Renesas Synergy™ Platform

USBX™ Device 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™ Device Class Human Interface Design (HID) module is a high-level API for HID applications and is implemented on g_ux_device_class_hid. The USBX Device Class HID module configures the USBX Device Class HID Source, USBX Host Configuration, USBX Source, and USBX Port HCD. The USBX Device Class HID module uses the USB peripheral on the Synergy MCU.

Contents

1. USBX Device Class HID Module Features

The USB Device Class HID module allows a USB host system to communicate with the device as a keyboard device, a mouse device, and other HID devices. This class is based on the USB standard and is a subset of the HID standard. The USBX Device Class HID module includes the following key features:

- Support for USB high speed (USBHS) or full speed (USBFS)
- Uses Receive and Transmit data-transfer drivers for improved performance
- Provides high-level APIs for read and write

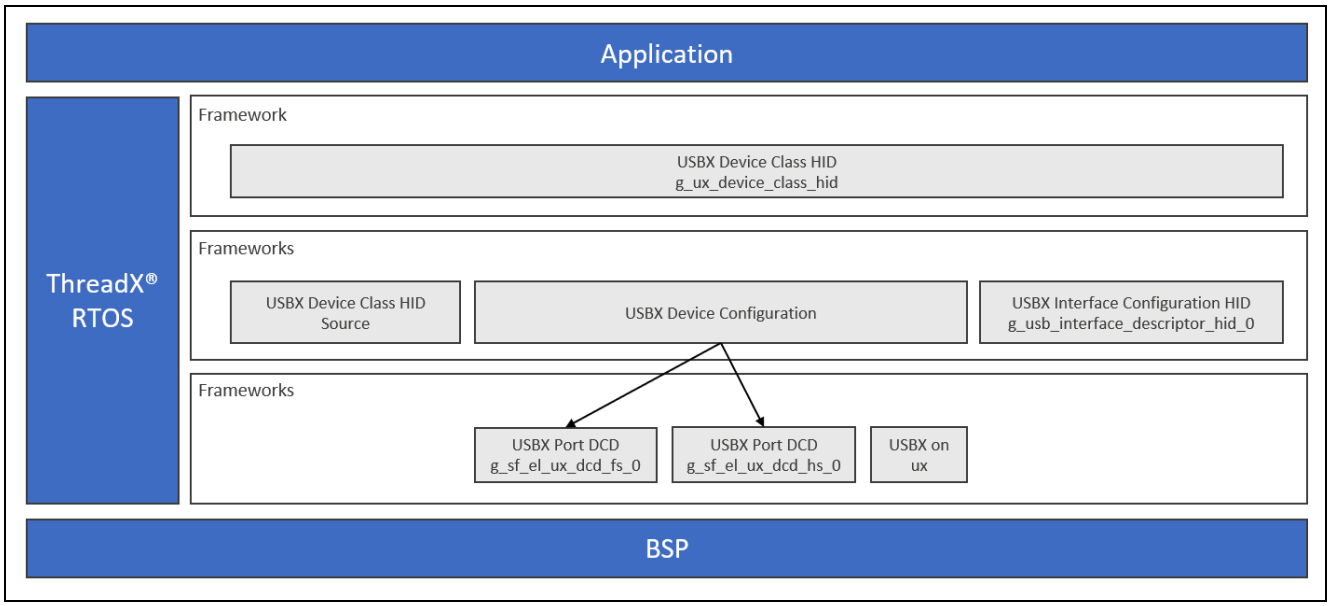

Figure 1. USBX Device Class HID Module Block Diagram

2. USBX Device Class HID Module APIs Overview

The USBX Device Class HID module defines APIs for sending and receiving HID events and reports. The following table has a complete list of the available APIs, an example API call, and a short description. A table of status return values follows the API summary.

Table 1. USBX Device Class HID Module API Summary

Function Name	Example API Call and Description
ux_device_class_hid_event_set	`ux_device_class_hid_event_set (hid, &hid_event);` This function is called when an application needs to send a HID event to the host.
ux_device_class_hid_event_get	`ux_device_class_hid_event_get (hid, & hid_event);` This function is called when an application needs to receive a HID event from the host.
ux_device_class_hid_report_set	`ux_device_class_hid_report_set (hid, descriptor_type, request_index, host_length);` This function is called when an application needs to send a HID report to the host.
ux_device_class_hid_report_get	`ux_device_class_hid_report_get (hid, descriptor_type, request_index, host_length);` This function is called when an application needs to receive a HID report to the host.

Note: For details on operation, as well as definitions of function data structures, typedefs, defines, API data, API structures, and function variables, review the associated Express Logic User's Manual 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. See the associated module in the *SSP User's Manual* API References for a definition of all the relevant status return values.

3. USBX Device Class HID Module Operational Overview

Initialization of USBX resources

The USBX has its own memory manager. The memory needs to be allocated to the USBX 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; in this case, the USB Device Class will be USB HID.

Definition of Device Classes

It is required to define one or more Device Classes with the USBX. A USB class is required to drive a USB device after the USB stack has configured the USB device. A USB class is very specific to the device. One or more classes may be required to drive a USB device depending on the number of interfaces contained in the USB device descriptors.

USB Class Binding

When the device is configured, the topology manager will let the class manager continue the device discovery by looking at the device-interface descriptors. A device can have one or more interface descriptors.

An interface represents a function in a device. For instance, a USB speaker has three interfaces, one for audio streaming, one for audio control, and one to manage the various speaker buttons.

The class manager has two mechanisms to join the device interface(s) to one or more classes. It can either use the combination of a PID/VID (product ID and vendor ID) found in the interface descriptor or the combination of Class/Subclass/Protocol.

The PID/VID combination is valid for interfaces that cannot be driven by a generic class. The Class/Subclass/Protocol combination is used by interfaces that belong to a USB-IF certified class such as a printer, hub, storage, audio, or HID.

The class manager contains a list of registered classes from the initialization of USBX. The class manager calls each class one-at-a-time until one class accepts to manage the interface for that device. A class can only manage one interface. For the example of the USB audio speaker, the class manager calls all the classes for each of the interfaces.

Once a class accepts an interface, a new instance of that class is created. The class manager then searches for the default alternate setting for the interface. A device may have one or more alternate settings for each interface; the alternate setting 0 is the one used by default until a class decides to change it.

For the default alternate setting, the class manager mounts all the endpoints contained in the alternate setting. If the mounting of each endpoint is successful, the class manager completes its job by returning to the class that finishes the initialization of the interface.

3.1 USBX Device Class HID Module Important Operational Notes and Limitations

3.1.1 USBX Device Class HID Module Operational Notes

- The application gets the HID instance of the slave device from the global variable, _ux_system_slave; sends and receives executed using this instance.
- Use the **Protocol code** property of [USBX Interface Configuration HID Driver] to determine the operation of the actual device.

3.1.2 USBX Device Class HID Module Limitations

- The module needs the interrupt of a USB Controller enabled.
- The module uses the interrupt of a USB Controller. Set the appropriate interrupt-priority level in the Synergy Configuration tool for proper operation.
- The module uses the interrupt of a transfer module (implemented as DMAC or DTC) if it is used. Set the appropriate priority level in the Synergy Configuration tool and the level must be higher than a USB Controller; otherwise, it does not work.

Note: See the latest SSP Release Notes for any other operational limitations applicable to this module.

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

This section describes including the USBX Device Class HID module in an application using the SSP configurator.

Note: It is assumed that 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 tasks, refer to the *SSP User's Manual* usage notes to learn how to manage these important steps in creating SSP-based applications.

To add the USBX Device 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 Device Class HID module is `g_ux_device_class_hid`. This name can be changed in the associated Properties window.)

Table 3. USBX Device Class HID Selection Sequence

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

When the USBX Device Class HID module is added to the thread stack, the configurator automatically adds any needed lower-level modules. Any modules needing additional configuration information are box text highlighted in Red. Modules with a Gray band are individual, standalone modules. Modules with a Blue band are shared or common; they only need to be added once and can be used by multiple stacks. Modules with a Pink band can require selecting lower-level modules; these are either optional or recommended. If lower-level modules need to be added, the module description includes "Add" in the text. Clicking on any Pink banded modules brings up the "New" icon and then displays the possible choices.

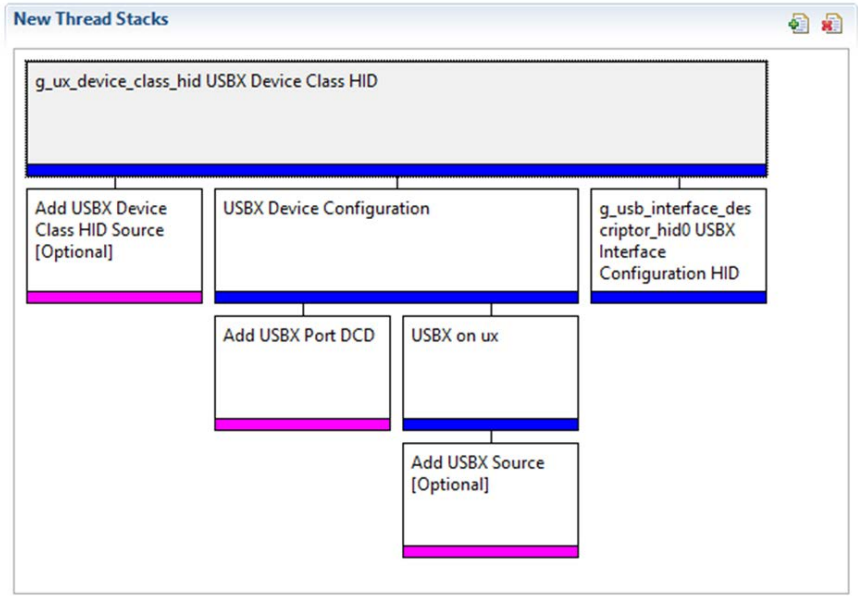

Figure 2. USBX Device Class HID Module Stack

5. Configuring the USBX DEVICE CLASS HID USBX Device Class HID Module

The USBX Device 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, which must be configured for lower-level modules for successful operation. Only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and unavailable for changes, and they are identified with a lock icon for the 'locked' property in the Properties window in the ISDE. This approach simplifies the configuration process and makes it much less error-prone than previous 'manual' approaches to configuration. The available configuration settings and defaults for all the user-accessible properties are given in the Properties tab within the SSP configurator and listed in the following tables for reference.

One of the properties often changed is the interrupt priority; this configuration setting is available in the Properties window of the associated module. Simply select the indicated module to view it in the Properties window; the interrupt settings are toward the bottom of the properties list, so scroll down until they become available. Also note the interrupt priorities listed in the Properties window in the ISDE indicates 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 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 looking over the following configuration table values. This helps to orient you and can be a useful 'hands-on' approach to learning the ins and outs of developing applications with the SSP.

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

ISDE Property	Value	Description
Name	g_ux_device_class_hid	Module name.
USBX HID User Callback Function	ux_hid_device_callback	USBX HID user callback function selection.

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

In some cases, settings other than the defaults for a module can be desirable. For example, it might be useful to select a different clock source than the default. As a reference, the configurable properties for the lower-level stack modules are given in the following sections.

Note: Most of the property settings for lower-level modules are intuitive to apply and usually can be determined by inspection of the associated Properties window from the SSP configurator.

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

Typically, only a small number of settings must be modified from the default for lower-level modules, and these are 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 are locked to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 5. Settings for the USBX Device configuration

ISDE Property	Value	Description
Vendor ID	0x045B	Vendor ID selection
Product ID	0x0000	Product ID selection
Device Release Number	0x0000	Device Release Number selection
Index of Manufacturing String Descriptor	0x00	Index of Manufacturing String Descriptor selection
Index of Product String Descriptor	0x00	Index of Product String Descriptor selection
Index of Serial Number String Descriptor	0x00	Index of Serial Number String Descriptor selection
Class Code	Communications (CDC), HID, Mass Storage, Miscellaneous, Vendor specific Default: Communications (CDC)	Class Code selection
Index of String Descriptor describing this configuration	0x00	Index of String Descriptor describing this configuration selection
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Size of USB Descriptor in bytes for this configuration selection
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Number of Interfaces selection
Self-Powered	Enable, Disable Default: Enable	Self-Powered selection
Remote Wakeup	Enable, Disable Default: Disable	Remote Wakeup selection
Maximum Power Consumption (in 2mA units)	50	Maximum Power Consumption selection
Supported Language Code	0x0409	Supported Language Code selection
Name of USBX String Framework	NULL	Name of USBX String Framework selection
Total index number of USB String Descriptors in USB String Framework	0	Total index number of USB String Descriptors in USB String Framework selection
Name of USBX Language Framework	NULL	Name of USBX Language Framework selection
Number of Languages to support (US English is applied if zero is set)	0	Number of Languages to support selection

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

Table 6. Settings for the USBX Interface Configuration HID

ISDE Property	Value	Description
Name	g_usb_interface_descriptor_hid0	Module name.
Interface Number of HID Class interface	0x00	Interface Number of HID Class interface selection.
Protocol code (None(0) /Keyboard(1) /Mouse(2))	1	Protocol code (None(0) /Keyboard(1) /Mouse (2)) selection.
Endpoint Number to be used for Interrupt-In	Endpoint 1-9 Default: Endpoint 1	Endpoint Number to be used for Interrupt-In selection.
Maximum packet size in bytes for Interrupt-In EP	0x8	Maximum packet size in bytes for Interrupt-In selection.
Interval for polling Interrupt-In EP for data transfers (milliseconds)	0x8	Interval for polling Interrupt-In EP for data transfers (milliseconds) selection.
Interrupt-Out Endpoint (Optional)	Enable, Disable Default: Disable	Interrupt-Out Endpoint (Optional) selection.
Endpoint Number for Interrupt-Out EP (Optional)	Endpoint 1-9 Default: Endpoint 2	Endpoint Number for Interrupt-Out EP (Optional) selection.
Maximum packet size in bytes for Interrupt-Out EP (Optional)	0x8	Maximum packet size in bytes for Interrupt-Out EP (Optional) selection.
Interval for polling Interrupt-Out EP for data transfers (milliseconds) (Optional)	0x8	Interval for polling Interrupt-Out EP for data transfers (milliseconds) (Optional) selection.

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

Table 7. Settings for the USBX DCD on sf_el_ux for the USBFS configuration

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.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	USB controller selection.

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

Table 8. Settings for the USBX DCD on sf_el_ux for USBHS configuration

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.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	USB controller selection.

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

Table 9. Settings for USBX on ux configuration

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 event notification (only valid for USB host)

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

5.2 USBX Device 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 Device 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 pin selection table lists the method used in selecting the pins within the SSP configuration window. The pin configuration table includes an example selection for the USB pins.

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

Table 10. 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 11. Pin Configuration Settings for the USBFS

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG (Default: Disabled)	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 settings 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.

Table 12. Pin Configuration Settings for the USBHS

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Disabled	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP Pin
USBHSDM	USBHSDM	USBHSDM Pin
OVRCURB	None	OVRCURB Pin

Property	Value	Description
OVRCURA	None	OVRCURA Pin
VBUSEN	None	VBUSEN Pin
VBUS	None	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
VCCUSBHS	VCCUSBHS	VCCUSBHS Pin
VSS1USBHS	VSS1USBHS	VSS1USBHS Pin
VSS2USBHS	VSS2USBHS	VSS2USBHS Pin

Note: The example settings 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 Device Class HID Module in an Application

The configurator generates processing to create and register the USBX Device Class HID module; however, communication must be done after the device is connected to the host.

Using the USBX Device Class HID module in an application typically involves these steps:

1. Get the _ux_system_slave slave device pointer
2. Wait until slave device's ux_slave_device_state is configured
3. For HID event sending, use the `ux_device_class_hid_event_set` API
4. For received HID event reading, use the `ux_device_class_hid_event_get` API
5. For HID report sending, use the `ux_device_class_hid_report_set` API
6. For received HID report reading, use the `ux_device_class_hid_report_get` API

The following figure depicts common steps in a typical operational flow diagram.

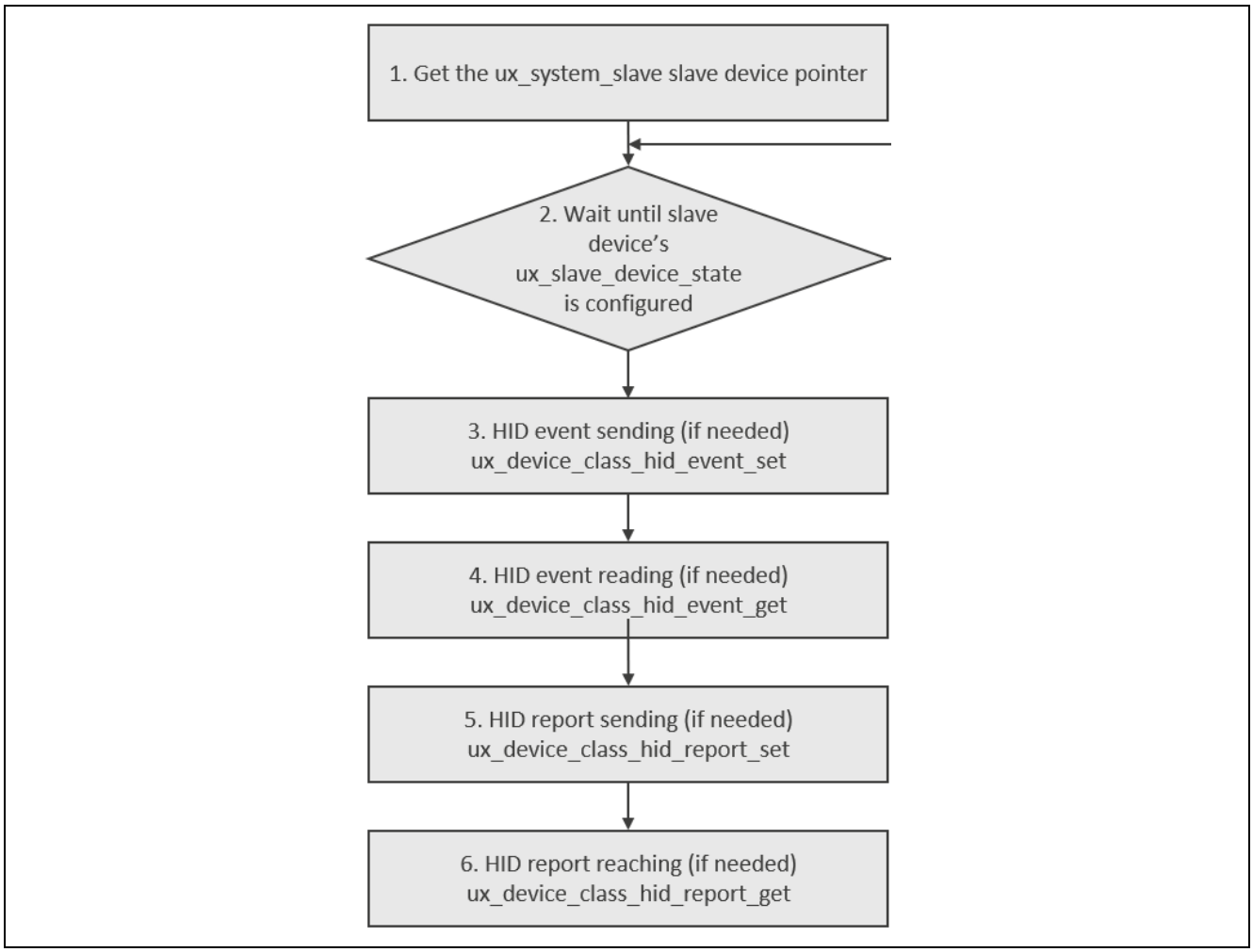

Figure 3. Typical steps in USBX Device Class HID Module application

7. The USBX Device Class HID Module Application Project

The application project demonstrates the typical use of the USBX Device Class HID module APIs. The application project main thread entry routine gets the device pointer from _ux_system_slave and sets the keyboard event periodically; the key code is updated every time the keyboard event is set and can be received on the connected host side. For example, if the host side is Windows and you display the command prompt, you can see that characters according to this key code are entered in the command prompt. The following table identifies the target versions for the associated software and hardware used by the Application Project.

Table 13. Software and hardware resources Used by the application project

Resource	Revision	Description
e^2 studio3	7.3.0 or later	Integrated Solution Development Environment
SSP	1.6.0 or later	Synergy Software Platform
IAR EW for Renesas Synergy	8.23.1or later	IAR Embedded Workbench for Renesas Synergy
SSC	7.3.0 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.3	Starter Kit

A simple flow diagram of the Application project is given in the following figure:

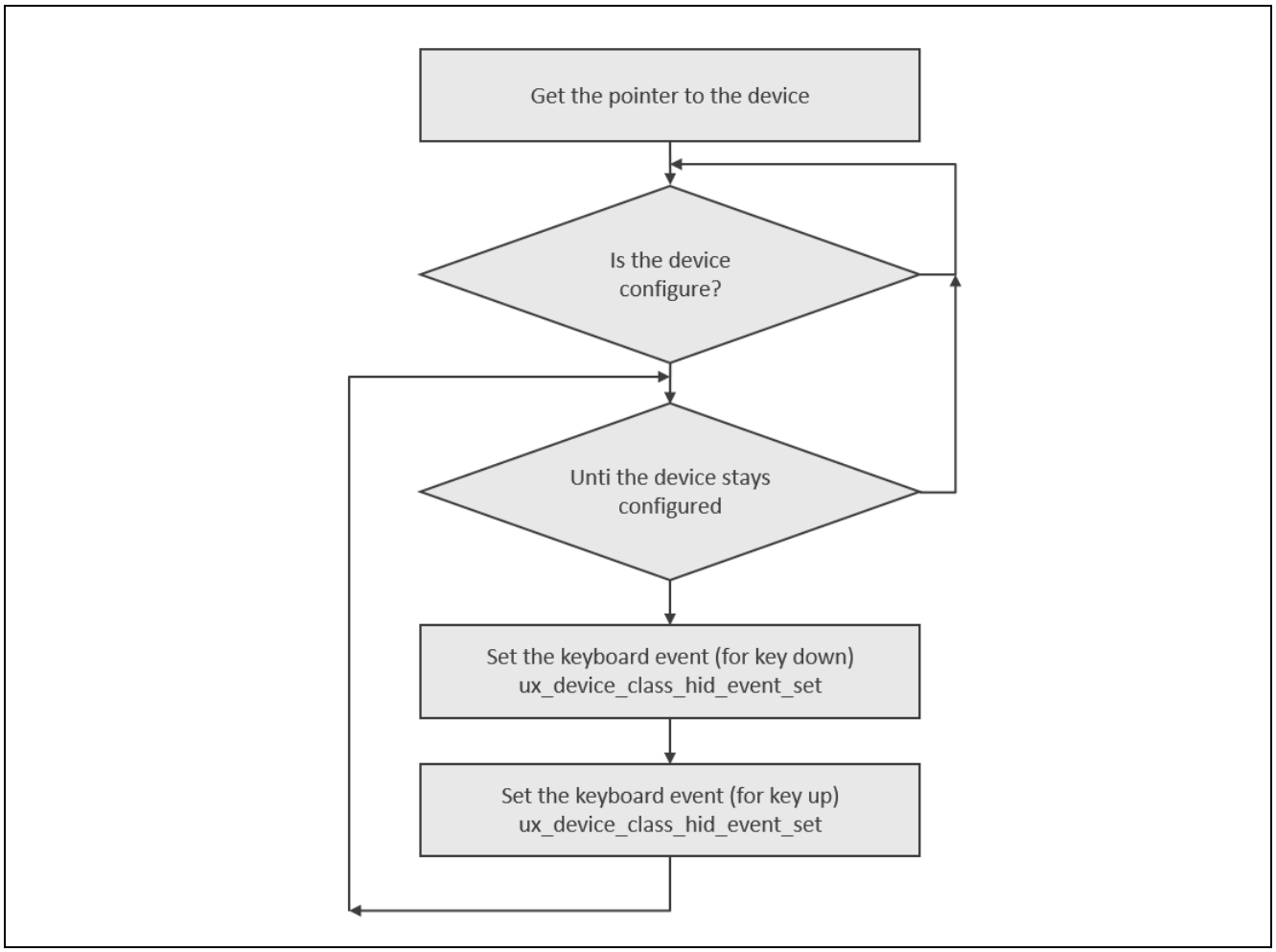

Figure 4. USBX Device Class HID Module application project flow

The first section of `usb_hid_keyboard_device_thread.c` has the header file that references the USB instance structure defined by the configurator for the user thread. The next section defines variables that store Num Lock and Caps Lock statuses of the keyboard. The last section is the user thread that sets the keyboard events. It gets the pointer of the slave device from _ux_system_slave->ux_system_slave_device where the instance of the device is stored. It then prepares the first key code 'a' and the data area for the key event. Next, it waits for the device instance status to become UX_DEVICE_CONFIGURED using the while loop. While the device instance status is UX_DEVICE_CONFIGURED, it prepares a key event in the data area and sets it with ux_device_class_hid_event_set(). The data to be set is the key code for the USB HID class. If the device instance status is no longer UX_DEVICE_CONFIGURED, wait until it is UX_DEVICE_ CONFIGURED again. In `ux_hid_callback.c`, a user-callback function for HID is defined.

The ux_hid_device_callback function is used to process HID host requests. Since this callback function is called when the cable is connected, processing to initialize Num Lock and Caps Lock status is implemented.

Table 14. USBX Device Class HID Module Configuration Settings for the Application Project

ISDE Property	Value Set
USBX Device Configuration Class Code	HID
g_usb_interface_descriptor_hid0 Protocol code (None (0)/Keyboard(1)/Mouse(2)/Keyboard+Mouse(3))	1
USBX Port DCD on sf_el_ux Full Speed Interrupt Priority	Priority 3

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

Some configuration settings are normally changed by the developer from those shown in the application project. For example, you can add a data-transfer module for data transfer of the USBX Port DCD. This data-transfer module can be added simply by clicking on the box for TX or RX displayed under the USBX Port DCD box of the configurator.

9. Running the USBX Device Class HID Module Application Project

To run the USBX Device Class HID module application project and to see it executed on a target kit, you simply import the project into your ISDE, compile, and run debug. See the *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf) included in this package for instructions on importing the project into e^2 studio or the IAR EW for Synergy to build and run the application.

To implement the USBX Device Class HID module application in a new project, use the following steps for defining, configuring, auto-generating files, as well as adding code, compiling, and debugging on the target kit. Following these steps provides a hands-on approach to help make the development process with SSP more practical, while just reading over this guide tends to be more theoretical.

Note: The steps are in sufficient detail for someone experienced with the basic Synergy development process flow. If these steps are unfamiliar, review the first few chapters in the *SSP User's Manual*.

To create and run the USBX Device Class HID Application Project simply follow these steps:

1. Create a new Renesas Synergy project for the S7G2-SK called **USBX_HID_Keyboard**.
2. Select the **Threads** tab.
3. Add the usb_hid_keyboard_device_thread to **Threads**
4. Add the **USBX Device Class HID** module to the **usb_hid_keyboard_device_thread** stack.
5. Click the **USBX Device Configuration** box on the **usb_hid_keyboard_device_thread** stack.
6. Change the Class Code to **HID** in the Properties window.
7. Click the **Add USBX Port DCD** box on the **usb_hid_keyboard_device_thread** and select **USBX Port DCD on sf_el_ux for USBFS**.
8. Change the Full Speed Interrupt Priority to **Priority 3** in the Properties window.
9. Click the **Generate Project Content** button.
10. Add the code from the supplied project file `usb_hid_keyboard_device_thread_entry.c` or copy over the generated `usb_hid_keyboard_device_thread_entry.c` file.
11. Copy the supplied project file `ux_hid_callback.c` to the project.
12. Connect to the host PC via a micro USB cable based on the board you are using for your testing

Board	Debug Connector (DEBUG_USB)
SK-S7G2	J19
PK-S5D9	J19
PE-HMI1	J12 via J-link Adaptor
DK-S7G2	J17 – On V3.1 J34 - On V4.1
DK-S3A7	J15
DK-S124	J18

13. Start to debug the application.
14. Start the command prompt on the host PC.
15. Connect to the host PC via a micro USB cable based on the board you are using for your testing:

Board	Connector (USB Device)
SK-S7G2	J5
PK-S5D9	J5
PE-HMI1	J2
DK-S7G2	J2 – On V3.1 J13 - On V4.1

Board	Connector (USB Device)
DK-S3A7	J2 Note: DIP switch S6 - 5 (USBF) should be ON
DK-S124	J14

16. The output can be viewed in the command prompt.

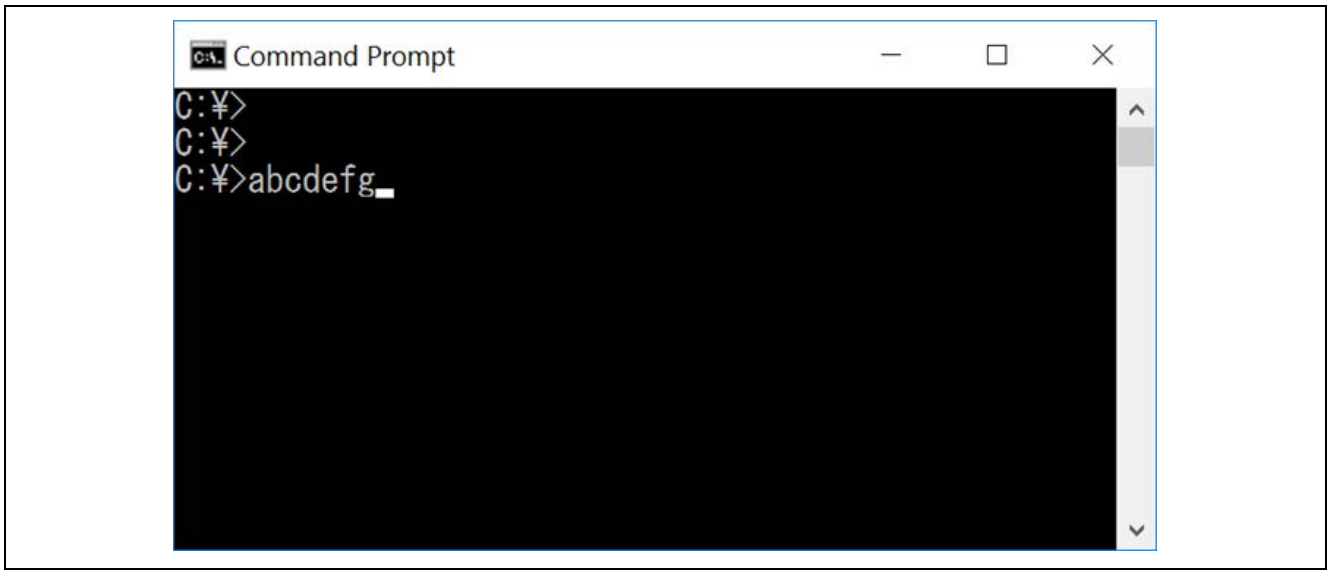

Figure 5. Example Output from USBX Device Class HID Module Application Project

10. USBX Device Class HID Module Conclusion

This module guide has provided all the background information 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 much less time consuming and removes the common errors, like conflicting configuration settings or incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrates additional development time savings by allowing work to begin at a high level and avoiding the time required in older development environments to use or, in some cases, create, lower-level drivers.

11. USBX Device Class HID Module Next Steps

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

You may find that the USBX Device Class HID Framework is a better fit for your target application. The USBX Device Class HID Framework module guide illustrates the use of the USB HID class within a ThreadX®-based implementation; this guide is available as described in the References section at the end of this document.

12. USBX Device 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 up-to-date USBX Device Class HID module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977571.

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

Rev.	Date	Description	
		Page	Summary
1.00	Jun.07.17	—	Initial version
1.01	Jan.12.18	—	Updated versions and configuration settings
1.02	May.10.18	—	Added .module_description to SK-S7G2 project
1.03	Jan.07.19	—	Updated versions and configuration settings
1.04	May.03.19	—	Updated versions and configuration settings

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/.