Introduction
This application note describes USB Peripheral Human Interface Devices Class Driver (PHID). This driver operates in combination with the USB Basic Peripheral Driver (USB-BASIC-F/W). It is referred to below as the PHID.

Target Device
RX62N/RX621 Group
RX63N/RX631 Group
RX630 Group
RX63T 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
5. RX63T Group User's Manual: Hardware (Document number .R01UH0238)
6. USB Basic Host and Peripheral Driver Application Note (Document number. R01AN0512)

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

USB Devices Page
http://www.renesas.com/prod/usb/
Contents

1. Overview ........................................................................................................................................ 3
2. Software Configuration .................................................................................................................. 4
3. API Information ............................................................................................................................ 5
4. USB Peripheral Human Interface Devices Class Driver (PHID) ...................................................... 7
5. API Functions .................................................................................................................................. 9
6. Configuration (r_usb_phid_config.h) ............................................................................................ 10
7. Sample Application ......................................................................................................................... 11
8. Setup ............................................................................................................................................... 18
9. Creating an Application .................................................................................................................. 21
10. Using the e² studio project with CS+ ........................................................................................... 22
1. **Overview**

PHID, when used in combination with the USB-BASIC-F/W, operates as a USB peripheral human interface device class driver (PHID). The PHID conforms to the USB Human Interface Device class specifications (referred to here as HID) and implements communication with a HID host.

This module supports the following functions.

- Data transfer to and from a USB host
- Response to HID class requests
- Response to function references from the HID host
- Interrupt OUT transfer

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 (*r01an0401jj0232_usb.pdf, r01an0512jj0232_usb.pdf*) under the "reference_documents" folder when using RX62N/RX621/RX63T/RX630.

1.3 **Terms and Abbreviations**

- API : Application Program Interface
- APL : Application program
- HID : Human Interface Device class
- 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
- RSK : Renesas Starter Kit
- USB : Universal Serial Bus

: USB Basic Peripheral Driver
2. Software Configuration

Figure 2-1 shows the configuration of the modules related to PHID

![Diagram](image)

**Figure 2-1 Source Code Block Diagram**

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHID</td>
<td>User switch operation on the RSK board is converted into HID reports.</td>
</tr>
<tr>
<td></td>
<td>The transfer result is notified to APL by the callback function.</td>
</tr>
<tr>
<td></td>
<td>In addition, communicate the output report of HID host to APL.</td>
</tr>
<tr>
<td>PCD</td>
<td>Peripheral Control Driver (USB Basic Driver)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>C compiler</td>
<td>Renesas Electronics C/C++ compiler for RX Family V.3.01.00</td>
</tr>
<tr>
<td></td>
<td>Compile Option : -lang = c99</td>
</tr>
<tr>
<td>Endian</td>
<td>Little Endian, Big Endian</td>
</tr>
<tr>
<td>Using Board</td>
<td>Renesas Starter Kits for RX63N</td>
</tr>
<tr>
<td>Host Environment</td>
<td>The operation of this USB Driver module connected to the following OSes has been confirmed.</td>
</tr>
<tr>
<td></td>
<td>1. Windows® 7</td>
</tr>
<tr>
<td></td>
<td>2. Windows® 8.1</td>
</tr>
<tr>
<td></td>
<td>3. Windows® 10</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX63N/RX631</td>
<td>USB10 Interrupt (Vector number: 35) / USBR0 Interrupt (Vector number: 90)</td>
</tr>
<tr>
<td></td>
<td>USB11 Interrupt (Vector number: 38) / USBR0 Interrupt (Vector number: 91)</td>
</tr>
</tbody>
</table>

3.4 Header Files

All API calls and their supporting interface definitions are located in r_usb_basic_if.h and r_usb_phid_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 6, Configuration (r_usb_phid_config.h) in this document and 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.

<p>| Checks arguments | Does not check arguments |</p>
<table>
<thead>
<tr>
<th></th>
<th>ROM size</th>
<th>RAM size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>17.5K bytes (Note 3)</td>
<td>17.0K bytes (Note 4)</td>
</tr>
<tr>
<td></td>
<td>9.0K bytes</td>
<td>9.0K bytes</td>
</tr>
</tbody>
</table>

**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. USB Peripheral Human Interface Devices Class Driver (PHID)

4.1 Class Requests (Host to Peripheral)

This driver notifies to the application program when receiving the following class request.

For the class request processing, refer to chapter "USB Class Requests" in the document (Document number: R01AN0512) for USB Basic Host and Peripheral Driver Application Note.

<table>
<thead>
<tr>
<th>Request</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get_Report</td>
<td>0x01</td>
<td>Receives a report from the HID host</td>
</tr>
<tr>
<td>Set_Report</td>
<td>0x09</td>
<td>Sends a report to the HID host</td>
</tr>
<tr>
<td>Get_Idle</td>
<td>0x02</td>
<td>Receives a duration (time) from the HID host</td>
</tr>
<tr>
<td>Set_Idle</td>
<td>0x0A</td>
<td>Sends a duration (time) to the HID host</td>
</tr>
<tr>
<td>Get_Protocol</td>
<td>0x03</td>
<td>Reads a protocol from the HID host</td>
</tr>
<tr>
<td>Set_Protocol</td>
<td>0x0B</td>
<td>Sends a protocol to the HID host</td>
</tr>
<tr>
<td>Get_Descriptor</td>
<td>0x06</td>
<td>Transmits a report descriptor (Standard)</td>
</tr>
<tr>
<td>Descriptor Type : Class Descriptor Type : Report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get_Descriptor</td>
<td>0x06</td>
<td>Transmits an HID descriptor (Standard)</td>
</tr>
<tr>
<td>Descriptor Type : Class Descriptor Type : HID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.2 Class Request Data Format

1. GetReport

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA1</td>
<td>GET_REPORT (0x01)</td>
<td>ReportType &amp; ReportID</td>
<td>Interface</td>
<td>ReportLength</td>
<td>Report</td>
</tr>
</tbody>
</table>

2. SetReport

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x21</td>
<td>SET_REPORT (0x09)</td>
<td>ReportType &amp; ReportID</td>
<td>Interface</td>
<td>ReportLength</td>
<td>Report</td>
</tr>
</tbody>
</table>

3. GetIdle

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA1</td>
<td>GET_IDLE (0x02)</td>
<td>0(Zero) &amp; ReportID</td>
<td>Interface</td>
<td>1(one)</td>
<td>Idle rate</td>
</tr>
</tbody>
</table>

4. SetIdle

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x21</td>
<td>SET_IDLE (0x0A)</td>
<td>Duration &amp; ReportID</td>
<td>Interface</td>
<td>0(zero)</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

5. GetProtocol

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA1</td>
<td>GET_PROTOCOL (0x03)</td>
<td>0(zero)</td>
<td>Interface</td>
<td>0(zero)</td>
<td>0 (Boot Protocol) / 1 (Report Protocol)</td>
</tr>
</tbody>
</table>

6. SetProtocol

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x21</td>
<td>SET_PROTOCOL (0x0B)</td>
<td>0 (Boot Protocol) / 1 (Report Protocol)</td>
<td>Interface</td>
<td>0(zero)</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
5. API Functions

For API used in the application program, refer to chapter "API Functions" in the document (Document number: R01AN0512) for USB Basic Host and Peripheral Driver Application Note.
6. Configuration (r_usb_phid_config.h)

Please set the following according to your system.

Note:

Be sure to set r_usb_basic_config.h file as well. For r_usb_basic_config.h file, refer to chapter "Configuration" in the document (Document number: R01AN0512) for USB Basic Host and Peripheral Driver Application Note.

1. Setting pipe to be used

Set the pipe number (PIPE6 to PIPE9) to use for Interrupt IN/OUT transfer. Do not set the same pipe number for the definitions of USB_CFG_PHID_INT_IN and USB_CFG_PHID_INT_OUT.

```c
#define USB_CFG_PHID_INT_IN Pipe number (USB_PIPE6 to USB_PIPE9)
#define USB_CFG_PHID_INT_OUT Pipe number (USB_PIPE6 to USB_PIPE9)
```

Note:

For a system that does not support the OUT transfer, set USB_NULL as the definition of USB_CFG_PHID_INT_OUT.
7. Sample Application

7.1 Application Specifications

The main functionalities of the PHID sample application (APL) are described below.

1. **Keyboard Mode: Keyboard Functionality**
   When the RSK connects to the USB host, the USB host recognizes the RSK as a keyboard. The RSK operates as a keyboard, transmitting keyboard data to the USB host using interrupt IN transfer.

2. **Echo Mode: USB Loopback Functionality (Interrupt IN/OUT Data Transfer)**
   The RSK connects to the USB host and performs interrupt IN/OUT data transfer. This functionality performs processing to transmit the data received from the USB host back to the USB host unaltered.

3. **Low-Power-Consumption Functionality**
   This functionality transitions the microcontroller to a low-power mode according to the USB state.
   
   a) In the USB suspend state, the microcontroller transitions to sleep mode.
   
   b) In the USB detached state, the microcontroller transitions to software standby mode.

[Note]
1. Make the selection of Keyboard mode or Echo mode by setting in r_usb_phid_apl_config.h.
2. Make the selection(using or unusing) of the low power consumption functionality in r_usb_phid_apl_config.h
3. Echo mode enables communication with USB hosts supporting USB loopback functionaly. Keyboard mode enable USB communication with PCs (USB hosts) supporting OSes such as Windows 7, Windows 8.1 and Windows 10.
7.2 Application Processing

The APL consists of two parts: processing of initial settings and the main loop. These are described in outline below.

7.2.1 Initial Setting

The initial settings include microcontroller pin settings, USB driver settings, and USB controller initial settings.

7.2.2 Main Loop (Keyboard mode)

In keyboard mode, processing is performed to send information about the switches(buttons) on the RSK board to the USB host. When the RSK (HID device) is connected to the USB host (PC), the RSK is recognized as a keyboard, and the main loop consists of a main routine that sends the switch(button) information to the USB host as key input data. For details of the switch(button) information (key input data), refer to 7.4.1, Switch specification.

1. When enumeration with the USB host completes, the USB host sends an HID class request to the HID device. After it receives the HID class request, the HID device calls the `R_USB_GetEvent` function and the return value is set to `USB_STS_REQUEST`. When the APL confirms `USB_STS_REQUEST`, it analyzes the received class request and performs processing corresponding to it.

2. When the `R_USB_GetEvent` function is called after processing of the class request mentioned in 1, above, completes, the return value is set to `USB_STS_REQUEST_COMPLETE`. The APL performs processing to make request information settings, etc.

3. The APL confirms whether a switch(button) on the RSK was depressed or not. If a switch(button) is depressed, the APL references the status management variable to confirm whether or not data transmission is in progress. If data transmission is not in progress, it calls the `R_USB_Write` function to send information on the depressed switch(button) as key information.

4. When the `R_USB_GetEvent` function is called after transmission of HID data as mentioned in 3, above, completes, the return value is set to `USB_STS_WRITE_COMPLETE`. When the APL confirms `USB_STS_WRITE_COMPLETE`, it calls the `R_USB_Write` function to request transmission of zero data (8 bytes). (In keyboard mode it is necessary to send zero data to inform the USB host that the key input has been released.

5. When the `R_USB_GetEvent` function is called after transmission of zero data as mentioned in 4, above, completes, the return value is set to `USB_STS_WRITE_COMPLETE`. When the APL confirms `USB_STS_WRITE_COMPLETE`, it checks whether or not transmission of zero data has completed, and if it is zero data, the APL makes a setting (NO_WRITING) to the status management variable.

6. If a suspend signal is received from the USB host or a detach event is confirmed while the processing in steps 1, to 5, above is repeating, the APL performs processing to transition the HID device (RSK) to low-power mode. For information on low-power mode, refer to 7.2.4, MCU Low power consumption processing. Note that confirmation of reception of a suspend signal or a detach event involves reading the return value (USB_STS_SUSPEND or USB_STS_DETACH) of the `R_USB_GetEvent` function. An outline of the processing of the APL is shown below.
Figure 7-1  Main Loop Processing (Keyboard mode)
7.2.3 Main loop (Echo mode)

The echo mode main loop performs loop-back processing in which data received from the USB host is transmitted unaltered back to the USB host as part of the main routine. An overview of the processing of the main loop is presented below.

1. When the `R_USB_GetEvent` function is called after enumeration with the USB host completes, `USB_STS_CONFIGURED` is set as the return value. When the APL confirms `USB_STS_CONFIGURED`, it calls the `R_USB_Read` function to make a data receive request for data sent by the USB host.

2. When the `R_USB_GetEvent` function is called after reception of data from the USB host has completed, `USB_STS_READ_COMPLETE` is set as the return value. When the APL confirms `USB_STS_READ_COMPLETE`, it calls the `R_USB_Write` function to transmit the received data to the USB host.

3. When the `R_USB_GetEvent` function is called after transmission of data to the USB host completes, `USB_STS_WRITE_COMPLETE` is set as the return value. When the APL confirms `USB_STS_WRITE_COMPLETE`, it calls the `R_USB_Read` function to make a data receive request for data sent by the USB host.

4. The processing in steps 2 and 3, above, is repeated.

5. When it confirms reception of a suspend signal from the USB host or DETACH, the APL performs processing to transition the CDC device (RSK) to low-power mode. For information on low-power mode, refer to 7.2.4, MCU Low power consumption processing. Note that confirmation of reception of a suspend signal or DETACH is performed by referencing the return value (`USB_STS_SUSPEND` or `USB_STS_DETACH`) of the `R_USB_GetEvent` function.

An outline of the processing of the APL is shown below.

![Flowchart](image-url)
7.2.4  MCU Low power consumption processing

MCU low-power processing occurs when the conditions in Table 7-1 are met, causing a transition to low-power mode. Note that this processing is enabled by setting the USE_LPW definition in the r_usb_phid_apl_config.h file to USE_LPW.

### Table 7-1 Conditions for Transition to Low-Power Mode

<table>
<thead>
<tr>
<th>Transition Condition</th>
<th>Transition Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBUS OFF</td>
<td>Software standby mode</td>
</tr>
<tr>
<td>VBUS ON Suspend</td>
<td>Sleep mode</td>
</tr>
<tr>
<td>VBUS ON Other</td>
<td>Normal mode (program running)</td>
</tr>
</tbody>
</table>

1. When the HID device (RSK) detaches from the USB host (VBUS OFF), the APL performs processing to transition the MCU to software standby mode. Recovery from software standby mode occurs when the HID device (RSK) attaches to the USB host.

2. When a suspend signal sent by the USB host is received while the HID device (RSK) is connected to the USB host, the APL performs processing to transition the MCU to sleep mode. Note that recovery from sleep mode occurs when a resume signal is received from the USB host.

![Flowchart of MCU Low Power Consumption Processing](image-url)

**Figure 7-3  Flowchart of MCU Low Power Consumption Processsing**
7.3  Configuration File for the application program (r_usb_phid_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. Specify one of the following settings for the USE_USBIP definition.

   ```
   #define USE_USBIP USE_USBIP0  // Specify USB_IP0.
   #define USE_USBIP USE_USBIP1  // Specify USB_IP1.
   ```

   **[Note]**

   Specify USE_USBIP0 when using RX63T or RX630.

2.  **OPERATION_MODE Definition**

   Specify one of the following settings for the OPERATION_MODE definition.

   ```
   #define OPERATION_MODE HID_KEYBOARD  // Keyboard mode
   #define OPERATION_MODE HID_ECHO       // Echo mode
   ```

3.  **Low-Power Function Definition**

   Specify whether or not the low-power function will be used. If the low-power function will be used, specify USB_APL_ENABLE to USB_SUPPORT_LPW definition.

   ```
   #define USE_SUPPORT_LPW USB_APL_DISABLE  // No use the low-power function
   #define USE_SUPPORT_LPW USB_APL_ENABLE   // Use the low-power function
   ```

4.  **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 Basic Host and Peripheral Driver Application Note* (Document number. R01AN0512EJ).

7.4  **Keyboard operation**

In keyboard mode the switches(buttons) on the RSK board are used to make the RSK operate as an HID device. The switch(button) input information is used as keyboard key data.

7.4.1  **Switch specification**

The specifications of the switches used in keyboard mode are listed below. The switch input information is used as the key data of the keyboard.

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch2(SW2)</td>
<td>One of the key codes for characters &quot;a&quot; to &quot;z&quot; or &quot;Enter&quot; is reported to the host each time SW is pressed.</td>
</tr>
<tr>
<td>Switch3(SW3)</td>
<td>One of the key codes for &quot;1&quot; to &quot;9&quot; and &quot;0&quot; or &quot;Enter&quot; is notified to the host each time SW is pressed.</td>
</tr>
</tbody>
</table>
7.4.2 Data Format
The table below shows the transmit report format used for sending data to and from USB Host. These data formats are set in conjunction with the HID report descriptor contents notified to USB Host.

<table>
<thead>
<tr>
<th>offset</th>
<th>Keyboard Mode (8Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Modifier keys</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Keycode 1</td>
</tr>
<tr>
<td>3</td>
<td>Keycode 2</td>
</tr>
<tr>
<td>4</td>
<td>Keycode 3</td>
</tr>
<tr>
<td>5</td>
<td>Keycode 4</td>
</tr>
<tr>
<td>6</td>
<td>Keycode 5</td>
</tr>
<tr>
<td>7</td>
<td>Keycode 6</td>
</tr>
</tbody>
</table>

7.5 Descriptor
The PHID’s descriptor information is contained in r_usb_phid_descriptor.c. Also, please be sure to use your vendor ID.
8. Setup

8.1 Hardware

8.1.1 Example Operating Environment

Figure 8-1 shows an example operating environment for the PHID. Refer to the associated instruction manuals for details on setting up the evaluation board and using the emulator, etc.

![Diagram of Example Operating Environment]

**Figure 8-1 Example Operating Environment**

8.1.2 RSK Setting

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

<table>
<thead>
<tr>
<th>RSK</th>
<th>Jumper Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSK+RX63N</td>
<td>J3: Shorted Pin1-2</td>
</tr>
<tr>
<td></td>
<td>J4: Shorted Pin1-2</td>
</tr>
<tr>
<td></td>
<td>J18: Shorted Pin2-3</td>
</tr>
</tbody>
</table>

Table 8-1 RSK Setting

Note:

For the detail of RSK setting, refer to the user's manual of RSK.
8.2 Software

1) Setup e2 studio
   a) Start e2 studio
   b) If you start up e2 studio at first, the following dialog is displayed. Specify the folder to store the project in this dialog.

   ![Dialog for selecting workspace](image)

   - Use this as the default and do not ask again
   - OK
   - Cancel

2) Import the project to the workspace
   a) Select [File] > [Import]
   b) Select [General] => [Existing Projects into Workspace]

   ![Dialog for importing projects](image)
c) Select the root directory of the project, that is, the folder containing the “.cproject” file.

![Import Projects dialog box]

- Select root directory: 
- Select archive file: 

Projects: 

- Select All
- Deselect All
- Refresh


d) Click “Finish”.

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

3) Generate the binary target program by clicking the “Build” button.

4) Connect the target board to the debug tool and download the executable. The target is run by clicking the “Run” button.
9. 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.
10. Using the e² studio project with CS+

The PHID contains a project only for e² studio. When you use the PHID 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 10-1 Using the e² studio project with CS+

Select the used project
e.g. Sample
The project name depends on the AN.
Website and Support

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

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

All trademarks and registered trademarks are the property of their respective owners.
## Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Mar.09.11</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.10</td>
<td>Aug.10.11</td>
<td>—</td>
<td>Add Target Device RX630, R8A66597 &lt;br&gt;Add the information on RX630 and R8A66597 (Hi-Speed USB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>5. Peripheral HID Sample Application (APL) change to Application &lt;br&gt; - Unused SW1 &lt;br&gt; - SW description is changed to the function switch description</td>
</tr>
<tr>
<td>2.00</td>
<td>Sep.30.12</td>
<td>—</td>
<td>Revision of the document by firmware upgrade</td>
</tr>
<tr>
<td>2.10</td>
<td>Apr.1.13</td>
<td>—</td>
<td>First Release for V.2.10 &lt;br&gt;Add Target Device RX63T and R8A66593. Add the information on RX63T and R8A66593.</td>
</tr>
<tr>
<td>2.20</td>
<td>Sep.30.15</td>
<td>—</td>
<td>Change the application program. &lt;br&gt;Change the folder structure. &lt;br&gt;RX63N, RX631, R8A66597 and R8A66593 are deleted from Target Device.</td>
</tr>
<tr>
<td>2.30</td>
<td>Sep 30, 2016</td>
<td>—</td>
<td>Supporting USB Host and Peripheral Interface Driver application note(Document No.R01AN3293EJ)</td>
</tr>
<tr>
<td>2.31</td>
<td>Sep 30, 2017</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>2.32</td>
<td>Mar 31, 2018</td>
<td>—</td>
<td>The revision of USB Basic driver has been updated.</td>
</tr>
<tr>
<td>2.33</td>
<td>Jul 31, 2019</td>
<td>—</td>
<td>RX63N and RX631 are added in Target Device.</td>
</tr>
</tbody>
</table>
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.
SALES OFFICES

Renesas Electronics Corporation

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

Renesas Electronics Corporation
TOYOSU FORRESIA, 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: +1-408-435-5551

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

Renesas Electronics Europe GmbH
Amadinstraße 10, 40472 Düsseldorf, Germany
Tel: +49-211-6500-3, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 101-701, Floor 1, Building 7, Yard No. 7, 8th Street, 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: +86-21-2226-6888, 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 Taiwain 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 05-02 Hylux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-9300

Renesas Electronics Malaysia Sdn.Bhd.
Unit No 3A-1 Level 3A Tower 8 IONA Business Park, No 1 Jalan Punggurata U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-9222-1298, Fax: +60-3-9222-1299

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

Renesas Electronics Korea Co., Ltd.
17F., KAMCO Yangee Tower, 262, Gangnam-daero, Gangnam-gu, Seoul 06265 Korea
Tel: +82-2-538-3737, Fax: +82-2-538-3338