RX Family

R01AN2168EJ0120
Rev.1.20
Jun 1, 2020

USB Host Human Interface Device Class Driver for USB Mini Firmware Using Firmware Integration Technology

Introduction

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

Target Device

RX111 Group
RX113 Group
RX231 Group
RX23W Group

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

Related Documents

1. Universal Serial Bus Revision 2.0 specification
   http://www.usb.org/developers/docs/
2. USB Class Definitions for Human Interface Devices Version 1.1
3. HID Usage Tables Version 1.1
   http://www.usb.org/developers/docs/
4. 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. 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 Devices Page
  http://www.renesas.com/prod/usb/

Contents

1. Overview

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

This module supports the following functions.

- Data communication with a connected HID device (USB mouse, USB keyboard)
- Issuing of HID class requests to a connected HID device
- Supporting Interrupt OUT transfer.

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 Limitations

The following limitations apply to the HHID.

1. The HID driver must analyze the report descriptor to determine the report format (This HID driver determines the report format from the interface protocol alone.)
2. This driver does not support DMA/DTC transfer.

1.4 Terms and Abbreviations

Terms and abbreviations used in this document are listed below.

APL	:	Application program
HCD	:	Host control driver of
HDCD	:	Host device class driver (device driver and USB class driver)
HHID	:	Host human interface device
HID	:	Human interface device class
IDE	:	Integrated Development Environment
MGR	:	Peripheral device state manager of HCD
RSK	:	Renesas Starter Kits
RTOS	:	USB driver for the real-time OS
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Basic Mini Host and Peripheral Driver

1.5 USB HHID FIT

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. Module Configuration

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

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

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

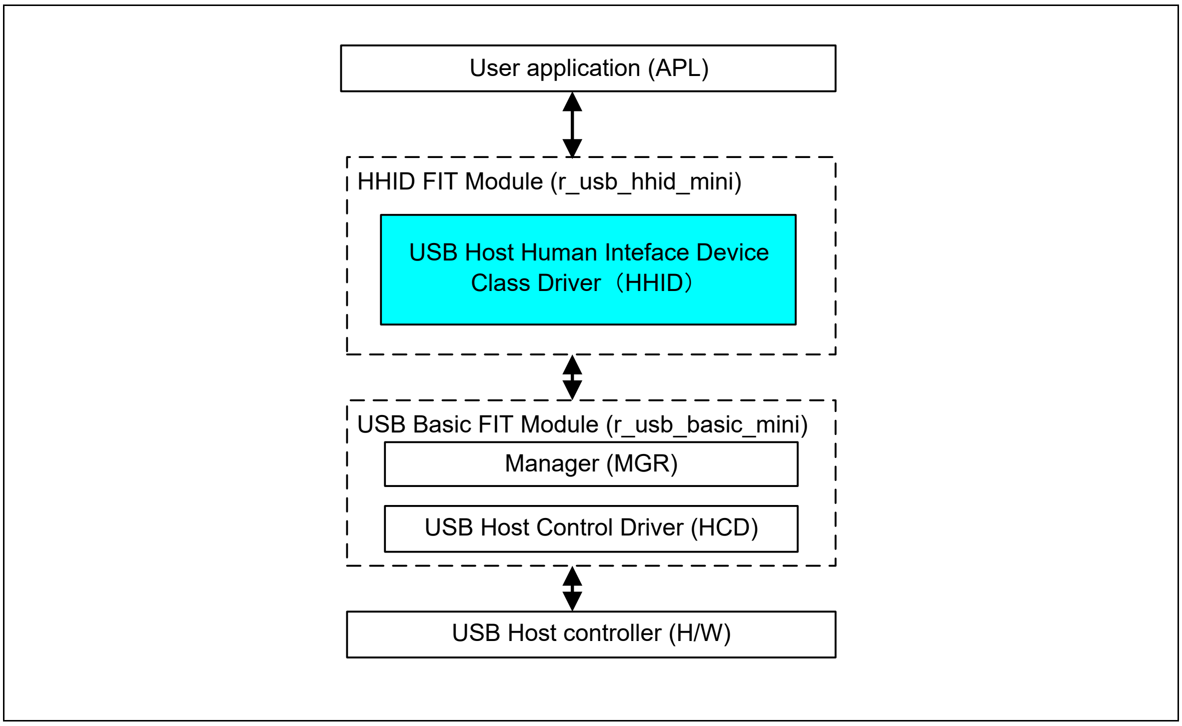

Figure 2.1 Software Module Structure

Table 2-1 Module Function Descriptions

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

3. API Information

This Driver API follows the Renesas API naming standards.

3.1 Hardware Requirements

This driver requires your MCU support the following features:

- USB

3.2 Software Requirements

This driver is dependent upon the following packages:

- r_bsp
- r_usb_basic_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.3.02.00 (The option "-lang=C99" is added to the default setting of IDE)
	GCC for Renesas RX 8.3.0.201904 (The option "-std=gnu99" is added to the default setting of IDE)
	IAR C/C++ Compiler for Renesas RX version 4.14.1
Real-Time OS	FreeRTOS V.10.0.0 RI600V4 V.1.06
Endian	Little Endian, Big Endian
USB Driver Revision Number	Rev.1.20
Using Board	Renesas Starter Kit for RX111 Renesas Starter Kit for RX113 Renesas Starter Kit for RX231 Renesas Solution Starter Kit for RX23W

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	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_hhid_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 **7, Configuration (r_usb_hhid_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	19.1K bytes (Note 3)	18.8K bytes (Note 4)
RAM size	3.7K bytes	3.7K bytes

(2). RI600V4

	Checks arguments	Does not check arguments
ROM size	36.1K bytes (Note 3)	35.8K bytes (Note 4)
RAM size	4.4K bytes	4.4K bytes

(3). FreeRTOS

	Checks arguments	Does not check arguments
ROM size	32.2K bytes (Note 3)	31.9K bytes (Note 4)
RAM size	4.4K bytes	4.4K bytes

2. GCC (Optimization Level: -O2)

	Checks arguments	Does not check arguments
ROM size	22.7K bytes (Note 3)	22.4K bytes (Note 4)
RAM size	3.3K bytes	3.3K bytes

3. IAR (Optimization Level: Medium)

	Checks arguments	Does not check arguments
ROM size	16.1K bytes (Note 3)	15.9K bytes (Note 4)
RAM size	2.5K bytes	2.5K bytes

[Note]

1. ROM/RAM size for BSP and USB Basic Driver is included in the above size.
2. The above is the size when specifying RX V2 core option.
3. The ROM size of “Checks arguments” is the value when *USB_CFG_ENABLE* is specified to *USB_CFG_PARAM_CHECKING* definition in *r_usb_basic_mini_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_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 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. Human Interface Device Class (HID)

5.1 Basic Functions

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

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

5.2 Class Requests (Host to Device Requests)

This driver supports the following class requests.

For the class request processing, refer to chapter "**USB Class Requests**" in the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note.*

Table 5-1 HID Class Requests

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

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

a). GetReport Request Format

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

Table 5-2 GetReport Format

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

b). SetReport Request Format

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

Table 5-3 SetReport Format

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

c). GetIdle Request Format

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

Table 5-4 GetIdle Format

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

d). SetIdle Request Format

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

Table 5-5 SetIdle Format

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

e). GetProtocol Request Format

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

Table 5-6 GetProtocol Format

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

f). SetProtocol Request Format

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

Table 5-7 SetProtocol Format

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

5.3 HID-Report Format

5.3.1 Receive Report Format

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

Table 5-8 Receive Report Format

Offset	Keyboard Mode	Mouse Mode
Data length	8 Bytes	3 Bytes
0 (Top Byte)	Modifier keys	b0: Button 1 b1: Button 2 b2-7: Reserved
+1	Reserved	X displacement
+2	Keycode 1	Y displacement
+3	Keycode 2	-
+4	Keycode 3	-
+5	Keycode 4	-
+6	Keycode 5	-
+7	Keycode 6	-

5.3.2 Transmit Report Format

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

Table 5-9 Transmit Report Format

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

5.3.3 Note

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

6. API Functions

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

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

Note:

Refer to chapter "API" in the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note* when using the other API.

6.1 R_USB_HhidGetType

Obtains type information for the HID device.

Format

usb_err_t R_USB_HhidGetType(uint8_t *p_type)

Arguments

p_drive Pointer to the area to store the type information

Return Value

USB_SUCCESS Successfully completed
USB_ERR_PARA Parameter error
USB_ERR_NG Other error

Description

This API obtains type information (mouse, keyboard, etc.) for the connected HID device. The type information is set to the area indicated by the argument (*p_type*). For the type information to be set, see Table 6-1.

Table 6-1 Type Information

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

Note

1. If *USB_NULL* is assigned to the argument (*p_type*), then *USB_ERR_PARA* will be the return value.
2. 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 )
{
    uint8_t   type;
                :
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
        {
                :
            case USB_STS_CONFIGURED:
                :
                R_USB_HhidGetType( &type );
                if( USB_HID_KEYBOARD == type )
                {
                :
                }
                :
            break;
                :
        }
    }
}
```

6.2 R_USB_HhidGetMxps

Obtains the max packet size of the HID device.

Format

usb_err_t R_USB_HhidGetMxps(uint16_t *p_mxps, uint8_t dir)

Arguments

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

Return Value

USB_SUCCESS	Successfully completed
USB_ERR_PARA	Parameter error
USB_ERR_NG	Other error

Description

This API obtains max packet size for the connected HID device. The max packet size is set to the area indicated by the 1st argument (*p_mxps*).

Set the direction (USB_IN / USB_OUT) of the max packet size which the user want to obtain to the 2nd argument.

Note

1. If *USB_NULL* is assigned to the argument (*p_mxps*), then *USB_ERR_PARA* will be the return value.
2. This function returns *USB_ERR_PARA* when the connected HID device does not support the transfer direction set the 2nd argument (*dir*).
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 )
{
   uint16_t  mxps;
   usb_ctrl_t ctrl;
              :
   while (1)
   {
      switch (R_USB_GetEvent(&ctrl))
       {
              :
         case USB_STS_CONFIGURED:
              :
            R_USB_HhidGetMxps(&mxps, USB_IN);
              :
         break;
              :
      }
   }
}
```

7. Configuration (r_usb_hhid_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 (PIPE6 to PIPE9) to use for Interrupt IN transfer and Interrupt OUT. Do not set the same pipe number.

```
#define    USB_CFG_HHID_INT_IN     Pipe number (USB_PIPE6 to USB_PIPE9)
#define    USB_CFG_HHID_INT_OUT    Pipe number (USB_PIPE6 to USB_PIPE9)
```

8. Configuration File (When using RI600V4)

It is necessary to register the OS resource used by HHID USB driver to RI600V4 when using RI600V4. Please add the following definition in the configuration file. For how to create the configuration file, refer to the chapter, "**RI600V4(Configuration File Creation)**" in the document (Document number: R01AN2166) for *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note.*

5.4 Mailbox Definition

name	:	ID_USB_RTOS_HHID_MBX
wait_queue	:	TA_FIFO
message_queue	:	TA_MFIFO

9. Creating an Application

Refer to the chapter "**Creating an Application Program**" in the document (Document number: 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	Summary
1.00	Dec 1, 2014	—	First edition 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). 7. Configuration (r_usb_hhid_mini_config.h) 3. The following chapters are changed. (1). 3. API Information (2). 5. Human Interface Device Class (HID) (3). 6. API Functions (4). 9. Creating an Application 4. The following chapters are deleted. "How to Register Class Driver", "System Resources", "Task ID and Priority Setting".
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	Jun 1, 2020	—	Support the real time OS.

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.

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

Renesas Electronics America Inc. San Jose Campus
6024 Silver Creek Valley Road, San Jose, CA 95138, USA
Tel: +1-408-284-8200, Fax: +1-408-284-2775

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, #06-02 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 9.0