Renesas Synergy™ Platform

USBX™ Host Class Hub 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 on the Renesas Synergy™ Knowledge Base (as described in the References section at the end of this document) and will be valuable resources for creating more complex designs.

The USBX™ Host Class Hub module is a high-level API for USBX Host Class Hub applications and is implemented on g_ux_host_class_hub. The USBX Host Class Hub module configures the USBX Host Class Hub Source, USBX Host Configuration, USBX Source, and USBX Port Host Controller Device. The USBX Host Class Hub module uses the USB peripheral on the Synergy MCU.

Prerequisites

The following prerequisites should be done so the reader is most efficient using this guide. References for all the material listed in the prerequisites can be found in the Reference Section at the end of this document.

- A reader who wishes to use this guide as a hands-on method for implementing the application example (as opposed to just a learning reference) will need to have an ISDE (e^2 studio or IAR EW for Renesas Synergy with the appropriate version of SSP) installed and running on a computer. The *Getting Started Guide* for the Renesas Synergy Platform can be used for this prerequisite.
- In addition to the ISDE and SSP, a hardware target is needed to see the example project running on a Synergy MCU. For this prerequisite, a kit can be purchased from any Renesas authorized distributor. From the distributor's website by simply search for the kit name. The kit targeted by this application example project is SK-S7G2.
- This guide assumes the reader is familiar with using a Synergy ISDE and the SSP to create projects, add threads, create stacks, configure modules, run, and debug. Reading *Getting Started Guides*, (for e^2 studio, IAR Embedded Workbench® for Renesas Synergy™, and the SSP), as well as viewing introductory videos and taking tutorial labs, all can be used to satisfy this prerequisite.

It typically takes about 40 minutes to complete this guide (including the hands-on exercises.)

A reader who completes and understands the material and methods used in this guide will be able to create their own USBX Host Class Hub application with confidence as well as view and understand the code and learn more advanced techniques on their own.

Contents

1. USBX Host Class Hub Module Features

The hub class is in charge of driving USB hubs. It includes the following features:

- Supports either a stand-alone hub or functions as part of a compound device such as a keyboard or a monitor.
- Supports either self-powered or bus-powered modes.
- Bus-powered hubs have a maximum of four downstream ports.
- Hubs can be cascaded.
- Up to five hubs can be connected to one another.
- Supports the connection of devices that are either self-powered or bus-powered using less than 100 mA of power.

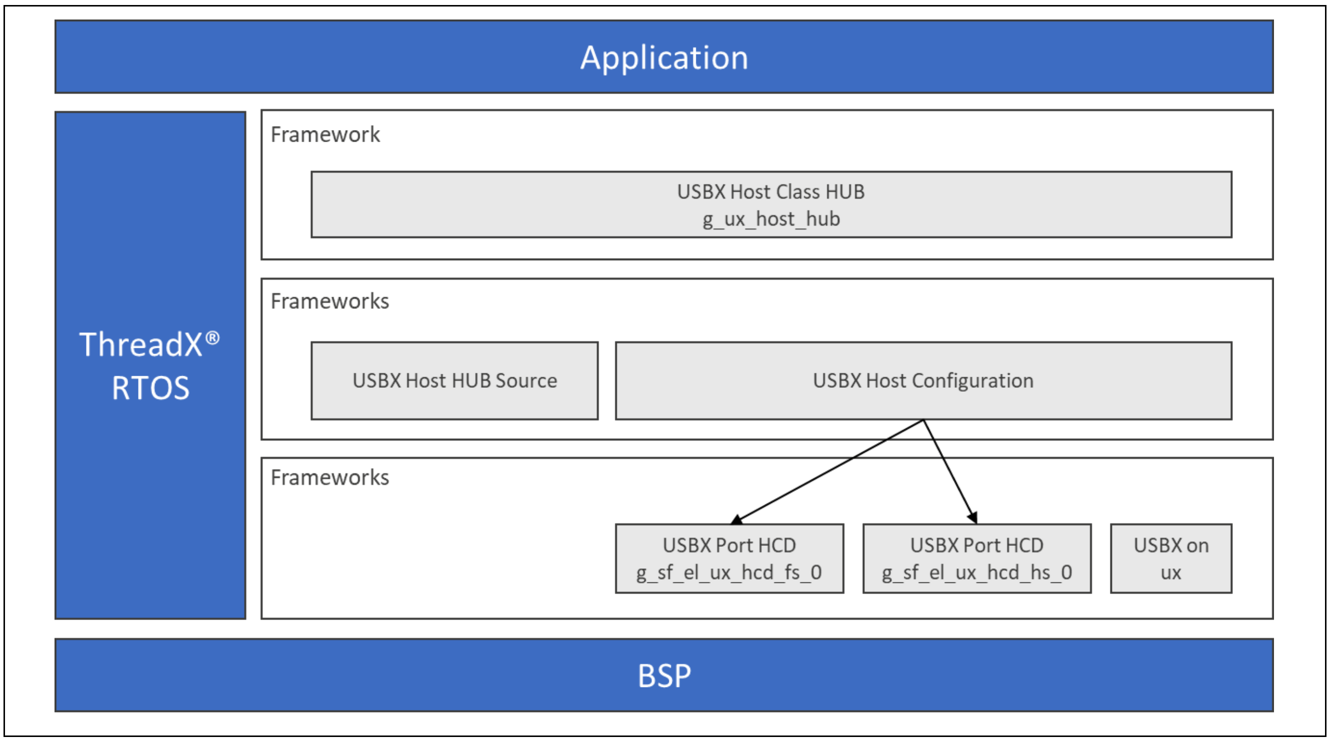


Figure 1. USBX Host Class Hub Module Organization, Options and Stack Implementations

2. USBX Host Class Hub Module APIs Overview

The USBX Host Class Hub Module does not have an API for the user application.

3. USBX Host Class Hub 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. SSP defines USB host controller when USB host controller driver is added to thread stacks.

Definition of Device Classes

It is required to define one or more device classes(s) 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 lets 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 Human Interface Design (HID).

The class manager contains a list of registered classes from the initialization of the USBX. The class manager calls each class one-at-a-time until one class accepts to manage the interface for that device; each class can only manage one interface. In the case 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 will be the one used by default until a class decides to change it.

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

3.1 USBX Host Class Hub Module Important Operational Notes and Limitations

3.1.1 USBX Host Class Hub Module Operational Notes

- The Hub class is registered by the auto-generated code.
- In the user application, no special operation other than the registration of the Hub class is necessary.

3.1.2 USBX Host Class Hub Module Limitations

- The module needs the interrupt of a USB Controller enabled.
- The module uses the interrupt of a USB Controller. Set 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 one is implemented. Set the appropriate priority level in the Synergy Configuration tool. The priority level must be higher than that of the USB Controller for proper operation.
- Refer to the most recent *SSP Release Notes* for any additional operational limitations for this module.

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

This section describes how to include the USBX Host Class Hub 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 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 Hub 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 Hub Module is `g_ux_host_class_hub0`. This name can be changed in the associated **Properties** window.)

Table 1. USBX Host Class Hub Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
`g_ux_host_class_hub0` USBX Host Class Hub	Threads	New Stack> X-Ware™> USBX> Host > Classes > Hub > USBX Host Class Hub

When the USBX Host Class Hub module is added to the thread stack (see the following figure), the configurator automatically adds any needed lower-level modules. Any modules that need additional configuration information have box text highlighted in Red. Modules with a Gray band are individual modules that stand alone. Modules with a Blue band are shared or common. They need only be added once and can be used by multiple stacks. Modules with a Pink band can require the selection of lower-level modules; these are either optional or recommended. (This is indicated in the block with the inclusion of this text.) If the addition of lower-level modules is required, the module description will include Add in the text. Clicking on any Pink banded modules will bring up the **New** icon and then display the possible choices.

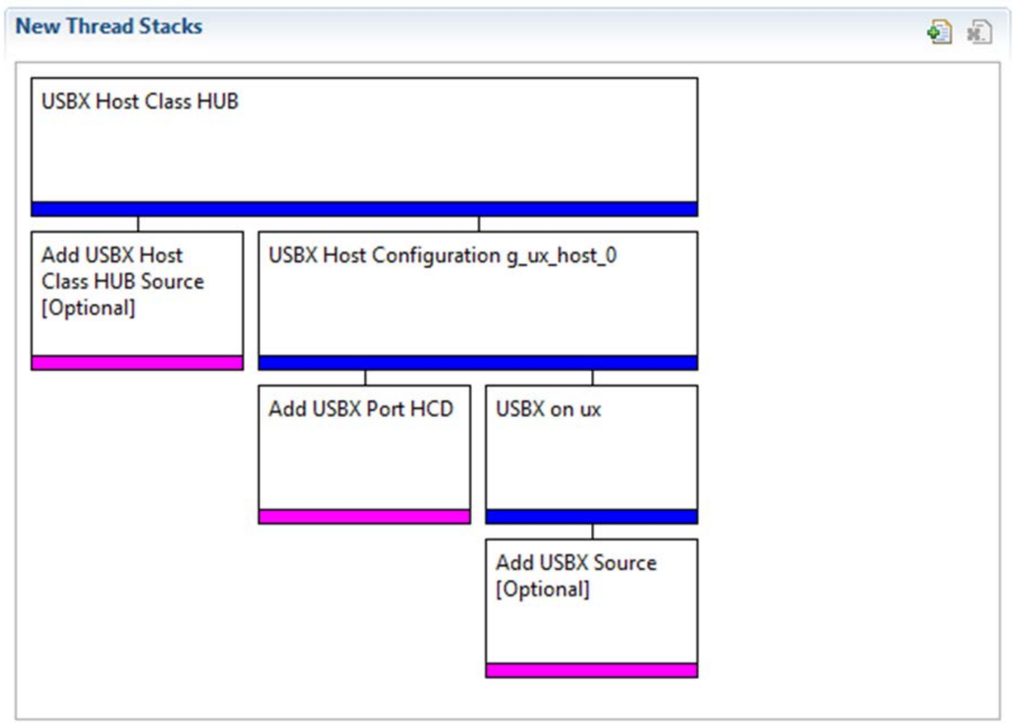

New Thread Stacks
USBX Host Class HUB
Add USBX Host Class HUB Source [Optional]
USBX Host Configuration g_ux_host_0
Add USBX Port HCD
USBX on ux
Add USBX Source [Optional]

Figure 2. USBX Host Class Hub Module Stack

5. Configuring the USBX Host Class Hub Module

The USBX Host Class Hub module has no configurable properties associated with it; configuration is performed using the lower-level modules as demonstrated in the following sections.

One of the properties most often identified as requiring a change is the interrupt priority; this configuration setting is available within the **Properties** window of the associated module. Simply select the indicated module and then view the **Properties** window; the interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Note that the interrupt priorities listed in the **Properties** window in the ISDE will include 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 with 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 with SSP.

Table 2. Configuration Settings for the USBX Host Class Hub Module

ISDE Property	Value	Description
No configurable settings		

In some cases, settings other than the defaults for lower-level modules can be desirable. For example, it might be useful to select different channels for the data transfer driver or the size of the USBX pool memory. The configurable properties for the lower-level stack modules are given in the following sections for completeness and as a reference.

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

5.1 Configuration Settings for the USBX Host Class Hub 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 will be locked to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 3. Configuration Settings for the USBX Host Configuration

ISDE Property	Value	Description
Name	`g_ux_host_0`	Name of the instance

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

Table 4. 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 settings and defaults are for a project using the S7 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 5. 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 settings and defaults are for a project using the S7 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 6. 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 event notification (only valid for USB host)

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

5.2 USBX Host Class Hub 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 **Configuring Clocks** tab or the CGC Interface at run-time.

5.3 USBX Host Class Hub 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 illustrates the method for selecting the pins within the SSP configuration window and the subsequent tables illustrate an example selection for USB pins.

Note: **Operation Mode** selection determines what peripheral signals are available and thus what MCU pins are required.

Table 7. 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 8. 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 S7G2 Synergy MCU Group and the SK-S7G2 Synergy Kit. Other Synergy MCUs and Synergy Kits may have different available pin configuration settings.

Table 9. 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
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
VSS1USBHS	VSS1USBHS	VSS1USBHS Pin
VSS2USBHS	VSS2USBHS	VSS2USBHS Pin

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

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

An application typically does not use the Hub module by itself; other classes (CDC-ACM, Storage, HID, and so on.) are also used at the same time.

The configurator generates processing to register the USBX Host Class Hub module. Specify the same module as **USBX Host Configuration** module registered in the class to be used at the same time.

The following figure shows the stack when registered with the USBX Host Class Mass Storage module at the same time:

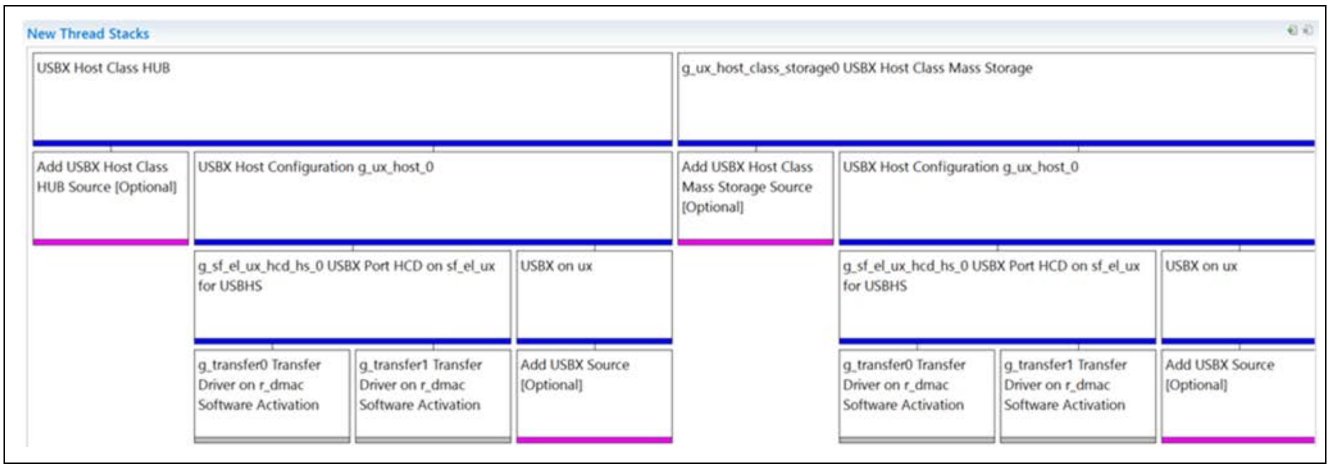

New Thread Stacks
USBX Host Class HUB
g_ux_host_class_storage0 USBX Host Class Mass Storage
Add USBX Host Class HUB Source [Optional]
USBX Host Configuration g_ux_host_0
Add USBX Host Class Mass Storage Source [Optional]
USBX Host Configuration g_ux_host_0
g_sf_el_ux_hcd_hs_0 USBX Port HCD on sf_el_ux for USBHS
USBX on ux
g_sf_el_ux_hcd_hs_0 USBX Port HCD on sf_el_ux for USBHS
USBX on ux
g_transfer0 Transfer Driver on r_dmac Software Activation
g_transfer1 Transfer Driver on r_dmac Software Activation
Add USBX Source [Optional]
g_transfer0 Transfer Driver on r_dmac Software Activation
g_transfer1 Transfer Driver on r_dmac Software Activation
Add USBX Source [Optional]

Figure 3. UXBX Host Class Hub with Mass Storage

7. The USBX Host Class Hub Module Application Project

The application project associated with this module guide demonstrates a full design. You may want to import and open the application project within ISDE and view the configuration settings for the USBX Host Class Hub module. Read over the code in `usb_thread_entry.c` that is used to illustrate the USBX Host Class Hub module APIs in a complete design.

The application project main thread entry periodically gets the key from the USB keyboard and writes it to the specified file on the USB disk; the instances of the mass storage and the keyboard devices are initialized beforehand. The host-event callback handles events such as device insertion or removal. The initialization of mass storage is automatically performed in the generated code; the callback defined in the main thread entry is responsible for HID class only.

The following table identifies the target versions for the associated software and hardware used by the application project.

Table 10. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	6.2.1 or later	Integrated Solution Development Environment
SSP	1.5.0 or later	Synergy Software Platform
IAR EW for Synergy	8.23.1 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	6.2.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.2	Starter Kit

The following diagram illustrates a simple application project flow:

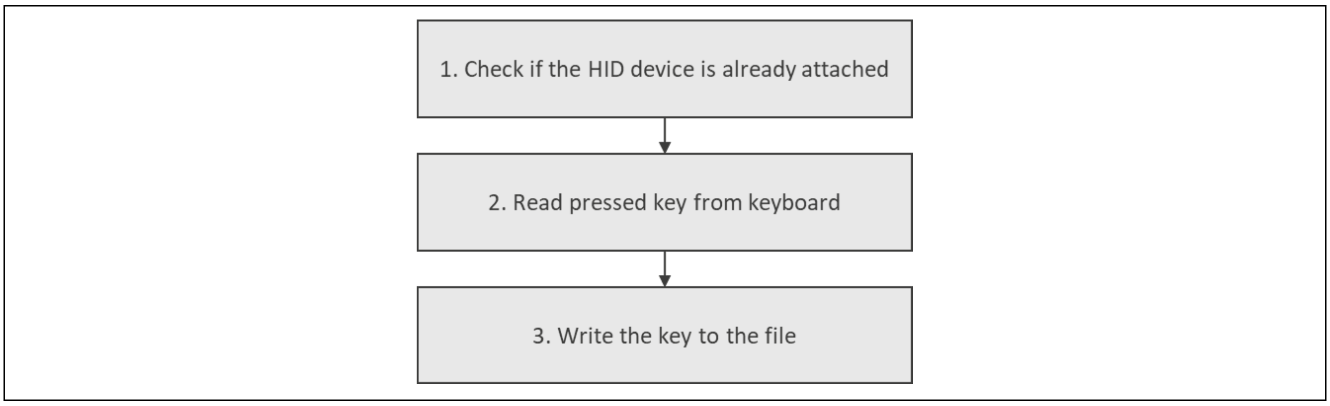

1. Check if the HID device is already attached
2. Read pressed key from keyboard
3. Write the key to the file

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

The `usb_thread_entry.c` file is located in the project once it has been imported into the ISDE; you can open this file within the ISDE and follow along with the description below to help identify key uses of APIs.

The first section of the `usb_thread_entry.c` has the header files that reference the USBX structures and a code section that contains macro constants, global variables, and a function prototype for writing the key to the file. The first macro definition (`KEYBOARD_FILE_NAME`) allows you to specify the file name where pressed keys are stored. The next section is the USBX Host event function, that handles the HID device; it checks if the device has been already inserted or removed. In the first case, the instance of the device is assigned to the global variable and the application waits until the HID status is live. In case of removal, the global variable, previously storing the instance of the HID, is set to null. This callback always returns a code indicating a successful operation has been performed, when called from the generated code. (If the error occurred here, the mass storage device would not be handled properly.) The function definition for writing the key to the file follows: it creates the file (if necessary) and opens it. The current position is moved to the end of the file and the key is written; if it succeeds, the file can be closed, and the data is flushed to the physical media.

Once the application is running and you have connected the hub to the SK-S7G2, on insertion of the mass storage device on the hub, wait for LED3 (RED LED) to light up, indicating that application has successfully completed handshakes with the mass storage device, and it is ready for use. Wait for LED1 (GREEN LED) to light up once the keyboard is inserted, to make sure that device is ready for use.

The application will turn the LED2(AMBER LED) on while accessing the media and turns it off after the media access is completed. You are advised not to disconnect the mass storage when the LED2 is on.

Disconnect the devices one by one from the hub, on successful disconnection the corresponding light associated with the devices is turned off by the application.

The next section is the entry function for the main-program control section. While loop begins and if there is an attached HID keyboard, the application reads the pressed key from the keyboard. If the key has been entered, it is written using the previously defined function. A thread sleep-function pauses execution for a single ThreadX-timer tick and the while loop functions are subsequently repeated.

The application uses 0xFFFFFFFF (TX_WAIT_FOREVER) for Timeout ticks and for Media Initialization setting in FileX on USB Mass Storage Configuration Settings. This results in infinite waiting time for USB Mass Storage device insertion.

A few key properties are configured in this application project to support the required operations and the physical properties of the target board and MCU device. The properties with the values set for this specific project are listed in the following tables; you can also open the application project and view these settings in the **Properties** window as a hands-on exercise.

Table 11. USBX Port HCD for USBHS Configuration Settings for the Application Project

ISDE Property	Value Set
High Speed Interrupt Priority	Priority 3
FIFO size for Bulk Pipes	512 bytes(default)

ISDE Property	Value Set
VBUSEN pin Signal Logic	Active High
Enable High Speed	Enable
Name	g_sf_el_ux_hcd_hs_0
USB Controller Selection	USBHS

Table 12. USBX Configuration Settings for the Application Project

ISDE Property	Value Set
USBX Pool Memory Name	`g_ux_pool_memory`
USBX Pool Memory Size	81920
User Callback for Host Event Notification	`ux_host_event_callback`

Table 13. USBX Host Class HID Configuration Settings for the Application Project

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

Table 14. FileX® on USB Mass Storage Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_fx_media0
Name of FileX Media Control block initialization	fx_media_init_function0
Auto Media Initialization	Enable
Timeout ticks for Media Initialization	0xFFFFFFFF

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

Some configuration settings will normally be changed by the developer from those shown in the application project. For example, you can easily change support for the keyboard, mouse, or remote control for the USBX Host Class HID module.

9. Running the USBX Host Class Hub Module Application Project

To run the USBX Host Class Hub module application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug. Refer to the *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 IAR EW for Synergy, and building/running the application.

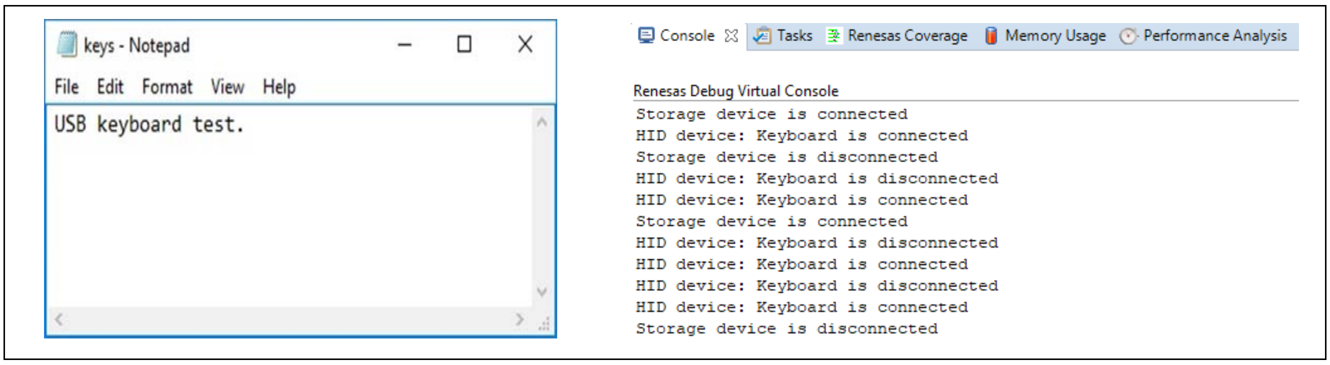
To implement the USBX Host Class Hub module application in a new project, follow the steps for defining, configuring, auto-generating files, adding code, compiling, and debugging on the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical.

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters of the *SSP User's Manual* for a description of how to accomplish these steps.

To create and run the USBX Host Class HUB module application project, follow these steps:

1. Create a new Renesas Synergy project for the S7G2-SK called `USBX_Host_Class_HUB_EL_MG_AP`.
2. Select the **Threads** tab.
3. Add a new thread
   A. Symbol: usb_thread
   B. Name: USB Thread
4. Add the USBX Host Class Hub module.
5. Add the USBX Port HCD for USBHS connected to the USBX Host Configuration module.

6. Add the USBX Host Class HID module.
7. Add the **FileX** on USB Mass Storage module.
8. Use the already added USBX Port HCD for USBHS instance in all required modules.
9. Configure the modules to the preceding tables.
10. Click on the **Generate Project Content** button.
11. Add the code from the supplied project file `usb_thread_entry.c` or copy over the generated `usb_thread_entry.c` file.
12. Connect to the host PC via a micro USB cable to J19 on SK-S7G2 Kit.
13. Connect a USB keyboard and a USB disk to a USB hub connected to J6 on SK-S7G2 Kit.
14. Start to debug the application.
15. The Renesas Debug Virtual Console will provide information about the status of the activities of the hub.
16. The pressed keys on USB keyboard will be saved on the USB disk attached via the USB hub.
17. The keypad output can be viewed in the specified file on the USB disk.



```
USB keyboard test.
```

```
Storage device is connected
HID device: Keyboard is connected
Storage device is disconnected
HID device: Keyboard is disconnected
HID device: Keyboard is connected
Storage device is connected
HID device: Keyboard is disconnected
HID device: Keyboard is connected
HID device: Keyboard is disconnected
HID device: Keyboard is connected
Storage device is disconnected
```

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

Note: Depending on the device to be used, the current capacity may be insufficient in some cases; use a self-powered hub. Check or change your USB stick memory if it did not create a file "keys.txt".

10. USBX Host Class Hub 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 SSP 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 Host Class Hub Module Next Steps

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

12. USBX Host Class Hub Module Reference Information

SSP User Manual: Available at www.renesas.com/us/en/products/synergy/software/ssp.html as a SSP distribution package, and also as a pdf from the Synergy Gallery.

- USBX Host Stack User's Manual
- USBX Device Stack User's Manual

Links to all the most up-to-date USBX Host Class Hub module reference materials and resources are available on the Renesas Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977575.

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	Nov.28.17	—	Initial version
1.01	Feb.22.19	—	Updated steps to run application, added note, and updated software versions.

Corporate Headquarters

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

Trademarks

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/.