R01AN0546EJ0215
Rev.2.15
Mar 28, 2016

Renesas USB MCU

USB Peripheral Human Interface Devices Class Driver (PHID) using Basic Mini Firmware

Introduction

This document is a manual describing use of the USB Peripheral Human Interface Devices Class Driver (PHID) built using the USB Basic Mini Firmware.

Target Device

RL78/G1C, RL78/L1C, R8C/3MU, R8C/3MK, R8C/34U, R8C/34K

This program can be used with other microcontrollers that have the same USB module as the above target devices. When using this code in an end product or other application, its operation must be tested and evaluated thoroughly.

This program has been evaluated using the corresponding MCU's Renesas Starter Kit board.

Contents

1. Overview

This document describes the USB Peripheral Human Interface Devices Class Driver (PHID), which is based on with the USB Basic Mini Firmware of Renesas USB MCU.

1.1 Functions and Features

The USB Peripheral Human Interface Device class driver conforms to the USB Human Interface Device class specifications (referred to here as HID) and implements communication with a HID host.

This class driver is intended to be used in combination with the USB Basic Mini Firmware, provided from Renesas Electronics.

1.2 Related Documents

1. Universal Serial Bus Revision 2.0 specification
2. USB Device Class Definition for Human Interface Devices (HID) 1.11
   [http://www.usb.org/developers/docs/]
3. USB HID Usage Tables Version 1.12
   [http://www.usb.org/developers/docs/]
4. User's Manual: Hardware
5. USB-BASIC-F/W Application Note

Available from Renesas Electronics WebSite

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

1.3 Terms and Abbreviations

Terms and abbreviations used in this document are listed below.

API	:	Application Program Interface
APL	:	Application program
cstd	:	Prefix for function and file of Host & Peripheral Common Basic (USB low level) F/W
HID	:	Human Interface Device class
HID Host	:	HID class USB Host
HM	:	Hardware Manual
H/W	:	“Hardware”; Renesas USB MCU
KBD	:	Keyboard device
MSE	:	Mouse device
PCD	:	Peripheral control driver of USB-BASIC-F/W
PDCD	:	Peripheral device class driver (device driver and USB class driver)
PHID	:	Peripheral Human Interface Devices
PP	:	Pre-processed definition
pstd	:	Prefix for peripheral function of USB-BASIC-FW
RSK	:	Renesas Starter Kit
Scheduler	:	Used to schedule functions, like a simplified OS.
Scheduler Macro	:	Used to call a scheduler function
SW1/SW2/SW3	:	User switches on the RSK
Task	:	Processing unit
USB	:	Universal Serial Bus
USB-BASIC-F/W	:	USB-BASIC-FW (Host & Peripheral USB Basic Mini Firmware (USB low level) for Renesas USB MCU)

1.4 How to Read This Document

This document is not intended for reading straight through. Use it first to learn the demo, then to look up information on functionality and interfaces as needed for your particular solution.

To get acquainted with the source code, read Chapter 4.3 and note which MCU-specific files you need to copy into directory "***devicename**\src\HwResource*". Observe which files belong to the application level.

Chapter 4, explains how the default Peipheral HID demo application works. You will change this to create your own solution.

Understand how all code modules are divided into tasks, and that these tasks pass messages to one another. This is so that functions (tasks) can execute in the order determined by a scheduler and not strictly in a predetermined order. This plus the use of a function callback mechanism enables the USB code to be non-blocking. The task mechanism is described in Chapter 1.2 above, "USB-BASIC-FW Application Note".

All PHID tasks are listed in Chapter 4.4.

2. How to Register Class Driver

For the class driver layer to be used, it must be registered with the USB-BASIC-FW.

Please consult function *usb_phid_driver_registration()* in *r_usb_phid_apl.c* and register a class driver into a USB-BASIC-FW. For details, please refer to USB-BASIC-FW application note.

3. Operating Confirmation Environment

3.1 Compiler

The compilers which is used for the operating confirmation are follows.

a. CA78K0R Compiler V.1.71

b. CC-RL Compiler V.1.01

c. IAR C/C++ Compiler for RL78 version 2.10.4

d. KPIT GNURL78-ELF v15.02

e. C/C++ Compiler Package for M16C Series and R8C Family V.6.00 Release 00

3.2 Evaluation Board

The evaluation boards which is used for the operating confirmation are follows.

a. Renesas Starter Kit for RL78/G1C (Product No: R0K5010JGC001BR)

b. Renesas Starter Kit for RL78/L1C (Product No: R0K50110PC010BR)

c. R8C/34K Group USB Peripheral Evaluation Board (Product No: R0K5R8C34DK2PBR)

4. Software Configuration

4.1 Module Configuration

PHID comprises the peripheral HID class driver and device drivers for mouse, keyboard, and vendor demonstration. The following figure shows the structure of the PHID software modules, and Table4-1 lists the modules with an overview of each.

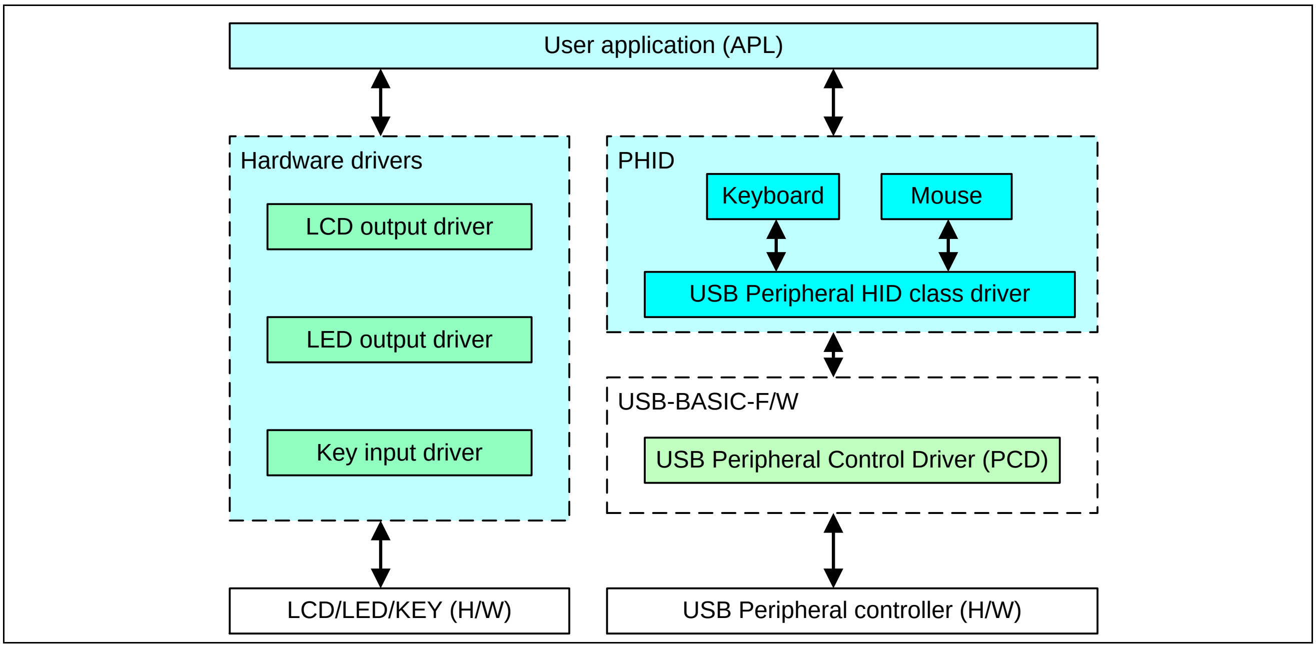


Figure 4-1 Module Structure

Table4-1 Module Function Descriptions

Module Name	Function Description
APL	User application program Switches initiate communication with HID host and control remote wakeup. The LCD displays the information of device state.
PHID	User switch operation on the RSK board is converted into HID reports. The following data transfers are requested of USB-BASIC-FW, depending on which mode the program is in 1) Send HID report for mouse movement 2) Send HID report for keyboard movement The transfer result is notified to APL by the callback function. In addition, communicate the output report of HID host to APL. 1) LED data for keyboard control
USB-BASIC-FW	USB Basic Mini Firmware (Peripheral Hardware Control)

4.2 Overview of Application Program

The main purpose of the SW package is to provide

1) A basic HID class driver, with device drivers to implement
   a) Mouse
   b) Keyboard
2) A simple demo application for each of above.
3) A complete HID host (PC) <=> peripheral demo that also engages board switches, LCD, LED, and the reading of an ADC value.

The demo operation modes are defined in "*\devicename\src\PHID\inc\usb_class_usrcfg.h*". One of the following is run at a given time:

1) Mouse
2) Keyboard

The HID Demo receives user board switch data by making a HID report for each user switch push. PHID sends the HID report data to the host via PCD.

Also, other HID host requests are accepted via PCD and reported to the APL for processing.

4.3 Structure of Files and folders

4.3.1 Folder Structure

The folder structure of the files supplied with the device class is shown below.

The source codes dependent on each device and evaluation board are stored in each hardware resource folder (***devicename****src**HwResource*).

```
workspace
  ＋[ RL78 / R8C ]
    ＋[ CCRL / CS+ / IAR / e2 studio / HEW ]
      ＋[RL78G1C / RL78L1C / R8C3MK / R8C3MU / R8C34K / R8C34U]
          ＋ PHID_KBD                                Keyboard build result
          ＋ PHID_MSE                                Mouse build result
          ＋ src
            ＋――― PHID [ Human Intergace Device Class driver ]  See Table4-2
            ｜         ＋――― inc                     Common header file of HID driver
            ｜         ＋――― src                     HID driver
            ＋―――SmplMain [ Sample Application ]
            ｜         ＋――― APL                     Sample application
            ＋―――USBSTDFW [Common USB code that is used by all USB firmware ]
            ｜         ＋――― inc                     Common header file of USB driver
            ｜         ＋――― src                     USB driver
            ＋――― HwResource [Hardware access layer; to initialize the MCU ]
                        ＋――― inc                    Common header file of hardware resource
                        ＋―――src                     Hardware resource
```

[Note]

a. The project for CA78K0R compiler is stored under the CS+ folder.

b. The project for KPIT GNU compiler is stored under the e^2 studio folder.

c. Refer to **11 Using the e2 studio project with CS+** section when using CC-RL compiler on CS+.

4.3.2 HID File Structure

Table4-3 shows the PHID file structure.

Table4-3 PHID Folders

Folder Name	File Name	Description	Note
PHID/inc	r_usb_phid_api.h	PHID API function define	
	r_usb_class_usrcfg.h	USB HID class define file	PHID mode select
	r_usb_phid_define.h	HID class type definitions and macro definitions	
PHID/src	r_usb_phid_api.c	PHID API functions	
	r_usb_phid_driver.c	PHID driver	
SmplMain	main.c	Main function	Sample source
SmplMain/ APL	r_usb_phid_apl.c	Sample application program	Sample source
	r_usb_phid_descriptor.c	PHID class descriptor table	Sample source

4.4 System Resources

Table4-4 lists ID and priority definitions used to register PHID with the non-OS scheduler.
These are defined in the *r_usb_ckernelid.h* header file.

Table4-4 Scheduler Registration IDs and Priorities

	Name	Description
Scheduler registration task	USB_PHID_TSK	PHID main task (R_usb_phid_task) Task priority : 1
	USB_PHIDSMP_TSK	APL task (usb_phid_main_task) Task priority : 2
	USB_PCD_TSK	PCD task (R_usb_pstd_PcdTask) Task priority : 0
Mailbox ID	USB_PHID_MBX (default value: USB_PHID_TSK)	PHID mailbox ID
	USB_PHIDSMP_MBX (default value: USB_PHIDSMP_TSK)	APL mailbox ID
	USB_PCD_MBX (default value: USB_PCD_TSK)	PCD mailbox ID

5. Hardware Environment

The physical environment in which the PHID firmware is intended to operate is shown below.

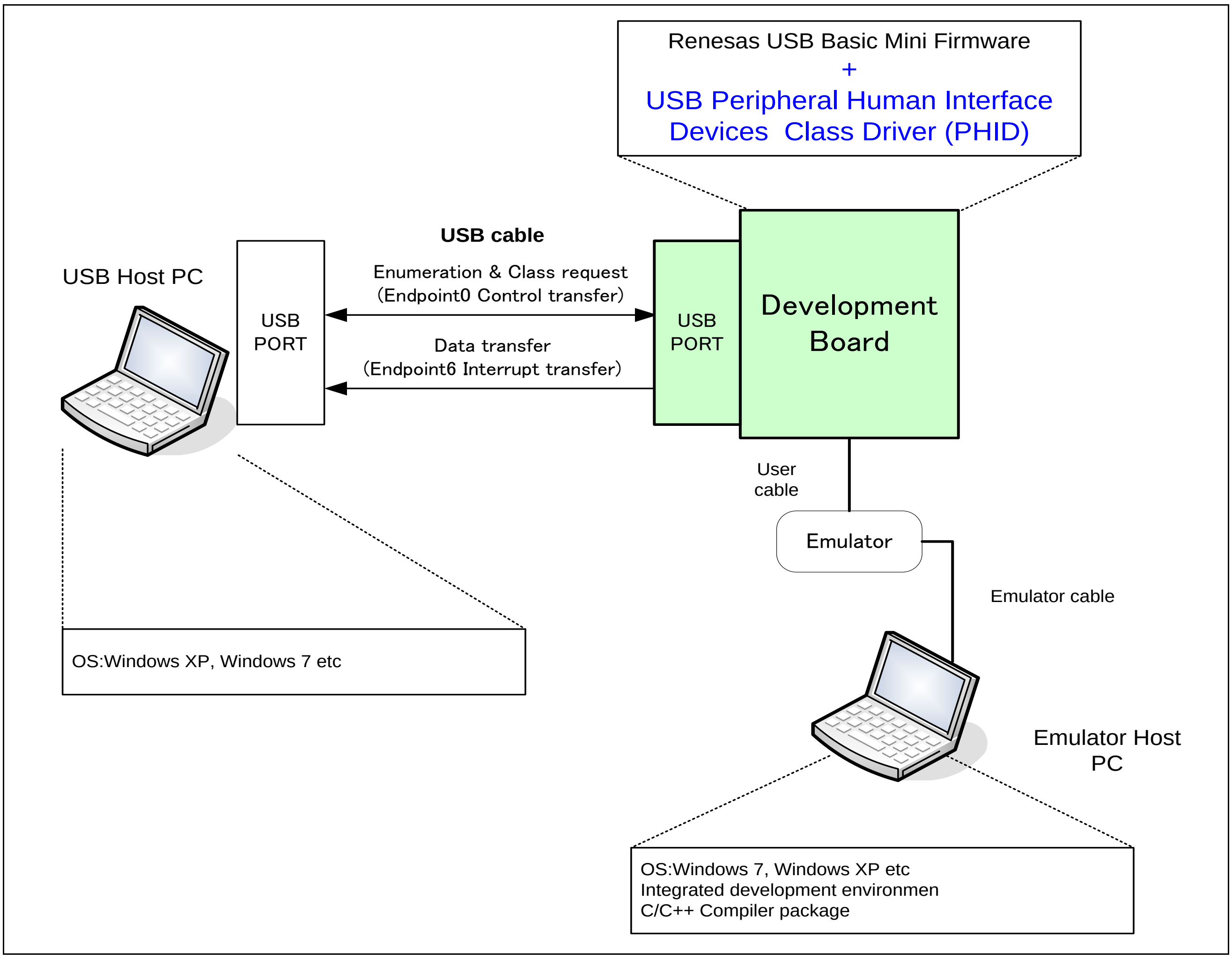


Figure 4-1 The physical environment for the PHID firmware.

6. Peripheral HID Application

6.1 Demo Summary

The peripheral HID sample application (APL) has three operation modes.

(a) Mouse mode

The target sends mouse HID reports, like a USB mouse, when a board switch 1 to 3 is pressed.

(b) Keyboard mode

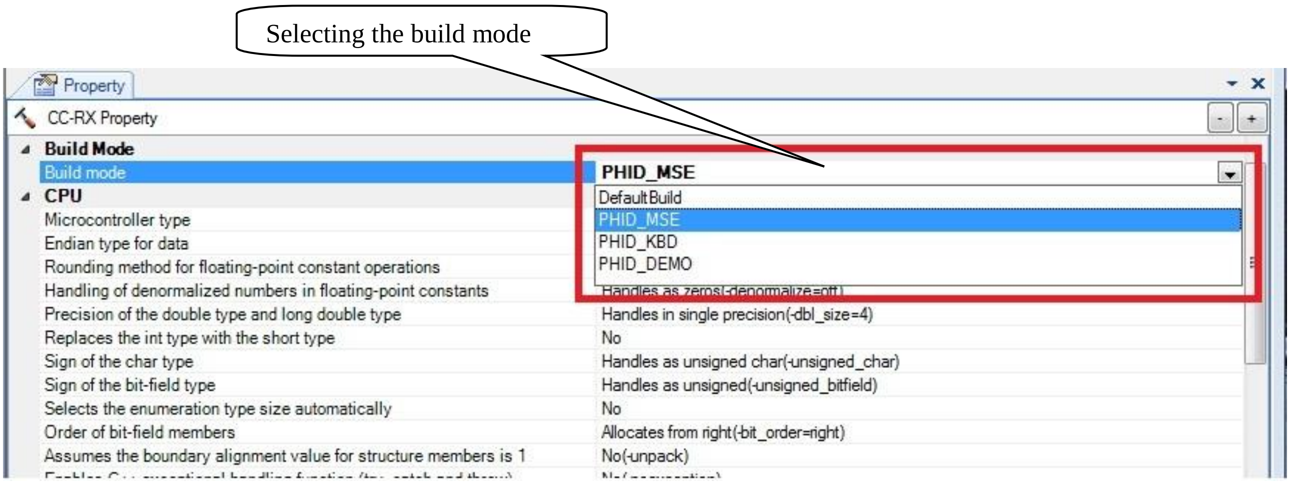
The target sends keyboard HID reports, like a USB keyboard, when board switch 2 or 3 is pressed.

6.2 Selecting Peripheral HID mode

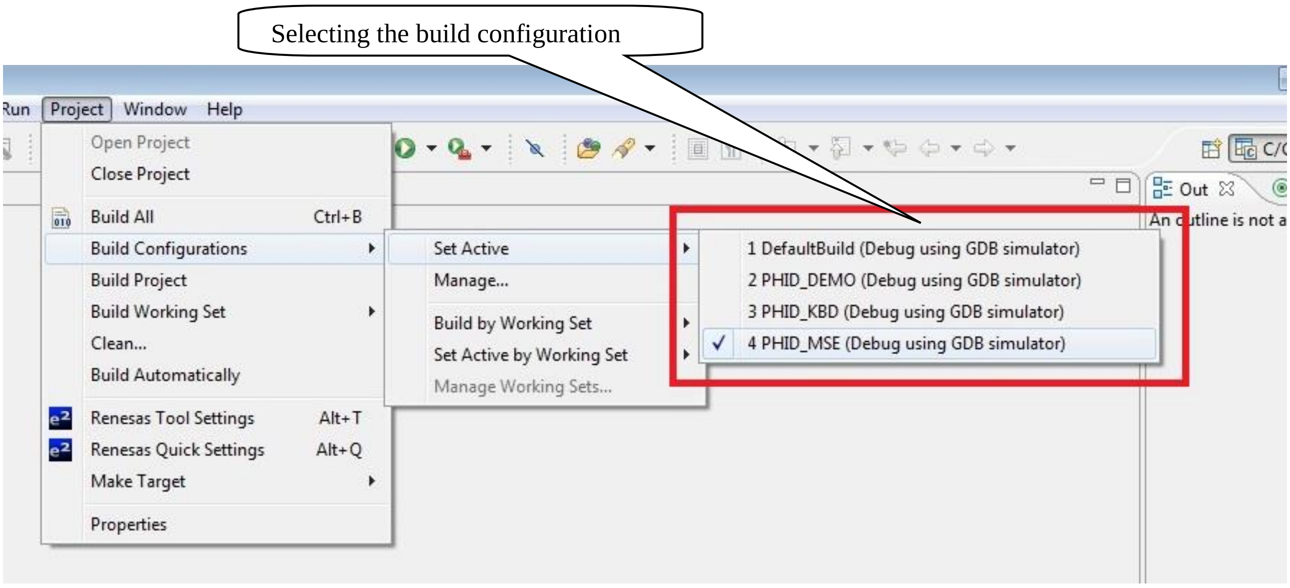
Select Peripheral HID mode on the integrated development environment (IDE) after starting the IDE is supported by each MCU.

Keyboard mode	:	PHID_KBD
Mouse mode	:	PHID_MSE

1) CS+



2) e² studio



6.3 Vendor ID/Product ID

The VID and PID values of the peripheral device are defined by USB_VENDORID and USB_PRODUCTID, respectively, in file "*../HID/inc/r_usb_phid_usrcfg.h*". Default value for Vendor ID is 0x0000. Default value for Product ID is in HID demo mode 0x0003, in Keyboard mode 0x0013, and in Mouse mode 0x0023. These must be changed by the product manufacturer to their registered VID (and a PID administered internally by the company).

6.4 Application Program Detail

Main features of the application:

1). In Keyboard mode, switch presses are converted to key codes that are reported to the HID host.

2). In Mouse mode, switch presses are converted to X/Y coordinates, or mouse button data. These are sent as HID reports to the HID host.

6.4.1 Program Processing

Detailed processing performed by the application program is described below.

(1) Mouse Mode

In Mouse mode, SW1, SW2 or SW3 on the RSK are pressed as shown below.

Table6-1 Simple Mouse Manipulations

RSK Switch Number	Operation
SW2	Reports motion data for mouse up/down direction. Lateral motion accelerates if switch is held down continuously, and stops when the switch is released. If the button is pressed again immediately, the pointer moves in the opposite direction. Up direction switch press time. Short : Distance 1 (Min:1～Max:127) Midterm : Distance 2 Long : Distance 4 Down direction switch press time. Short : Distance -1 (Min:-1～Max:-127) Midterm : Distance -2 Long : Distance -4
SW3	Reports motion data for mouse right/left direction. Vertical motion accelerates if switch is held down continuously, and stops when the switch is released. If the button is pressed again immediately, the pointer moves in the opposite direction. Right direction switch press time. Short : Distance 1 (Min:1～Max:127) Midterm : Distance 2 Long : Distance 4 Left direction switch press time. Short : Distance -1 (Min:-1～Max:-127) Midterm : Distance -2 Long : Distance -4
SW1	Left click.

The HID report is transferred to USB host using interrupt IN transfers.

The Mouse INPUT Report format is shown in Table6-2 .

Table6-2 Mouse INPUT Report format

offset(Byte)	Value
0	Mouse button click information b0：Button 1（Left click ） b1：Button 2（Right click ：Not used） b2-b7：Reserved
1	Left-Right direction and distance（X displacement：-127～127）
2	Up-Down direction and distance（Y displacement：-127～127）

Notes:

The report format used for data communication follows the Report Descriptor format of the the HID class specification.

User modifications must conform to the HID class specifications.

(2) Keyboard Mode

In Keyboard mode, SW2 or SW3 on the RSK are pressed as shown below.

Table 6-3 Simple Keyboard Manipulations

RSK Switch Number	Operation
SW2	One of the key codes for characters “a” to “z” or “Enter” is reported to the host each time SW is pressed.
SW3	One of the key codes for “1” to “9” and “0” or “Enter” is notified to the host each time SW is pressed.

The key code is incremented when the same switch is pressed again. The key code is reset when the other SW2/SW3 is pressed.

Switch operation example:

A text editor must be active on host PC.

1. SW2 is pressed 30 times -> Text editor input "a,b,c,d....x,y,z,<Enter>,a,b,c"
2. SW3 is pressed 14 times -> Text editor input "1,2,3,4....8,9,0,<Enter>,1,2,3"
3. SW2 is pressed 3 times -> Text editor input "a,b,c"
4. SW3 is pressed 3 times -> Text editor input "1,2,3"

Text editor input result below.

```
abcdefghijklmnopqrstuvwxyz
abc1234567890
123abc123
```

The INPUT Report is transmitted to USB host using Interrupt IN transfers.

Keyboard INPUT Report format is shown in Table 6-4 .

Table 6-4 Keyboard INPUT Report format

offset（Byte)	Value
0	Modifier keys : Not used (=0x00) Bit0:Left Control Bit1:Left Shift Bit2:Left Alt Bit3:Left GUI Bit4:Right Control Bit5:Right Shift Bit6:Right ALT Bit7:Right GUI
1	Reserved : 0x00
2	Keycode 1 : Value depends on number of switch presses.
3	Keycode 2 : Not used (=0x00)
4	Keycode 3 : Not used (=0x00)
5	Keycode 4 : Not used (=0x00)
6	Keycode 5 : Not used (=0x00)
7	Keycode 6 : Not used (=0x00)

Key codes are defined in the related document "USB HID Usage Tables Version 1.12". This software uses Key codes as shown in Table 6-5 .

Table 6-5 Key codes

Usage Name	a	b	c	d	e	f	g	h	i	j	k	l	m
Usage ID(HEX)	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
Usage Name	n	o	p	q	r	s	t	u	v	w	x	y	z
Usage ID(HEX)	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
Usage Name	1	2	3	4	5	6	7	8	9	0	Enter		
Usage ID(HEX)	1E	1F	20	21	22	23	24	25	26	27	28		

(Usage Name : Key name,　Usage ID： Key codes)

The keyboard LED (one of the RSK LEDs) is controlled by an OUTPUT Report which is transferred by a SetReport request from host.

The Keyboard OUTPUT Report format is shown in Table 6-6.

Table 6-6 Keyboard OUTPUT Report format

offset（Byte）	Value
0	b0：LED 0(NumLock) b1：LED 1(CapsLock) b2：LED 2(ScrollLock) b3：---- b4：---- 0:OFF,　1:ON

Notes:

The report format used for data communication follows the HID class report descriptor specifications. User modifications must also conform to these HID class specifications.

6.5 Application Functions

Table 3-9 lists the HID peripheral application level (APL) functions.

Table6-9 APL Functions

	Function Name	Description
1	usb_phid_driver_registration	Register the PHID driver
2	usb_phid_apl_open	USB PHID device class open function
3	usb_phid_apl_close	USB PHID Device class close function
4	usb_phid_main_task	PHID demo sample application task
5	usb_phid_apl_trans_cb	HID Host Tx complete callback
6	usb_phid_apl_init	Initialize the application
7	usb_phid_apl_clear	Initialize the application (for detach)
8	usb_phid_apl_keyboard_led_control	Keyboard mode SetReport processing
9	usb_phid_apl_kbd_data_set	Keyboard data creation processing
10	usb_phid_demo_mode_report	HID demo mode SetReport processing
11	usb_phid_apl_create_report	Input report data creation processing
12	usb_phid_apl_mse_data_set	Mouse data creation processing
13	usb_phid_apl_key_on_wakeup	Remote wakeup
14	usb_phid_smpl_message_send	Transfer message to message box of demo sample application
15	usb_phid_disconnected	Get the USB cable connection state
16	usb_psmpl_main_init	Initialize the application program
17	usb_phid_change_device_state	Change device state callback
18	usb_phid_apl_DummyFunction	Dummy function
19	usb_phid_apl_memcpy	Memory copy
20	usb_phid_apl_memset	Memory set
21	usb_phid_apl_AdData	Read AD data

6.6 Operation Sequence

This section shows the operation sequence of the PHID program.

6.6.1 Startup to PHID Enumeration

The figure below shows the sequence from a hardware reset to activation of the PHID driver.

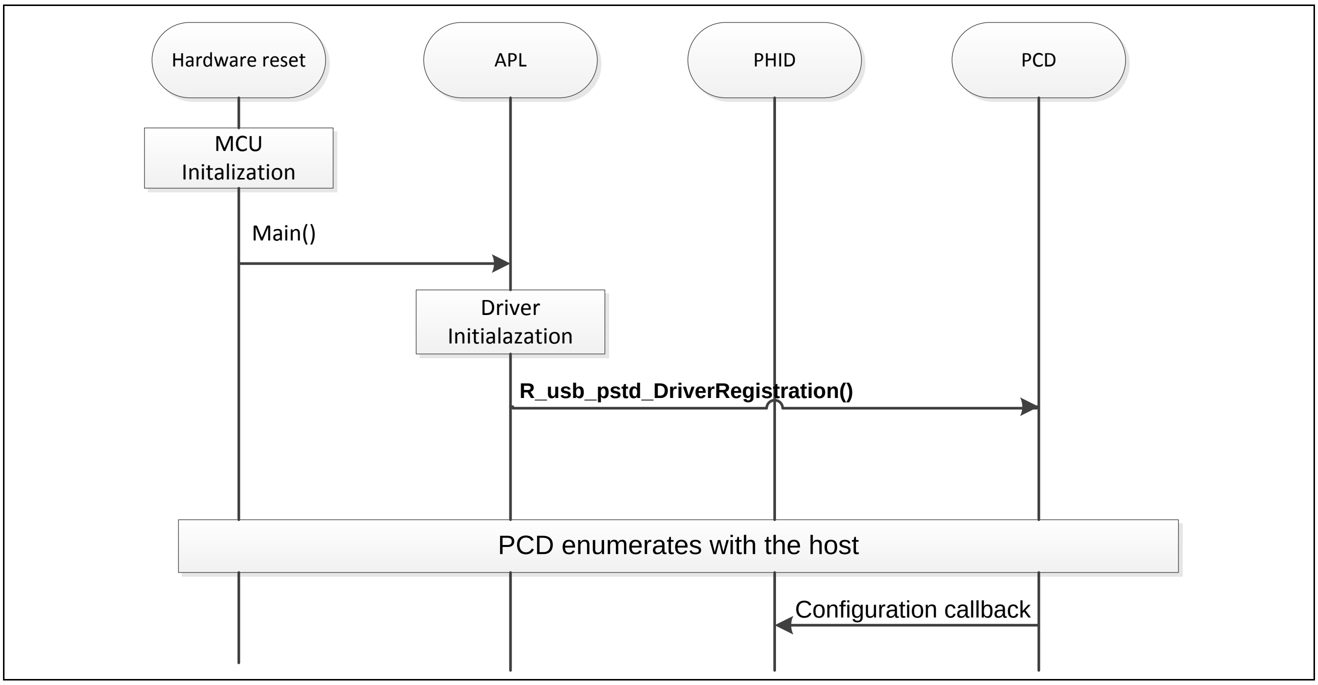


Figure 6-3 Startup Sequence

6.6.2 Notifying Host of User Key Presses and other Events

The figure below shows the data communication sequence for user key manipulations and other events from the peripheral application. R_usb_phid_send_data() is used for report transfers to host.

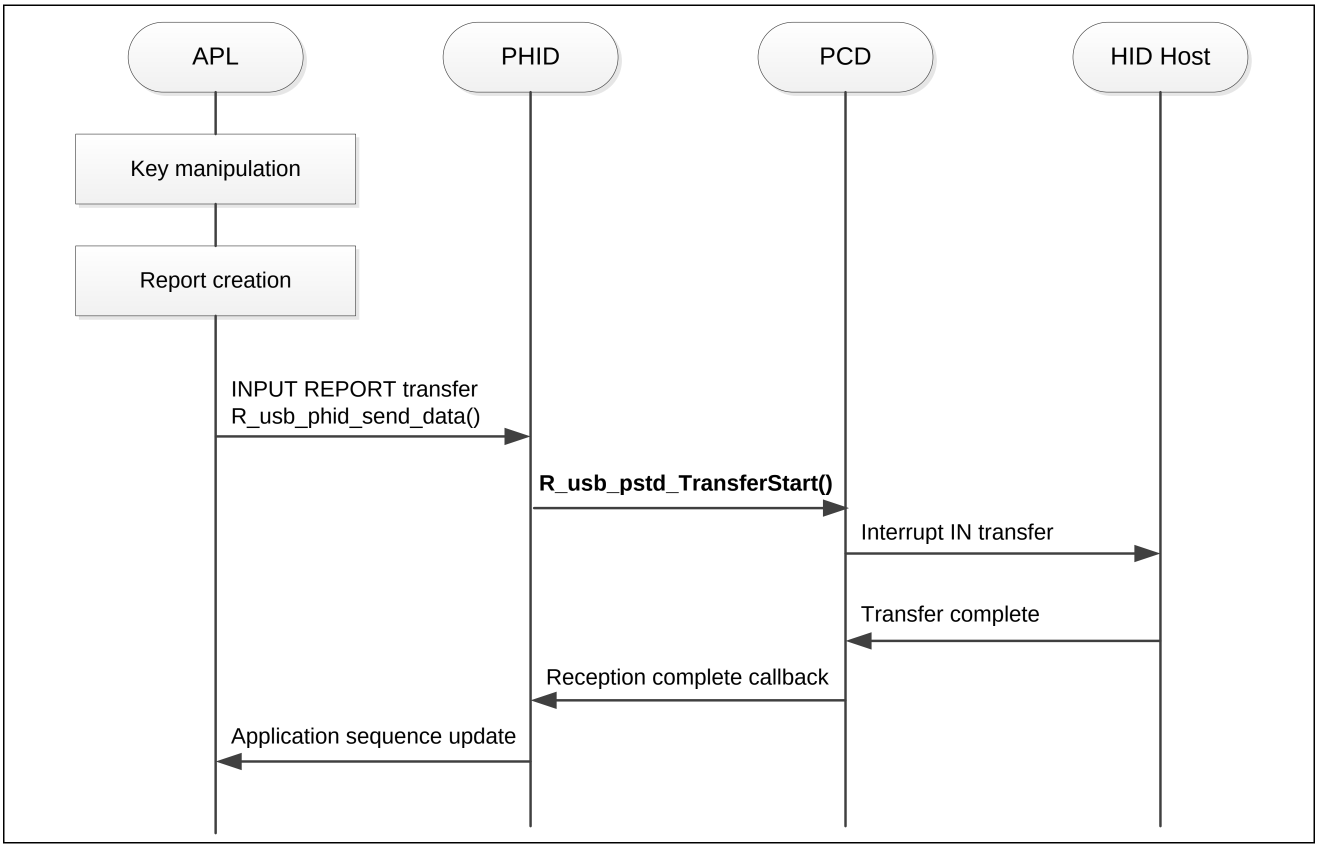


Figure 6-4 Report Transmission Sequence

6.6.3 Control Read Transfer

The figure below shows the communication sequence for control read transfers when a HID class request GetReport, GetIdle or GetProtocol (not supported) occurs.

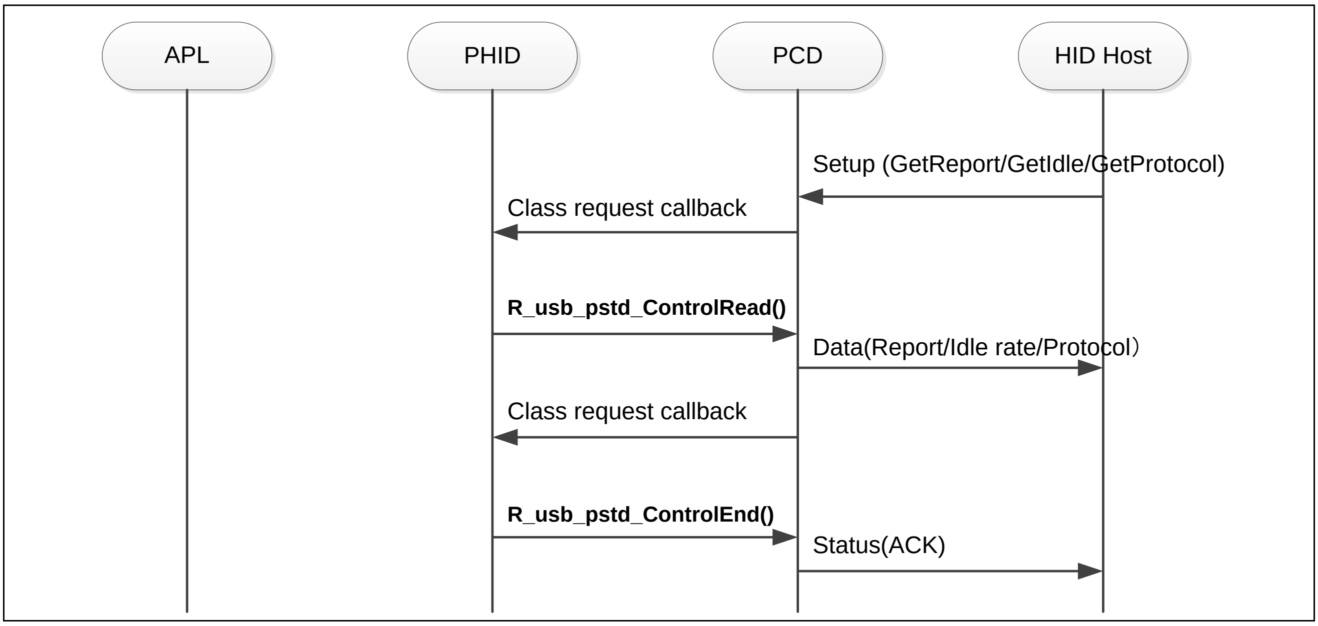


Figure 6-5 Control Read Sequence

6.6.4 Control Write Transfer

The figure below shows the communication sequence when a control write transfer occurs due to a SetReport class request.

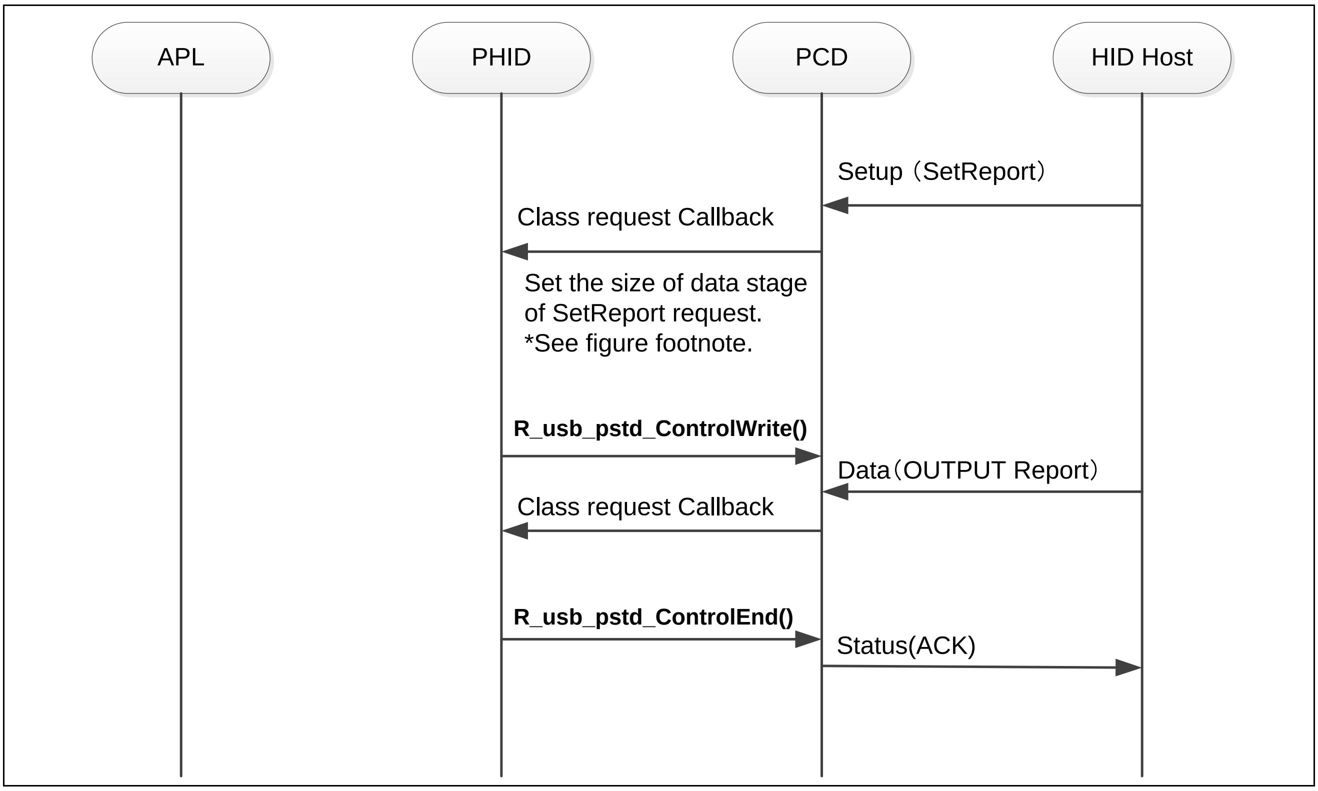


Figure 6-6 Control Write Sequence

[Note]

The size of the data stage of a SetReport request differs with PHID mode. The following shows the size for each.

Mouse mode	:	Does not receive SetReport requests
Keyboard mode	:	Keyboard LED control request: 1 byte (Refer to "Table 6-6 Keyboard OUTPUT Report format")

6.6.5 "No Data" Control Transfer

The figure below shows a control transfer communication sequence for a class request where there is no data-stage.

SetIdle and SetProtocol (Not supported) are "non-data-stage" control transfer HID class requests.

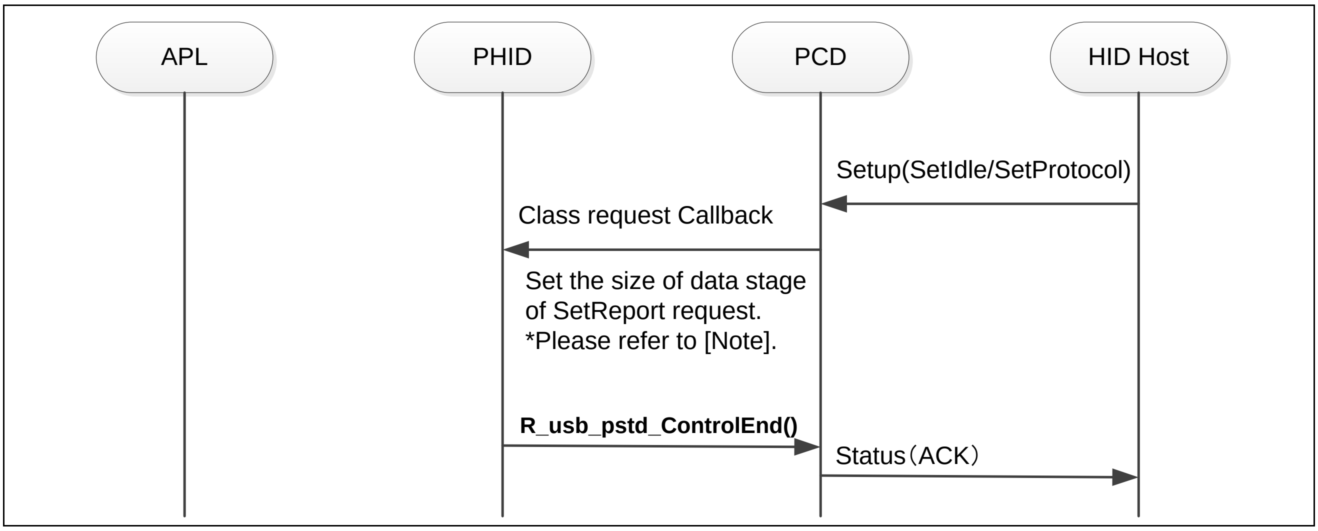


Figure 6-7 No data Control Sequence

7. Human Interface Devices (HID)

7.1 Basic Functions

The software presented here conforms to the HID Class specification

This software has the following main functions.

(1) Response to function references from the HID host
(2) Response to class requests from the HID host
(3) INPUT Report transfers to USB Host due to user input action on board.

7.2 HID Class Overview

7.2.1 Class Requests

The HID class requests (host to device) are listed below.

Table 7-1 HID Class Requests

Request	Code (bRequest)	Description	Supported
Get_Report	0x01	Receives a report from the HID host	Yes
Set_Report	0x09	Sends a report to the HID host	Yes
Get_Idle	0x02	Receives a duration (time) from the HID host	Yes
Set_Idle	0x0A	Sends a duration (time) to the HID host	Yes
Get_Protocol	0x03	Reads a protocol from the HID host	No*
Set_Protocol	0x0B	Sends a protocol to the HID host	No*
Get_Descriptor Descriptor Type : Class Class Descriptor Type : Report	0x06 (Standard)	Transmits a report descriptor	Yes
Get_Descriptor Descriptor Type : Class Class Descriptor Type : HID	0x06 (Standard)	Transmits an HID descriptor	Yes

* not supported (Stall response)

7.2.2 Class Request Data Format

The data format of the class requests supported by the class driver software is described below.

(1) GetReport

This is a class request for an Input or FeatureReport.

Table 7-2 GetReport Format

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

(2) SetReport

This is a class request to set an Output or Feature Report.

Table 7-3 SetReport Format

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

(3) GetIdle

This is a class request for the Idle rate from a HID device.

Table 7-4 GetIdle Format

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

(4) SetIdle

This is a class request to set the Idle rate for a HID device.

Table 7-5 SetIdle Format

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

(5) GetProtocol

This is a class request for the protocol setting of a HID device.

Table 7-6 GetProtocol Format

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

(6) SetProtocol

This is a class request to set the protocol of a HID device.

Table 7-7 SetProtocol Format

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

8. Pipe Specification

Pipe specifications differ depending on PHID demo mode (Mouse, Keyboard, or HID Demo).

Pipe specifications for each operating mode are specified in the Pipe Information Table in file "*r_usb_phid_descriptor.c*", and are as follows.

Table 8-1 Mouse Mode

Pipe Number	bEndpointAddress		bmAttributes	wMaxPacketSize	Description
	EP Number	Direction	Transfer Type	Max. Packet Size	
PIPE0	EP0	In/Out	Control	8/64	Standard request or class request
PIPE6	EP6	In	Interrupt	3	Data transfer from device to host

Table 8-2 Keyboard Mode

Pipe Number	bEndpointAddress		bmAttributes	wMaxPacketSize	Description
	EP Number	Direction	Transfer Type	Max. Packet Size	
PIPE0	EP0	In/Out	Control	8/64	Standard request or class request
PIPE6	EP6	In	Interrupt	8	Data transfer from device to host

Table 8-3 Demo Mode

Pipe Number	bEndpointAddress		bmAttributes	wMaxPacketSize	Description
	EP Number	Direction	Transfer Type	Max. Packet Size	
PIPE0	EP0	In/Out	Control	8/64	Standard request or class request
PIPE6	EP6	In	Interrupt	16	Data transfer from device to host

[Notes]

The Max Packet Size of PIPE0 is "8 bytes", when the PHID is a LS device. PHID uses PIPE6 for INPUT Report transfers. To use a pipe other than PIPE6 (PIPE7 through 9 can also be used for interrupt type) change the value of "USB_PHID_USE_PIPE".

Example using Pipe 7

File "*../Workspace/PHID/inc/r_usb_phid_define.h*"

```
//#define USB_PHID_USE_PIPE        USB_PIPE6
#define USB_PHID_USE_PIPE          USB_PIPE7
```

9. PHID Device Class Driver

9.1 Basic Functions

The PHID provides the following basic functions.

(1) Executes data transmission/reception with a HID host.
(2) Responds to HID host class requests

9.2 API

Table 9-1 below shows the PHID API (Application Programming Interface) functions.

Table 9-1 API Functions

Function Name	Description
R_usb_phid_send_data	Send a data transmit request message to the PHID task.
R_usb_phid_DeviceInformation	Gets the device's USB state information.
R_usb_phid_ChangeDeviceState	Changes the device's USB state.
R_usb_phid_driver_start	PHID driver start processing messages.
R_usb_phid_TransferEnd	USB data transfer termination request to the HCD.
R_usb_phid_transfer_length	Gets the report length depending on PHID mode.
R_usb_phid_control_callback	Control transfer processing for HID host requests.
R_usb_phid_task	The PHID task.

9.2.1 API

R_usb_phid_send_data

Send USB Data to Host

Format

```
void    R_usb_phid_send_data (  uint8_t*      Table,
                                usb_leng_t    size,
                                usb_cb_t      complete)
```

Arguments

*Table	Pointer to buffer containing data to transmit
size	Transfer size
complete	Transmit completion callback function

Return Value

—

Description

This function transfers the specified USB data of the specified size from the address specified by the transmit data address "Table".

When the transmission is done, the callback function "complete" is called.

Note

1. The USB transmit process results are found via the *usb_utr_t* pointer in the callback function’s arguments.
2. See "USB Communication Structure" (*usb_utr_t*) in the USB Basic Mini Firmware application note.

Example

```
void usb_apl_task( void )
{
  uint8_t    send_data[] = {0x01,0x02,0x03,0x04,0x05};  /* USB send data */
  uint16_t   size = 5;                                  /* Data size */

  R_usb_phid_send_data((uint8_t *)send_data, size, (USB_CB_t)&usb_complete)
}

/* Callback function */
void  usb_complete( usb_utr_t *mess );
{
  /* Processing at the time of the completion of USB transmitting */
}
```

R_usb_phid_DeviceInformation

Gets PHID Device State

Format

void R_usb_phid_DeviceInformation(uint16_t *deviceinfo)

Argument

*deviceinfo Pointer to the table address for device information storage

Return Value

— —

Description

This function gets the given USB peripheral's device information. It stores the information at the address designated by the argument "deviceinfo".

[0] : USB device state

b15-b8	Not used	
b7	VBSTS	VBUS Input Status 0 : USBm_VBUS pin is low. 1 : USBm_VBUS pin is high.
b6-b4	DVSQ[2:0]	Device State 0 0 0 : Powered state 0 0 1 : Default state 0 1 0 : Address state 0 1 1 : Configured state 1 x x : Suspended state x : Don't care
b3-b0	Not used	

[1] : USB transfer speed

0x0000 : Not connected
0x00C0 : Hi-Speed connected(Not supported)
0x0080 : Full-Speed connected
0x0040 : Low-Speed connected

[2] : Configuration number used
[3] : Interface number used
[4] : Remote Wakeup Flag (0: Wakeup control disable,1: Wakeup control enable)

Notes

1. Call this function from the user application or class driver.

Example

```
void  usb_smp_task(void)
{
  uint16_t res[5];
    :
  /* Get USB Device Information */
  R_usb_pstd_DeviceInformation((uint16_t *)res);
    :
}
```

R_usb_phid_ChangeDeviceState

Changes PHID Device State

Format

void R_usb_phid_ChangeDeviceState(uint16_t msginfo)

Argument

msginfo USB communication status

Return Value

— —

Description

This function sends the following request. to PCD.

Valid "msginfo" values

Msginfo	Description
USB_DO_REMOTEWAKEUP	Issues remote wakeup execution request to PCD

Notes

1. Call this function from the user application or the class driver.

Example

```
void  usb_smp_task( void )
{
    :
  /* Change the device state request */
  R_usb_phid_ChangeDeviceState(USB_DO_REMOTEWAKEUP);
    :
}
```

9.2.2 Common API

R_usb_phid_driver_start

Start PHID Driver

Format

void R_usb_phid_driver_start(void)

Argument

— —

Return Value

— —

Description

This function starts the PHID task and initialize the variables.

Note

1. Call this function from the user application during initialization.

Example

```
void usb_pstd_task_start( void )
{
    :
  usb_phid_driver_registration();   /*Peripheral Application Registration*/
  usb_papl_task_start();      /*Peripheral Application Task Start Setting*/
  R_usb_phid_driver_start(); /*Peripheral Class Driver Task Start Setting*/
  R_usb_pstd_usbdriver_start();   /* Peripheral USB Driver Start Setting */
    :
}
```

R_usb_phid_TransferEnd

Terminate USB Data Transfer

Format

USB_ER_t R_usb_phid_TransferEnd(void)

Argument

— —

Return Value

— —

Description

This function forces data transfer via the pipes to end.

The function executes a data transfer forced end request to PCD. After receiving the request, PCD executes the data transfer forced end request processing.

When a data transfer is forcibly ended, the function calls the callback function, set in R_usb_phid_send_data, at the time the data transfer was requested. The remaining data length of transmission and reception, status, the number of times of a transmission error, and the information on forced termination are set to the argument (mess) of this callback function

Note

1. Call this function from the user application program or class driver.

Example

```
 void  usb_smp_task(void)
 {
     :
   /* Transfer end request */
   R_usb_phid_TransferEnd();
     :
}
```

R_usb_phid_transfer_length

Get Report Size

Format

uint16_t R_usb_phid_transfer_length(void)

Argument

— —

Return Value

— INPUT Report size by the PHID mode.

Description

This function gets the INPUT Report size in Bytes of the PHID mode.

- Mouse mode : 3 Byte
- Keyboard mode : 8 Byte
- HID demo mode : 5 Byte

Note

Call this function from the user application program or class driver.

Example

```
void usb_smp_task( void )
{
  uint16_t   usb_smp_report_length;
    :
  usb_smp_report_length = R_usb_phid_transfer_length();
    :
}
```

R_usb_phid_control_callback

PHID Control Transfer Processing

Format

void R_usb_phid_control_callback(usb_request_t *request, uint16_t ctsq)

Argument

*request	Pointer to a class request message.		
ctsq	Control transfer stage information		
		USB_CS_IDST	Idle or setup stage
		USB_CS_RDDS	Control read data stage
		USB_CS_WRDS	Control write data stage
		USB_CS_WRND	Control write no data status stage
		USB_CS_RDSS	Control read status stage
		USB_CS_WRSS	Control write status stage
		USB_CS_SQER	Sequence error

Return Value

— —

Description

For HID class requests, this function calls processing for the control transfer stage.

This is a callback function registered earlier during PHID "driver registration". It is called at the time of a host HID control transfer.

Note

—

Example

```
void usb_apl_task( void )
{
  usb_pcdreg_t  driver;

        :
  /* Control Transfer */
  driver.ctrltrans = R_usb_phid_control_callback;
  R_usb_pstd_DriverRegistration(&driver);
        :
}
```

R_usb_phid_Task

The PHID Task

Format

void R_usb_phid_task(void)

Argument

— —

Return Value

— —

Description

This is the PHID task which processes messages from the application and notifies the application *usb_phid_main_task* of the results using messages.

Note

In non-OS operation, the function is registered with the scheduler.

Refer to the USB-BASIC-FW Application Notes for more information concerning the scheduling process.

Example

```
void usb_apl_task_switch(void)
{
  while( 1 )
  {
   if( USB_FLGSET == R_usb_cstd_Schedule() )
   {
      /* PCD Task */
      R_usb_pstd_PcdTask();

       /* Peripheral HID Task */
      R_usb_phid_task();

      /* Peripheral HID demo sample application Task */
       usb_phid_main_task();
   }
  }
}
```

9.3 Pipe (Endpoint) Information Table

It is necessary to create a descriptor table, and to edit the Pipe Information Table, which are both used by PCD. Refer to the sample in file "*r_usb_phid_descriptor.c*". For details, see the USB-BASIC-FW application note.

No.	Table Name	Description	Remarks
1	usb_gphid_EpTbl	Pipe Information Table (Endpoint table): Define endpoints used, and the pipes used for the target.	Different table used depending on PHID mode.
2	usb_gphid_SmplDeviceDescriptor	Device Descriptor	—
3	usb_gphid_Configuration	Configuration Descriptor	Different data table by the PHID mode. *1
4	usb_gphid_StringDescriptor0	String Descriptor 0	Language ID
5	usb_gphid_StringDescriptor1	String Descriptor 1	Manufacturer name (iManufacturer)
6	usb_gphid_StringDescriptor2	String Descriptor 2	Product Name (iProduct)
7	usb_gphid_StringDescriptor3	String Descriptor 3	Serial Number (iSerialNumber)
8	usb_gphid_StringDescriptor4	String Descriptor 4	Configuration name (HID demo)
9	usb_gphid_StringDescriptor5	String Descriptor 5	Configuration name (Keyboard)
10	usb_gphid_StringDescriptor6	String Descriptor 6	Configuration name (Mouse)
11	usb_gphid_StrPtr	String Descriptor pointer : String Descriptor address array.	
12	usb_gphid_ReportDiscriptor	Report Descriptor : Define the Report data format.	Data table used depending on PHID mode. *2

*1. The PHID configuration descriptor varies by the device used (PHID mode). The mode is chosen in the header file *r_usb_phid_usrcfg.h*. (The configuration descriptor includes the interface, HID and endpoint descriptors.)

The bInterface protocol of the interface descriptor depends on the PHID mode:

PHID mode	Protocol code	Product ID	Configuration num	Max packet size	Descriptor length
HID demo	0 (None)	0x0003	4	3	34
Keyboard	1 (Keyboard)	0x0013	5	5	63
Mouse	2 (Mouse)	0x0023	6	8	50

*2. The report descriptor value for Keyboard mode and Mouse mode are shown in"E.6 Report Descriptor (Keyboard)" and "E.10 Report Descriptor (Mouse)" chapter of "USB Device Class Definition for Human Interface Devices (HID) 1.11".

10. Setup for the e² studio project

(1). Start up e² studio.

* If starting up e² studio for the first time, the Workspace Launcher dialog box will appear first. Specify the folder which will store the project.

(2). Select [File] → [Import]; the import dialog box will appear.

(3). In the Import dialog box, select [Existing Projects into Workspace].

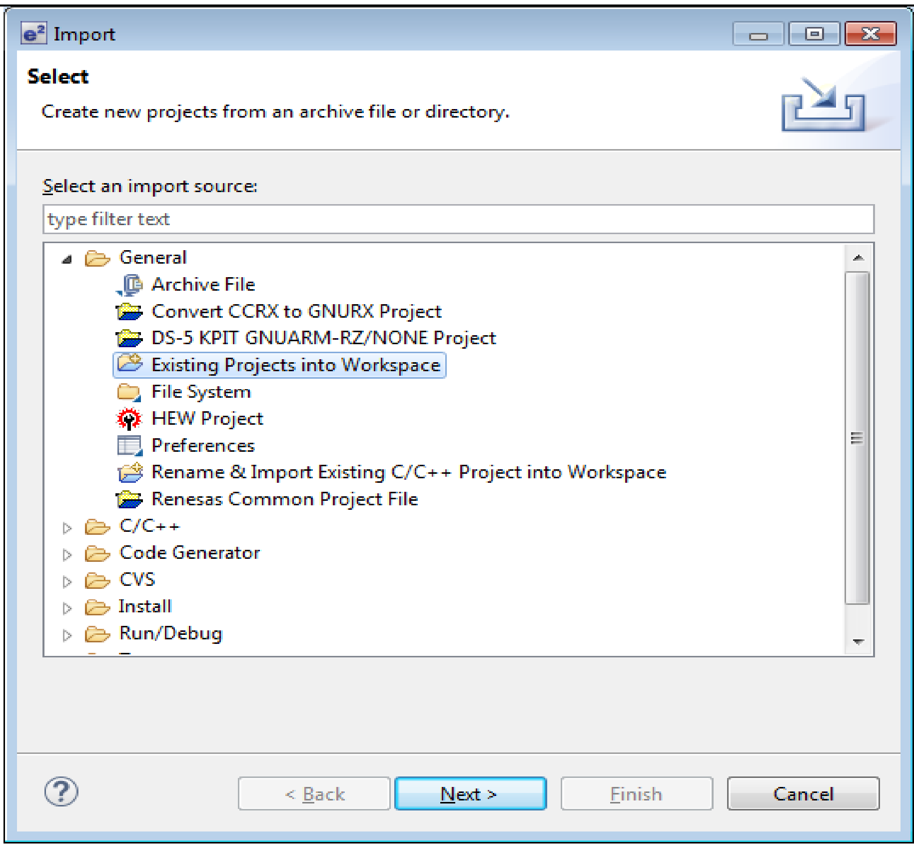


Figure 10-1 Select Import Source

(4). Press [Browse] for [Select root directory]. Select the folder in which [.cproject] (project file) is stored.

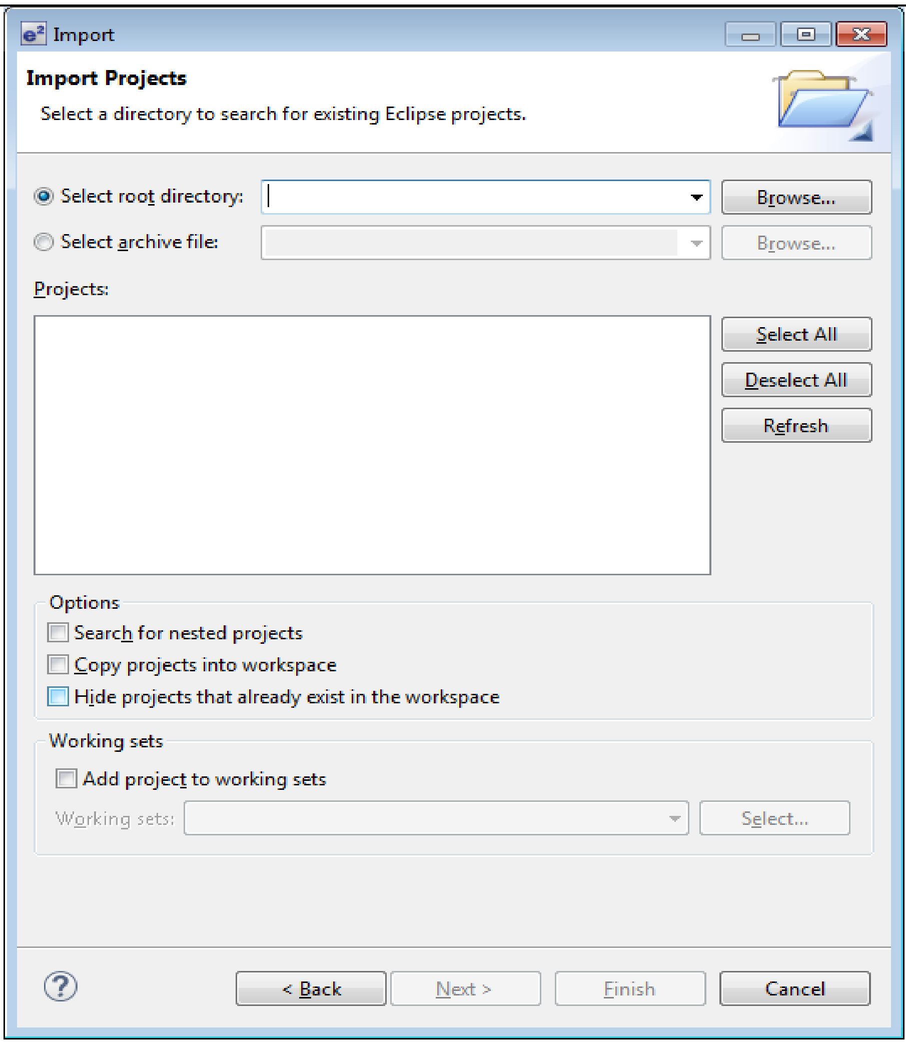


Figure 10-2 Project Import Dialog Box

(5). Click [Finish].

This completes the step for importing a project to the project workspace.

11. Using the e² studio project with CS+

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

Note:

The *rcpc* file is stored in "workspace\RL78\CCRL*devicename*" folder.

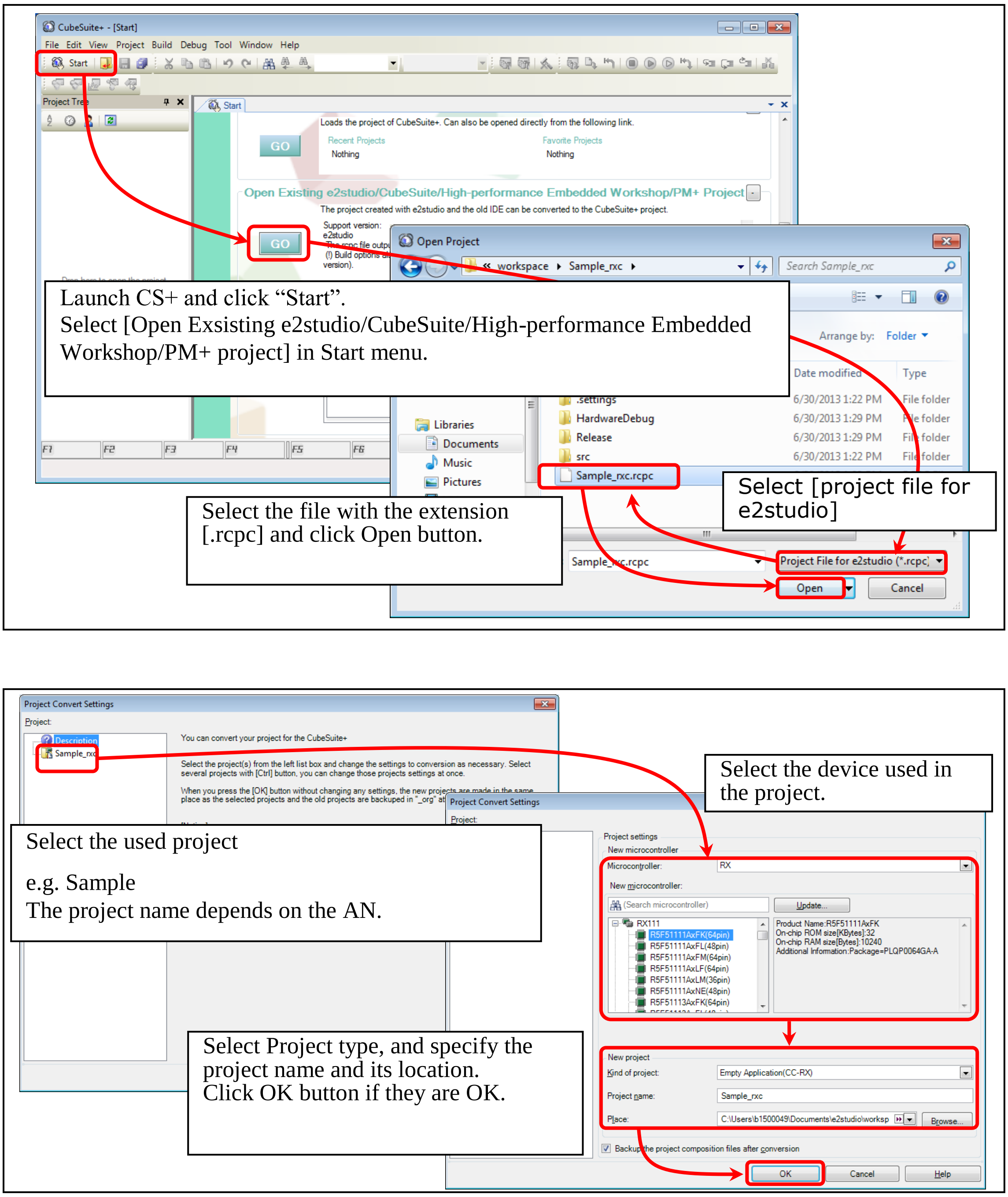


Figure 11-1 Using the e² studio project with CS+

12. Limitations

USB Peripheral Human Interface Devices Class Driver (PHID) is subject to the following restrictions.

Low power mode is not supported.

Keyboard mode is not supported for multi key pressed and special key (Control key, Shift key).

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	May.12.11	—	First edition issued
2.00	Jan.24.13	—	Revision of the document by firmware upgrade
2.10	Aug 1.13	—	RL78/L1C, RX111 is added as new supported device.
2.11	Oct 31.13	—	1.4 Folder path fixed. 3.3.1 Folder Structure was corrected.
2.12	Mar 31.14	—	R8C is added as new supported device.
2.13	Mar 16.15	—	RX111, R8C/3MU and R8C/3MK are deleted from Target Device.
2.14	Jan 18. 16	—	Supported Technical Update (Document No. TN-RL*-A055A/E and TN-RL*-A033B/E)
2.15	Mar 28. 16	—	CC-RL compiler is supported.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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 an MPU or MCU 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

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 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.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, P. R. China 200333
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 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0