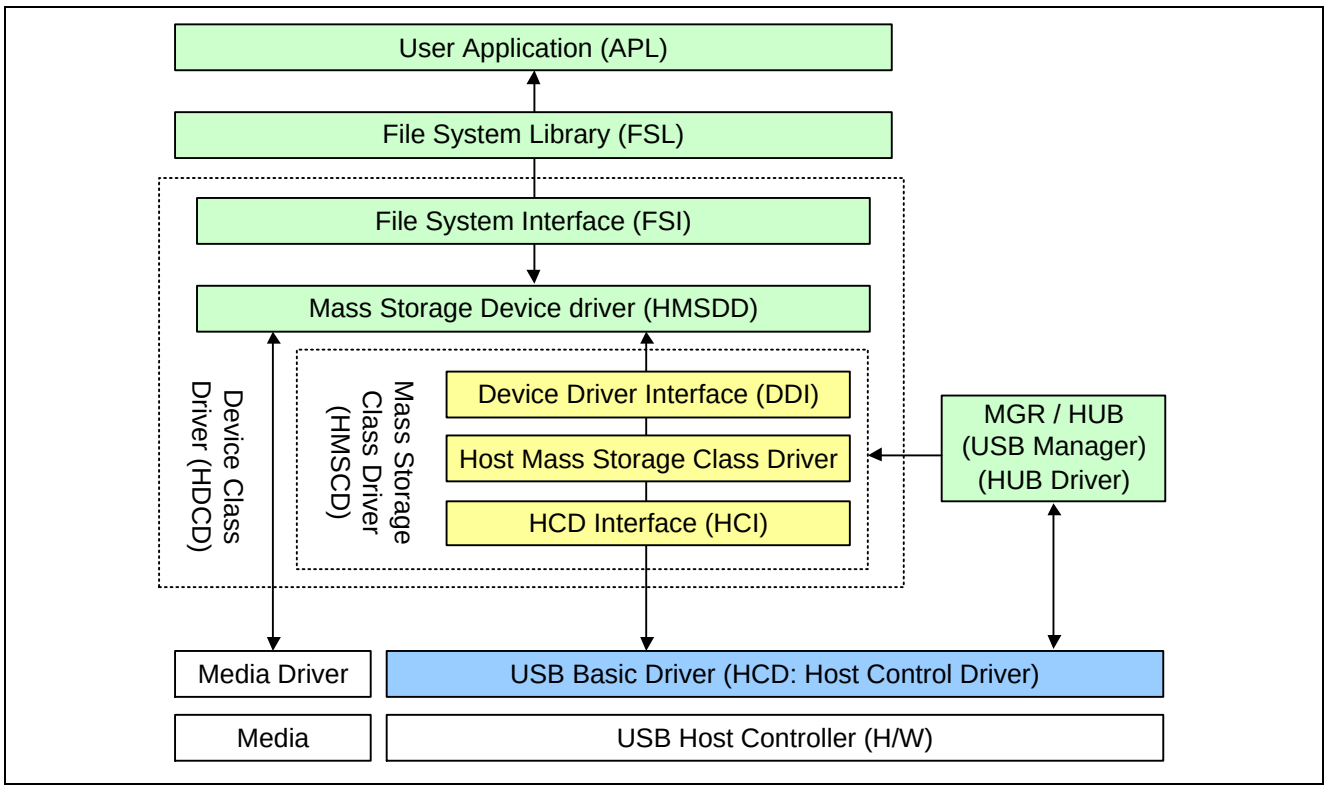


Figure 2-1 Software Block Diagram

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/HUB	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 Operating Confirmation Environment

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

Table 3-1 Operation Confirmation Environment

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

3.3 Usage of Interrupt Vector

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

Table 3-2 List of Usage Interrupt Vectors

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

3.4 Header Files

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

3.5 Integer Types

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

3.6 Compile Setting

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

3.7 ROM / RAM Size

The follows show ROM/RAM size of this driver.

	Checks arguments	Does not check arguments
ROM size	40.2K bytes (Note 3)	39.6K bytes (Note 4)
RAM size	27.5K bytes	27.5K bytes

Note:

1. ROM/RAM size for USB Basic Driver is included in the above size.
2. The default option is specified in the compiler optimization option.
3. The ROM size of "Checks arguments" is the value when *USB_CFG_ENABLE* is specified to *USB_CFG_PARAM_CHECKING* definition in *r_usb_basic_config.h* file.
4. The ROM size of "Does not check arguments" is the value when *USB_CFG_DISABLE* is specified to *USB_CFG_PARAM_CHECKING* definition in *r_usb_basic_config.h* file.

3.8 Argument

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

4. Target Peripheral List（TPL）

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

5. 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_HmscGetDriveNo()	Obtains the drive number.

Note:

1. Uses the FAT (File Allocation Table) API to access storage media.
2. Refer to chapter "API" in the document (Document number: R01AN0512) for *USB Basic Host and Peripheral Driver Application Note* when using other API.

6.1 R_USB_HmscStrgCmd

Issues a Mass Storage command

Format

usb_err_t R_USB_HmscStrgCmd(usb_ctrl_t *p_ctrl, uint8_t *p_buf, uint16_t command)

Arguments

p_ctrl	Pointer to usb_ctrl_t structure area
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 that is specifed by the members (*address* and *module*) in the argument (*p_ctrl*). 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

Mass Storage 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

Reentrant

This API is not reentrant.

Note

1. Before calling this API, assign the module number to the member (*module)* and the device address to the member (*address)*. If something other than *USB_IP0* or *USB_IP1* is assigned to the member (*module*), 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 t 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. Do not assign a pointer to the auto variable (stack) area to the arguments (*p_buf*).
5. Assign *USB_NULL* to the argument (*p_buf*) when issuing the mass storage command without the response data.

6. 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.
7. 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.
8. Refer to chapter "**7. Return Value (USB_STS_MSC_CMD_COMPLETED) of R_USB_GetEvent Function**" about CSW.
9. 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.
10. Set the page code (1 Byte) of the "*Mode Sense10*" command set to the start address of the area indicated by the 2nd argument.
11. 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).
12. 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;
    usb_err_t err;
                    :
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
        {
                    :
            case USB_STS_CONFIGURED:
                    :
                g_buf[0] = 0x3F;     /* Page Code */
                ctrl.module = USB_IP1;
                ctrl.address = adr;
                R_USB_HmscStrgCmd( &ctrl, &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;
                    :
        }
    }
}
```

6.2 R_USB_HmscGetDriveNo

Obtains the drive number

Format

usb_err_t R_USB_HmscGerDriveNo(usb_ctrl_t *p_ctrl, uint8_t *p_drive)

Arguments

p_ctrl	Pointer to usb_ctrl_t structure area
p_drive	Pointer to the area to store the drive number

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 the related drive number. The drive number is stored in the area indicated by the argument (*p_drive*).

Reentrant

This API is reentrant.

Note

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

Example

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

7. Return Value (USB_STS_MSC_CMD_COMPLETED) of R_USB_GetEvent Function

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. In addition, the following members of the *usb_ctrl_t* structure also have information:

module	:	USB module number where Mass Storage command has been completed.
address	:	Device address of USB device where Mass Storage command has been completed.
size	:	Size of response data
status	:	CSW information

Note:

1. The member (*module*) of the *usb_ctrl_t* structure has the USB module number (USB_IP0 / USB_IP1) connected to that USB device. The member (*address*) has the device address of the USB device where the Mass Storage command has been completed.

2. The member (*size*) has the size of the response data sent from MSC device.

3. 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. Sample Application

8.1 Application Specifications

The main functions of the APL are as follows:

1. Performs enumeration and drive recognition processing on MSC devices.
2. After the above processsing finisihes, the APL writes the file “hmscdemo.txt” to the MSC device once.
3. After writing the above file, the APL repeatedly reads the file “hmscdemo.txt.” It continues to read the file repeatedly until the switch is pressed again.

[Note]

When an MSC device on which on which the file “hmscdemo.txt” is stored is connected to the RSK board and the above steps are performed, “hmscdemo.txt” will be overwritten.

8.2 Application Processing

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

8.2.1 Initial Setting

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

8.2.2 Main Loop

This main loop controls the program using the return values from the *R_USB_GetEvent* function and state variables. The return values of the *R_USB_GetEvent* function are used for the USB event management such as attach and detach. File write/read is performed on the MSC device based on the state checked with the state variable.

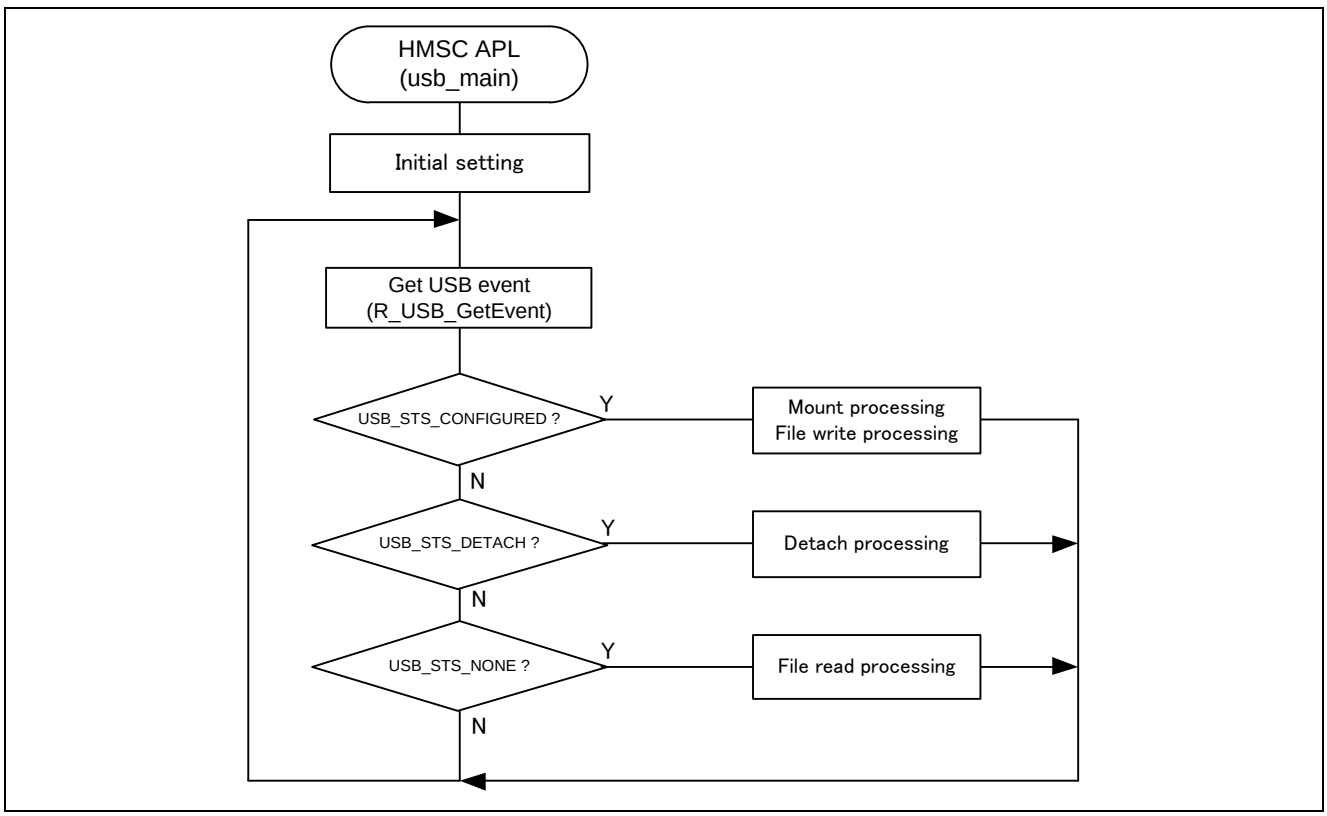


Figure 8-1 Main Loop

1. Mount processing and File write processing (USB_STS_CONFIGURED)

When the *R_USB_GetEvent* function is called after the MSC device is attached to the RSK and the enumeration and drive recognition processing is completed, *USB_STS_CONFIGURED* is set for the return value. After the application program checks that *USB_STS_CONFIGURED* has been set for the return value and performs the Mount processing and the file write processing, then it assigns STATE_FILE_READ to the state management variable to perform the file write/read processing.

2. File read processing (USB_STS_NONE)

When the R_USB_GetEvent function is called with no USB-related event, USB_STS_NONE is set for the return value. The application program checks that USB_STS_NONE has been set for the return value and then performs the following file read processing.

(1). File read processing

Performs the file read processing on an MSC device using the FAT API. After completion of the file read, the file read processing is continued until the MSC is detached.

3. Detach processing (USB_STS_DETACH)

If the *R_USB_GetEvent* is called after the MSC device has been detached from the RSK, then *USB_STS_DETACH* will be the return value. After the application program checks that *USB_STS_DETACH* has been set for the return value, then it assigns STATE_DETACH to the state variable.

8.3 Configuration File for the application program (r_usb_hmsc_apl_config.h)

Make settings for the definitions listed below.

1. USE_USBIP Definition

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

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

[Note]

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

2. Note

The above configuration settings apply to the application program. USB driver configuration settings are required in addition to the above settings. For information on USB driver configuration settings, refer to *USB Host and Peripheral Driver Application Note* (Document number. R01AN0512EJ).

8.4 Connecting Multiple MSC Devices

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

r_usb_hmsc_apl_multi.c

9. Setup

9.1 Hardware

9.1.1 Example Operating Environment

Figure 9-1 shows an example operating environment for the HMSC. Refer to the associated instruction manuals for details on setting up the evaluation board and using the emulator, etc.

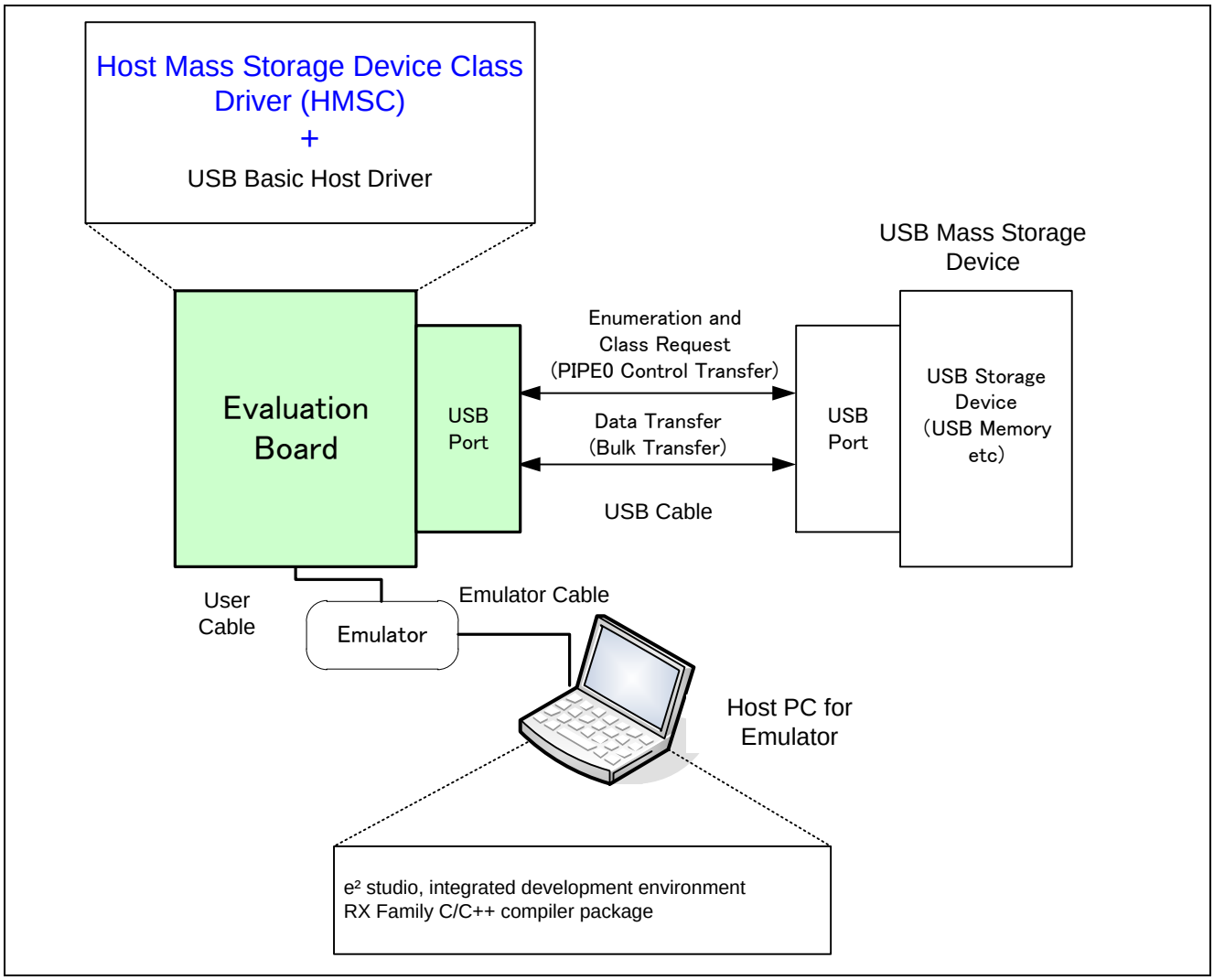


Figure 9-1 Example Operating Environment

9.1.2 RSK Setting

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

Table 9-1 RSK Setting

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

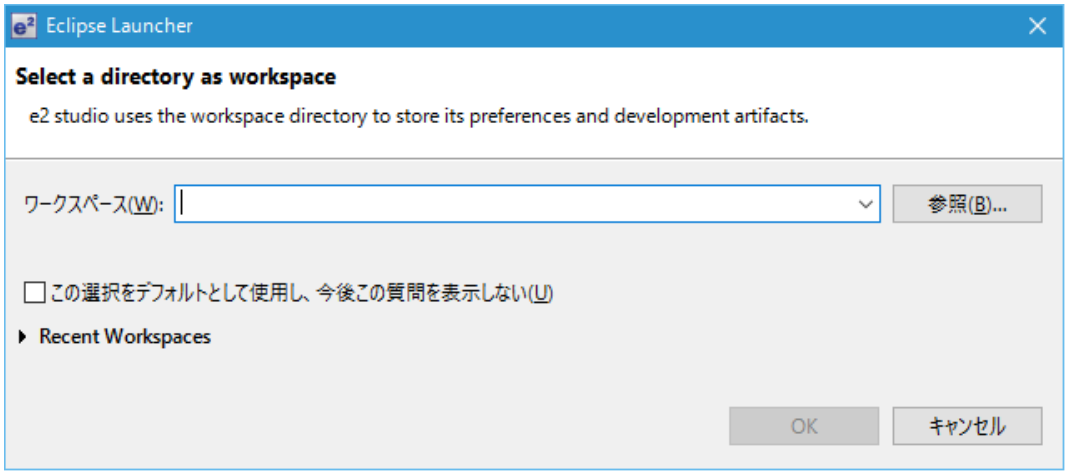
Note:
For the detail of RSK setting, refer to the user's manual of RSK.

9.2 Software

1) Setup e^2 studio

a) Start e^2 studio

b) If you start up e^2 studio at first, the following dialog is displayed. Specify the folder to store the project in this dialog.

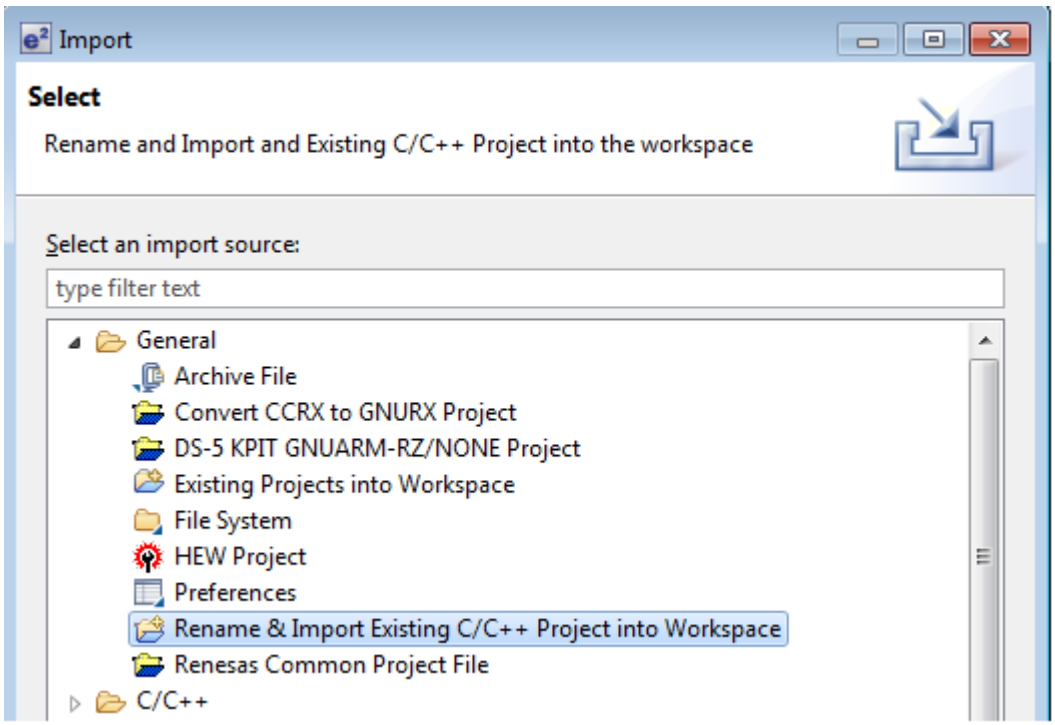


2)

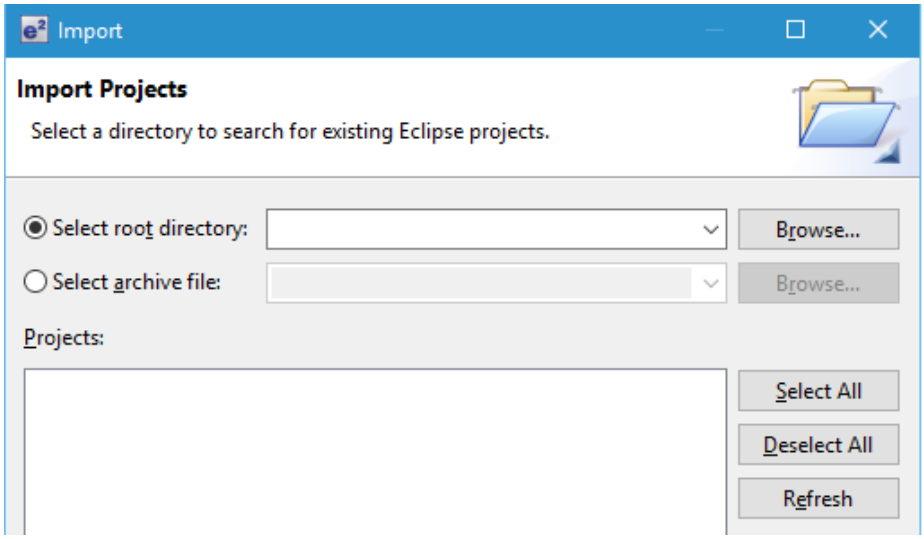
3) Import the project to the workspace

a) Select [File] > [Import]

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



c) Select the root directory of the project, that is, the folder containing the ".cproject" file.



d) Click "Finish".

You have now imported the project into the workspace. Note that you can import other projects into the same workspace.

4) Generate the binary target program by clicking the "Build" button.

5) Connect the target board to the debug tool and download the executable. The target is run by clicking the "Run" button.

10. Creating an Application

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

11. Using the e² studio project with CS+

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

[Note]

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

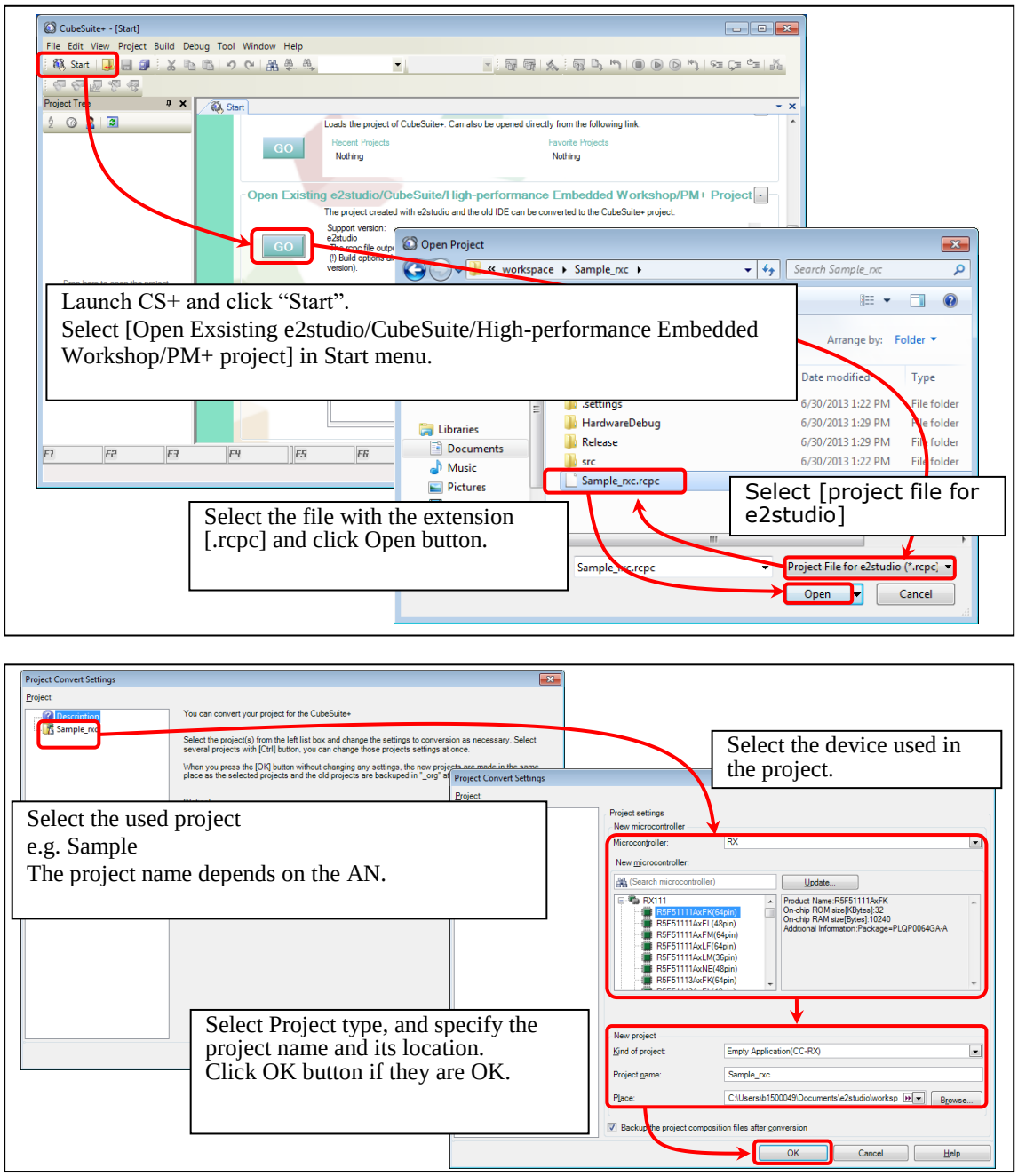


Figure 11-1 Using the e² studio project with CS+

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.13.2010	—	First edition issued
1.10	Aug.10.2011	—	Add target device R8A66597
		2	Add 1.3 Related Documents
		7,8	Add 2.4.1 Folder Structure Update 2.4.2 File Structure
		9	Add 2.5.2 System Resource Definitions when Running Non-OS Version
		9	Update Fig.2.2
		42-54	Add sections 8 and 9
2.00	June. 1.2012	—	Revision of the document by firmware upgrade
2.01	Feb 1.2013	—	“1.4 How To Read This Document” is added and the careless misitake is fixed.
2.10	Apr.1.2013	—	First Release for V.2.10 Add Target Device RX63T.Add the information on RX63T
2.20	Sep.30.2015	—	Change the application program. Change the folder structure. RX63N, RX631 and R8A66597 are deleted from Target Device. The following APIs are added. R_usb_hmsc_alloc_drvno, R_usb_hmsc_free_drvno R_usb_hmsc_ref_drvno The argument “drvno” is added to the following APIs. R_usb_hmsc_SetDevSts, R_usb_hmsc_GetDevSts The argument “ipno” is added to the following APIs. R_usb_hmsc_Information The multiple connecting of MSC device is supported.
2.30	Sep 30, 2016	—	1. Supporting DMA transfer. 2. Supporting USB Host and Peripheral Interface Driver application note(Document No.R01AN3293EJ)
2.31	Sep 30, 2017	—	1. DMA/DTC transfer has been changed to unsupported. 2. The contents of USB Host and Peripheral Interface Driver application note (Document number: R01AN3293EJ) is moved to this document and USB Host and Peripheral Interface Driver application note is deleted.
2.32	Mar 31, 2018	—	The revision of USB Basic driver has been updated.
2.33	Jul 31, 2019	—	RX63N and RX631 are added in Target Device.

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

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

1. Handling of Unused Pins

Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

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

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

- The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

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

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

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

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

(Rev.4.0-1 November 2017)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai 200333, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

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

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

Colophon 8.0