Renesas Synergy™ Platform

USBX™ Host Class HID Module Guide

Introduction

This module guide will enable you to effectively use a module in your own design. Upon completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application and write code, using the included application project code as a reference and efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are available in the Renesas Synergy™ Knowledge Base (as described in the References section at the end of this document) and should be valuable resources for creating more complex designs.

The USBX™ Host Class HID module is a high-level API for human interface device (HID) applications and is implemented on `g_ux_host_class_hid`. The USBX Host Class HID module configures the USBX Host Class HID Source, USBX Host Configuration, USBX Source and USBX Port HCD. The USBX Host Class HID module uses the USB peripheral on the Synergy MCU.

Contents

1. USBX Host Class HID Module Features

The USBX Host Class Human Interface Device (HID) module supports the USBX HID class. It provides the following features:

- Supports HID report data transfers
- Supports the following clients:
  - Keyboard
  - Mouse
  - Remote-Control

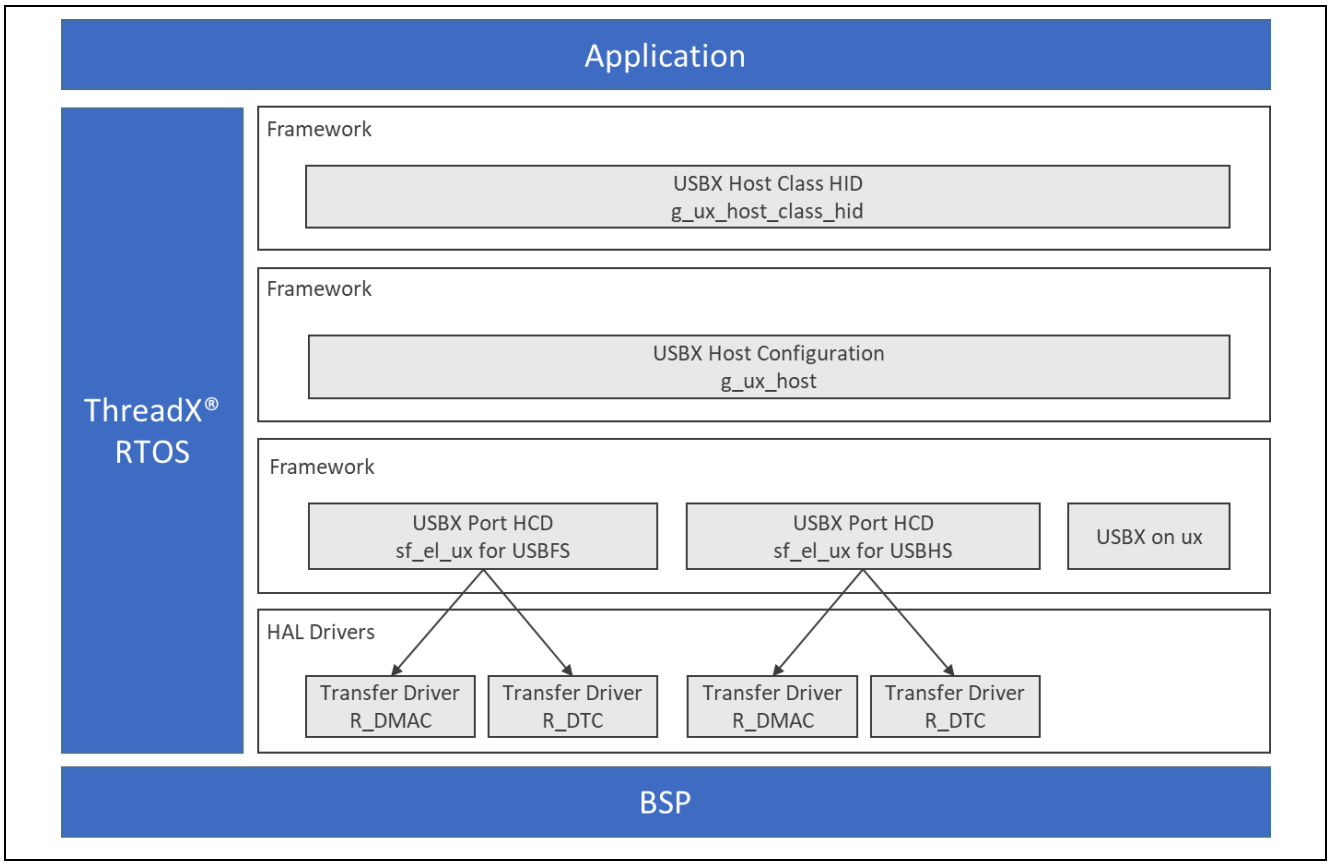

Figure 1. USBX Host Class HID Module Block Diagram

2. USBX Host Class HID Module APIs Overview

The USBX Host Class HID module defines APIs for registering callbacks, starting and stopping periodic reports, and reporting gets and sets. The following tables summarizes the available APIs and includes an example of each, along with a brief description. A table of status return values follows the API summary table.

Table 1. USBX Host Class HID Module API Summary

Function Name	Example API Call and Description
ux_host_class_hid_report_callback_register	`ux_host_class_hid_report_callback_register(hid, &call_back);` This function is used to register a callback from the HID class to the HID client when a report is received.
ux_host_class_hid_periodic_report_start	`ux_host_class_hid_periodic_report_start(hid);`

Function Name	Example API Call and Description
	This function is used to start the periodic (interrupt) endpoint for the instance of the HID class that is bound to this HID client.
ux_host_class_hid_periodic_report_stop	`ux_host_class_hid_periodic_report_stop(hid);` This function is used to stop the periodic (interrupt) endpoint for the instance of the HID class that is bound to this HID client.
ux_host_class_hid_report_get	`ux_host_class_hid_report_get(hid, &client_report);` This function is used to receive a report directly from the device without relying on the periodic endpoint.
ux_host_class_hid_report_set	`ux_host_class_hid_report_set(hid, &client_report);` This function is used to send a report directly to the device.
ux_host_class_hid_keyboard_key_get	`ux_host_class_hid_keyboard_key_get(keyboard_instance, &keyboard_char, &keyboard_state);` This function is used to read the keyboard data received from the device.
ux_host_class_hid_mouse_buttons_get	`ux_host_class_hid_mouse_buttons_get(&mouse_instance, &mouse_buttons);` This function is used to read the mouse button information received from the device.
ux_host_class_hid_mouse_position_get	`ux_host_class_hid_mouse_position_get(mouse_instance, &x_position, &y_position);` This function is used to read the position information of the mouse received from the device.
ux_host_class_hid_remote_control_usage_get	`ux_host_class_hid_remote_control_usage_get(remote_control_instance, &usage, &value);` This function is used to read the remote controller information received from the device.

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures and function variables, review the associated *Express Logic User's Manual* listed in the Reference section.

Table 2. Status Return Values

Name	Description
UX_SUCCESS	The data transfer was completed.
UX_TRANSFER_TIMEOUT	Transfer timeout, reading/writing not completed.
UX_MEMORY_INSUFFICIENT	Not enough memory.
UX_HOST_CLASS_UNKNOWN	Wrong class instance.
UX_FUNCTION_NOT_SUPPORTED	Unknown IOCTL function.

Note: Lower-level drivers may return common error codes. Refer to the *SSP User's Manual* API references for the associated module for a definition of all relevant status return values.

3. USBX Host Class HID Module Operational Overview

Initialization of USBX resources

The USBX has its own memory manager. The memory allocation to the USBX must occur before the host or device side of the USBX is initialized. The USBX memory manager can accommodate systems where memory can be cached.

Definition of USB Host Controllers

It is required to define at least one USB host controller for USBX to operate in host-mode. The application-initialization file should contain this definition. SSP defines USB host controller when USB host controller driver is added to thread stacks.

3.1 USBX Host Class HID Module Important Operational Notes and Limitations

3.1.1 USBX Host Class HID Module Operational Notes

- By default, the module supports a keyboard, mouse, and remote-control clients.
- Use a class container for the USBX Host Class HID that is obtained by auto-generated code to get a HID instance.
- Pole the flag, ux_host_class_hid_state, in the instance and ensure the status is live.
- Check whether or not a client local instance is available in the HID instance.
- Perform HID communication with the USB device.

3.1.2 USBX Host Class HID Module Limitations

- The module uses the interrupt of the USB Controller, which needs to be enabled. For proper operation, set the appropriate interrupt-priority level in the Synergy Configuration tool.
- If transfer module is used, the module uses the interrupt (implemented as DMAC or DTC). Set the priority level in the Synergy Configuration tool. The level must be higher than a USB Controller.
- For other operational limitations to this module, see the latest SSP Release Notes at the Synergy Gallery site.

4. Including the USBX Host Class HID Module in an Application

Note: It is assumed you are familiar with creating a project, adding threads, adding a stack to a thread, and configuring a block within the stack. If you are unfamiliar with any of these items, refer to the first few chapters of the *SSP User's Manual* to learn how to manage each of these important steps in creating SSP-based applications.

To add the USBX Host Class HID module to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the USBX Host Class HID module is `g_ux_host_class_hid0`. This name can be changed in the associated Properties window.)

Table 3. USBX Host Class HID Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_host_class_hid0 USBX Host Class HID	Threads	New Stack> X-Ware> USBX> Host > Classes > HID > USBX Host Class HID

When the USBX Host Class HID module is added to the thread stack (see figure), the configurator automatically adds any needed lower-level modules. Any modules requiring additional configuration have their box text highlighted in Red. Modules with a Gray band are individual, standalone modules. Modules with a Blue band are shared or common, and only need to be added to be used by multiple stacks. Modules with a Pink band may require optional or recommended lower-level modules as indicated. If lower-level modules are needed, the module description includes "Add" in the text. Clicking any Pink banded modules displays the "New" icon that lists possible choices.

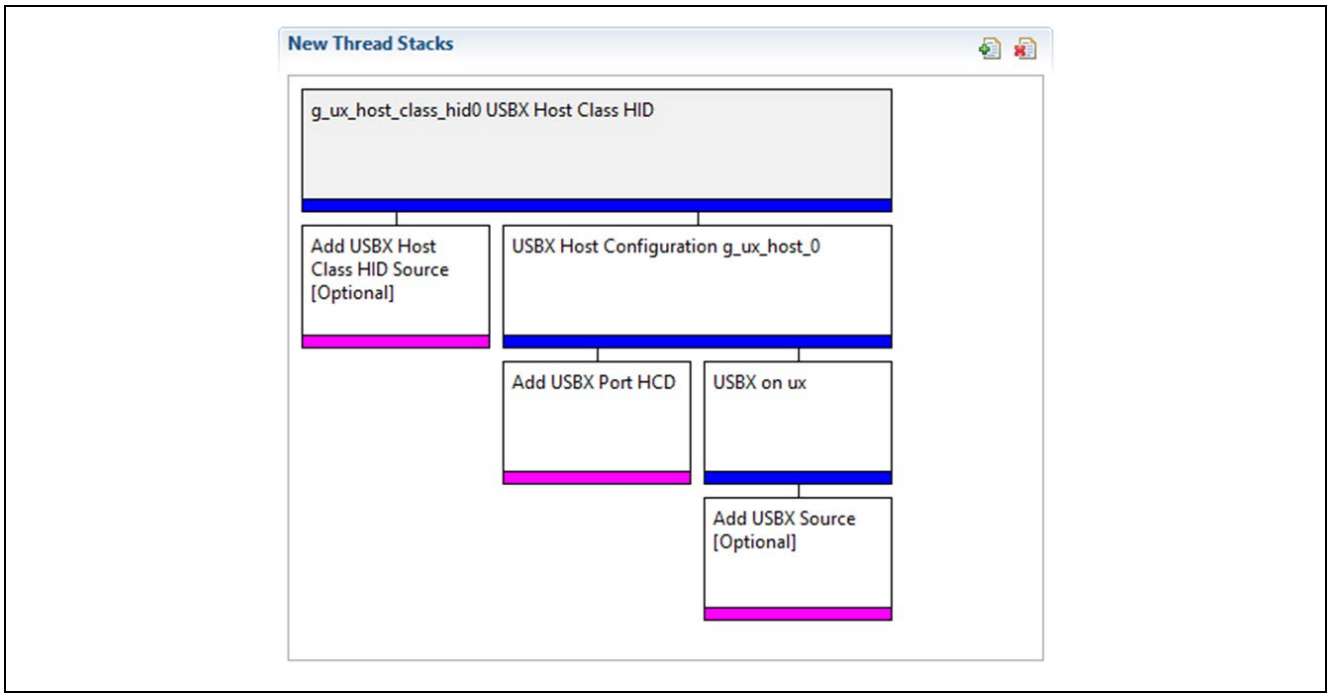

Figure 2. USBX Host Class HID Module Stack

5. Configuring the USBX Host Class HID Module

The USBX Host Class HID module must be configured by the user for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in red) any required configuration selections, such as interrupts or operating modes, that must be configured for lower-level modules for successful operation. Only properties that can be changed without causing conflicts are available for modification. Other properties are 'locked,' not available for changes and identified with a lock icon for the 'locked' property in the ISDE Properties window. This approach simplifies the configuration process and making it much less error-prone than previous 'manual' approaches. The available configuration settings and defaults for all the user-accessible properties are given in the Properties tab within the SSP configurator and indicated in the following tables for easy reference.

One of the properties most often identified as requiring a change is the interrupt priority; this configuration setting is available in the Properties window of the associated module. Simply select the indicated module; the interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Also note that the interrupt priorities listed in the Properties window includes an indication as to the validity of the setting based on the targeted MCU (CM4 or CM0+). This level of detail is not included in the following configuration properties tables but is easily visible within the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE, create the module, and explore the property settings in parallel with reading the following configuration table values. This help orients you and can be a useful 'hands-on' approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for the USBX Host Class HID Module

ISDE Property	Value	Description
Name	g_ux_host_class_hid0	Module name
HID Client - Keyboard Support	Enable, Disable Default: Enable	Keyboard support selection
HID Client - Mouse Support	Enable, Disable Default: Enable	Mouse support selection
HID Client - Remote Control Support	Enable, Disable Default: Enable	Remote control support selection

Note: The example values and defaults for a project using the Synergy S7G2. Other MCUs may have different default values and available configuration settings.

5.1 Configuration Settings for the USBX Host Class HID Module Low-Level Modules

Typically, only a small number of settings must be modified from the default for lower-level modules as indicated via the red text in the thread stack block. Notice that some of the configuration properties must be set to a certain value for proper framework operation and locks to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 5. Configuration Settings for the USBX Host Configuration

ISDE Property	Value	Description
Name	g_ux_host0	Module name
Name of generated initialization function	ux_host_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

Note: The example values and defaults for a project using the Synergy S7G2. Other MCUs may have different default values and available configuration settings.

Table 6. Configuration Settings for the USBX HCD on sf_el_ux for USBFS

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	Full speed interrupt priority selection.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	VBUSEN pin signal logic selection
Name	g_sf_el_ux_hcd_fs_0	Module name.
USB Controller Selection	USBFS	USB controller selection.

Note: The example values and defaults are for a project using the Synergy S7G2. Other MCUs may have different default values and available configuration settings.

Table 7. Configuration Settings for the USBX HCD on sf_el_ux for USBHS

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	High speed interrupt priority selection.
FIFO size for Bulk Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	FIFO size for bulk pipes selection
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	VBUSEN pin signal logic selection
Name	g_sf_el_ux_hcd_hs_0	Module name.
USB Controller Selection	USBHS	USB controller selection.

Note: The example values and defaults are for a project using the Synergy S7G2. Other MCUs may have different default values and available configuration settings.

Table 8. Configuration Settings for USBX on ux

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	USBX pool memory name selection.
USBX Pool Memory Size	18432	USBX pool memory size selection.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	User callback for host even notification selection
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

Note: The example values and defaults are for a project using the Synergy S7 MCU Family. Other MCUs may have different default values and available configuration settings.

Table 9. Configuration for TX Transfer Driver on r_dmac

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	Channel number
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14 ,15 (lowest, not valid if using Thread X), Disabled Default: Disabled	Interrupt priority selection

Note: The above setting examples and defaults are for a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings.

Table 10. Configuration for TX Transfer Driver on r_dtc

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, disabled (Default: Disabled)	Option to start TX transfer through software
Linker section to keep DTC vector table	Default: .ssp_dtc_vector_table	Section to keep DTC vector table in memory.
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Even Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled Default: Disabled	Interrupt priority selection

Note: The example values and defaults are for a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings.

Table 11. Configuration for RX Transfer Driver on r_dmac

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer1	Module name
Channel	1	Channel number
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection

ISDE Property	Value	Description
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15 (lowest, not valid if using Thread X), Disabled Default: Disabled	Interrupt priority selection

Note: The example values and defaults are for a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings.

Table 12. Configuration for RX Transfer Driver on r_dtc

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Option to start TX transfer through software
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to keep DTC vector table in memory.
Name	g_transfer1	Module name
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 2	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Even Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,1 4,15 (lowest, not valid if using Thread X), Disabled Default: Disabled	ELC Interrupt priority

Note: The example values and defaults are for a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings.

5.2 USBX Host Class HID Module Clock Configuration

The USB peripheral module is clocked based on the UCLK frequency. The UCLK frequency must be 48 MHz for USB operation. You can set the UCLK frequency using the clock configurator in e^2 studio or the CGC Interface at run-time.

5.3 USBX Host Class HID Module Pin Configuration

The USB peripheral module uses pins on the MCU to communicate to external devices. I/O pins must be selected and configured as required by the external device. The following table lists the pin-selection methods in the SSP configuration window, with subsequent tables listing USB pin selection examples.

Note: The selected mode of operation determines the peripheral signals available and the MCU pins required.

Table 13. Pin Selection Sequence for USBFS and USBHS

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

Note: The selection sequence assumes USBFS0 or USBHS0 are the desired hardware target for the driver.

Table 14. Pin Configuration Settings for the USBFS

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG (Default: Custom)	Select Device as the Operation Mode
USBDP	USBDP	USBDP Pin
USBDM	USBDM	USBDM Pin
OVRCURB	None	OVRCURB Pin
OVRCURA	None	OVRCURA Pin
VBUSEN	None	VBUSEN Pin
VBUS	None, P407 (Default: P407)	VBUS Pin
EXICEN	None	EXICEN Pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB Pin
VSSUSB	VSSUSB	VSSUSB Pin

Note: The example values are for a project using the Synergy S7G2 and the SK-S7G2 Kit. Other Synergy Kits and other Synergy MCUs may have different available pin configuration settings.

Table 15. Pin Configuration Settings for the USBHS

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP Pin
USBHSDM	USBHSDM	USBHSDM Pin
OVRCURB	None	OVRCURB Pin
OVRCURA	None	OVRCURA Pin
VBUSEN	PB00	VBUSEN Pin
VBUS	PB01	VBUS Pin
EXICEN	None	EXICEN Pin
ID	None	ID Pin
USBHSRREF	USBHSRREF	USBHSRREF Pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS Pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS Pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS Pin

Property	Value	Description
VCCUSBHS	VCCUSBHS	VCCUSBHS Pin
VSS1USBHS	VSS1USBHS	VSS1USBHS Pin
VSS2USBHS	VSS2USBHS	VSS2USBHS Pin

Note: The example values are for a project using the Synergy S7G2 MCU and the SK-S7G2 Kit. Other Synergy MCUs and other Synergy kits may have different available pin configuration settings.

6. Using the USBX Host Class HID Module in an Application

The processing needed to create and register the USBX Host Class HID module is generated by the configurator, with communication done after the device is connected to the host.

The typical steps in using the USBX Host Class HID module in an application are:

1. Get the first instance of the connected device with the `ux_host_stack_class_instance_get` API.
2. Wait for return success
3. Wait for the device status to become live.
4. Wait for the client instance to become live.
5. If the client is a keyboard, for received data reading use the `ux_host_class_hid_keyboard_key_get` API.
6. If the client is a mouse, for received data reading use the `ux_host_class_hid_mouse_buttons_get` API and `ux_host_class_hid_mouse_position_get` API.
7. If the client is a remote controller, for received data reading, use the `ux_host_class_hid_remote_control_usage_get` API.

The following figure has common steps used in a typical operational flow:

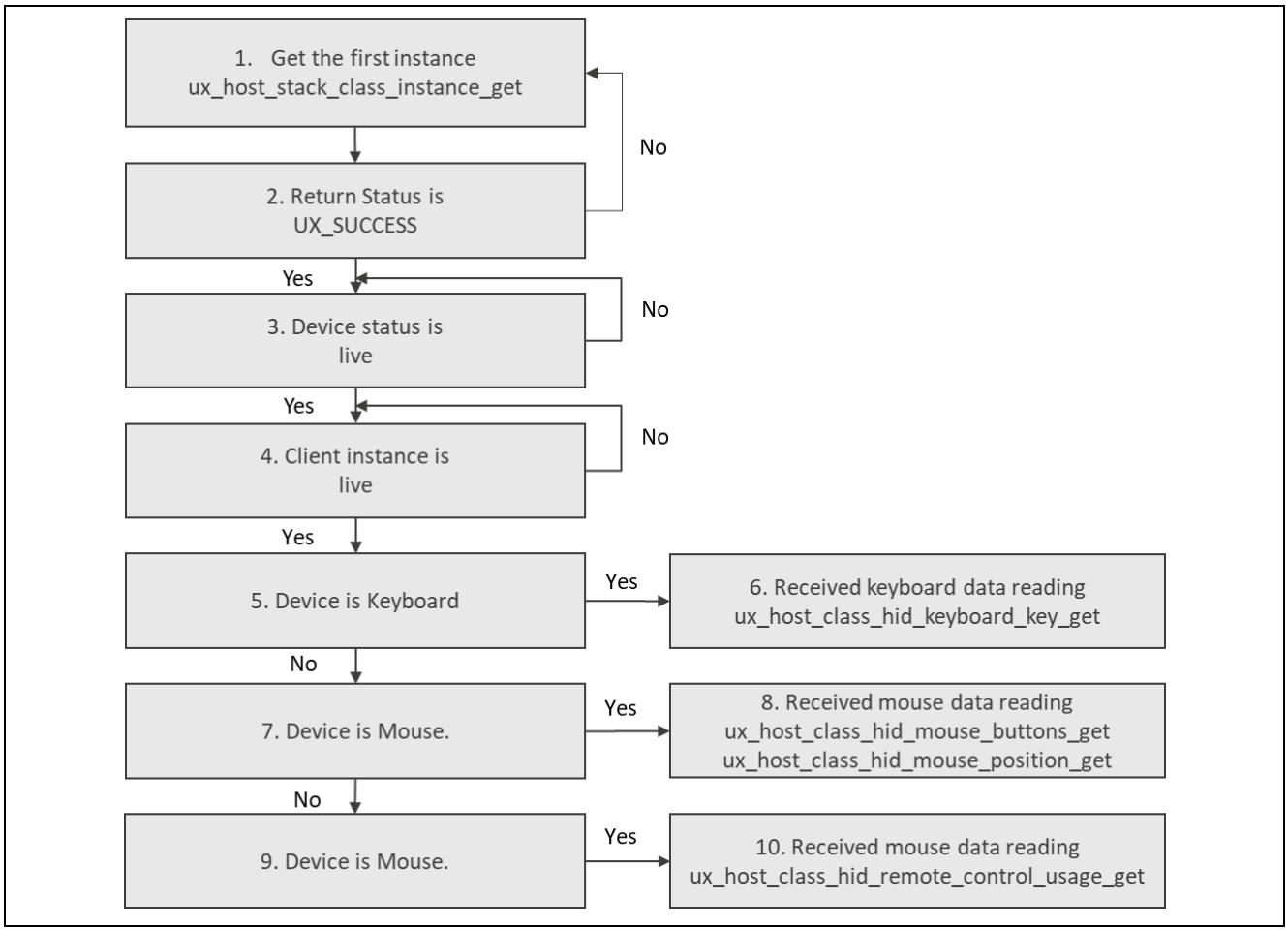

Figure 3. Flow Diagram of a Typical USBX Host Class HID Module Application

7. The USBX Host Class HID Module Application Project

The application project's main thread entry (`hid_thread_entry.c`) initializes the USB Host HID framework. The callback for USB host HID change notifies the application about insertion and deletion events. The application categorizes the device as a HID keyboard device or HID mouse device by reading the interface protocol field in the associated interface descriptor. The Yellow LED (LED1) turns ON/OFF for the keyboard attachment/detachment. The Red LED (LED2) turns ON/OFF on mouse attachment/detachment. The main thread updates the interface pointer used by the `keyboard_update_entry.c` and `mouse_update_entry.c` threads on HID device join/disjoin events and in case multiple similar type devices are connected, the main thread updates the pointer in a round robin fashion to read status of all the connected devices.

Table 16. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	5.4.0.023 or later	Integrated Solution Development Environment
SSP	1.3.0 or later	Synergy Software Platform
IAR EW for Renesas Synergy	7.71.3 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	5.4.0.023 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following figure shows a flow diagram of key application events.

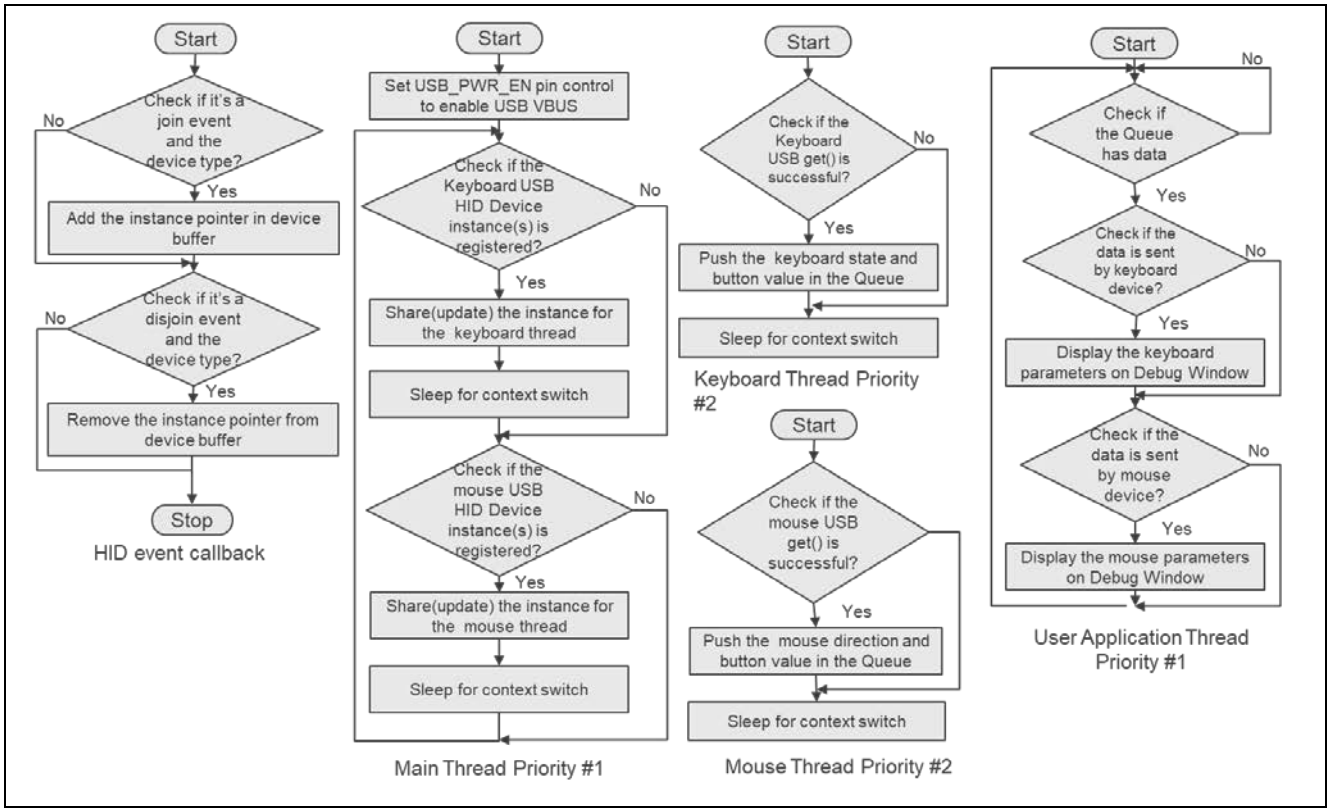

Figure 4. USBX Host Class HID Module Application Project Flow Diagram

The application's flow begins with the HID event callback function. The HID event callback function, once called, determines the nature of event. For a join event, the callback function identifies the type of device connected using device descriptor's protocol interface field. If the device is a keyboard or mouse, then the instance is stored into the keyboard or mouse interface buffer. On a disjoin event, the callback function scans the devices buffers to remove the instance of the device that triggered the disjoin event.

The main thread has the highest priority. It checks if the registered instances in a while loop. If the registered instances are valid, then the instance pointed by keyboard or mouse thread is updated to the valid instance. Control is then passed to the lower priority device threads to get the device parameters by calling thread sleep.

The keyboard_update_entry and mouse_update_entry threads use the get() calls to read device parameters using the interface pointer passed by the main thread. On successful read, the device_update threads pushes the data to the user application (application_entry) thread via shared queues, on an unsuccessful read, no action is performed

The user application thread shares the same priority as of the main thread, however it is blocked on the queue. As soon as the device threads put the value in the queue, that unlocks the user application thread. The user application thread uses the SEMI-HOSTING function to print the keyboard keys and state and/or mouse buttons and horizontal or vertical direction received from the queue on Renesas Debug Virtual Console.

Table 17. USBX Host Class HID Module's USB HS Settings for the Application Project

ISDE Property	Value Set
High Speed Interrupt Priority	Priority 3
FIFO size for Bulk Pipes	2048 bytes
VBUSEN pin Signal Logic	Active High
Enable High Speed	Enable
Name	g_sf_el_ux_hcd_hs_0
USB Controller Selection	USBX Port HCD on sf_el_ux for USBHS or USBFS

Table 18. USBX Host Class HID0 USBX Host Class Settings for the Application Project

ISDE Property	Value Set
Name	g_ux_host_class_hid0
HID Client- Keyboard Support	Enable
HID Client- Mouse Support	Enable
HID Client- Remote Control Support	Disable

Table 19. DMAC TX Transfer Driver's Settings for the Application Project

ISDE Property	Value Set
Parameter Checking	BSP, Enabled
Name	g_transfer0
Channel	0
Mode	Block
Transfer Size	1 Byte
Destination Address Mode	Fixed
Source Address Mode	Incremented
Repeat Area (Unused in Normal Mode	Source
Destination Pointer	NULL
Source Pointer	NULL
Number of Transfers	0
Number of Blocks (Valid only in Block Mode)	0
Activation Source (Must enable IRQ)	Software Activation
Auto Enable	FALSE
Callback (Only valid with Software start)	NULL
Interrupt Priority	Priority 2

Table 20. DMAC RX Transfer Driver's Settings for the Application Project

ISDE Property	Value Set
Parameter Checking	BSP, Enabled
Name	g_transfer1
Channel	1

ISDE Property	Value Set
Mode	Block
Transfer Size	1 Byte
Destination Address Mode	Fixed
Source Address Mode	Incremented
Repeat Area (Unused in Normal Mode	Destination
Destination Pointer	NULL
Source Pointer	NULL
Number of Transfers	0
Number of Blocks (Valid only in Block Mode)	0
Activation Source (Must enable IRQ)	Software Activation
Auto Enable	FALSE
Callback (Only valid with Software start)	NULL
Interrupt Priority	Priority 2

Table 21. Configuration Settings for the USBX on ux

ISDE Property	Value Set
USBX Pool Memory Name	g_ux_pool_memory
USBX Pool Memory Size	133120
User Callback for Host Event Notification	ux_system_host_hid_change_function
Name of generated initialization function	ux_common_init0
Auto Initialization	Enable

8. Customizing the USBX Host Class HID Module for a Target Application

Some configuration settings can normally be changed by the developer from those shown in the application project. For example, the user can easily change the configuration settings for the USBx on ux; the user can modify “pool memory size” to increase or decrease the default number of HID devices (8) that could connect to the bus.

9. Running the USBX Host Class HID Module Application Project

To run the USBX Host Class HID module application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug. See the *Renesas Synergy™ Project Import Guide* (in the project package) for instructions on importing the project into e^2 studio or the IAR EW for Synergy and building/running the application.

Note: The following steps are in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, see the SSP User’s Manual; it includes a user tutorial in the first few chapters.

Use the following steps to create and run the USBX Host Class HID Application Project:

1. Import and build the example project according to the *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf) included with this module guide.
2. Connect a USB keyboard or USB mouse or USB wireless receiver dongle (for keyboard and mouse) to SK-S7G2 via connector J6.
3. The output can be viewed in the Renesas Debug Console.

```
Renesas Debug Virtual Console

**** Keyboard is connected ****

 vID:[redacted]    pID:[redacted]
**** Mouse is connected ****

 vID:[redacted]    pID:[redacted]
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
 NUM Lock is ON
 key pressed is:     g
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
 NUM Lock is ON
 key pressed is:     g
Mouse position X:0,Y:0 ,Left click,0,Middle click 0 , Right click 0
 NUM Lock is ON
 key pressed is:     h
```

Figure 5. Example Output from USBX Host Class HID Module Application Project

10. USBX Host Class HID Module Conclusion

This module guide has provided the background needed to select, add, configure, and use the module in an example project. Many of these steps were time-consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy Platform makes these steps less time-consuming and removes common errors, like conflicting configuration settings or incorrect selection of lower-level drivers. The use of high-level APIs demonstrated in this application project highlights the development time savings that can be achieved. Allowing work to begin at a higher level avoids the time needed to use or create, lower-level drivers prevalent in older development environments.

11. USBX Host Class HID Module Next Steps

After you have mastered a simple USBX Host Class HID module project, you may want to review a more complex example. Other application projects and application notes that demonstrate USBX Host Class HID use can be found as described in the References section at the end of this document.

12. USBX Host Class HID Module Reference Information

SSP User Manual: Available in html format in the SSP distribution package and as a pdf from the Synergy Gallery.

Links to all the most up-to-date USBX Host Class HID module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977574.

Website and Support

Visit the following vanity URLs to learn about key elements of the Synergy Platform, download components and related documentation, and get support.

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Dec 8, 2017	—	Initial version
1.01	Mar.08.19	5, 14	Updates to the configuration description and running the application project

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

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

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

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