Renesas Synergy™ Platform

Communications Framework on USBX™ Module Guide

R11AN0136EU0100
Rev.1.00
Aug 31, 2017

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 should be valuable resources for creating more complex designs.

The Communications Framework on USBX™ is a high-level API for Communications Framework applications, implemented on the USBX Device Class CDC-ACM (Communications Device Class-Abstract Control Model). The Communications Framework uses the USB peripheral on a Synergy MCU device. This module is a replacement for the previous version, which was indicated as deprecated in SSP 1.2.0.

Contents

1. Communications Framework on USBX Module Features

- High-level connectivity is supported on USB but is easily changed to UART and Ethernet connectivity without API modification
- Supports channel locking for exclusive access
- Supports USB high speed (HS) or full speed (FS) operation
- Supports data transfer (DMA or DTC) peripheral operation
- ThreadX®-aware implementation uses mutex and event flags internally.

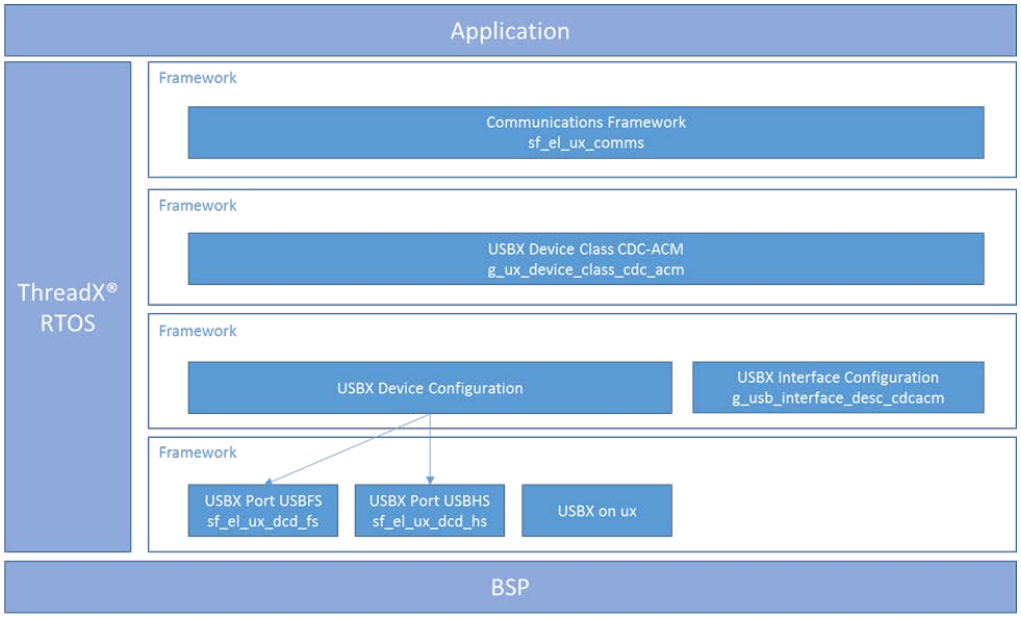

Figure 1 Communications Framework on USBX Module Block Diagram

2. Communications Framework on USBX Module APIs Overview

The communications framework on USBX defines APIs for opening, closing, reading, and writing over the USB connection. A complete list of the available APIs, an example API call, and a short description of each API can be found in the following table. A table of status return values follows the API summary table.

Table 1 Communications Framework on USBX Module API Summary

Function Name	Example API Call and Description
.open	`g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg);` Initialize communications driver.
.close	`g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl);` Clean up communications driver.
.read	`g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout);` Read data from communications driver. This call returns after the number of bytes requested is read or if a timeout occurs while waiting for access to the driver.
.write	`g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout);` Write data to communications driver. This call returns after all bytes are written or if a timeout occurs while waiting for access to the driver.
.lock	`g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout);` Lock the communications driver. Reserve exclusive access to the communications driver.
.unlock	`g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type);` Unlock the communications driver. Release exclusive access to the communications driver.
.versionGet	`g_sf_comms0.p_api->versionGet(&version);` Retrieve the API version in the version pointer.

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures, and function variables, review the *SSP User's Manual* API References for the associated module.

Table 2 Status Return Values

Name	Description
SSP_SUCCESS	Channel opened successfully.
SSP_ERR_IN_USE	Channel already in use.
SSP_ERR_ASSERTION	Pointer to COMM control block or configuration structure is NULL.
SSP_ERR_INTERNAL	Internal error occurs.
SSP_ERR_TIMEOUT	Timeout error.
SSP_ERR_NOT_OPEN	Module is not opened.

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. Communications Framework on USBX Module Operational Overview

The communications framework on USBX provides an easy-to-use connection over the USB port. The high-level APIs in the framework are compatible with other connection implementations (such as UART and Ethernet), facilitating easy switching from one implementation to another without changing APIs. The module uses ThreadX objects like mutex for synchronization for the completion of a transaction. The USBX communication framework module supports the locking functionality, meaning that the user can lock the communication framework to a thread so that multiple threads can use the same USBX port safely. The locking allows the application to reserve a USB port for a given period of time available between a call made to the `lock` API and the `unlock` API. The high-level APIs are used by the `read` and `write` APIs to support receiving or sending data to a host over the USBX CDC-ACM communication interface.

3.1 Communications Framework on USBX Module Operational Notes

The USBX driver can be implemented on either the HS or FS USB peripherals, depending on the version supported by the target MCU.

3.2 Communications Framework on USBX Module Limitations

Refer to the latest *SSP Release Notes* for any additional operational limitations for this module.

4. Including the Communications Framework on USBX Module in an Application

This section describes how to include the Communications Framework on USBX 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 communications framework on USBX to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the communications framework on USBX is `g_sf_comms0`. This name can be changed in the associated Properties window.)

Table 3 Communications Framework on USBX Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 Communications Framework on sf_el_ux_comms	Threads	New Stack> Framework> Connectivity> Communications Framework on sf_el_ux_comms

When the communications framework on USBX on sf_el_ux_comms is added to the thread stack as shown in the following figure, the configurator automatically adds any needed lower-level modules. Any modules that need additional configuration information will have text box 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 includes "Add" in the text. Clicking on any Pink banded modules brings up the "New" icon and then displays the possible choices.

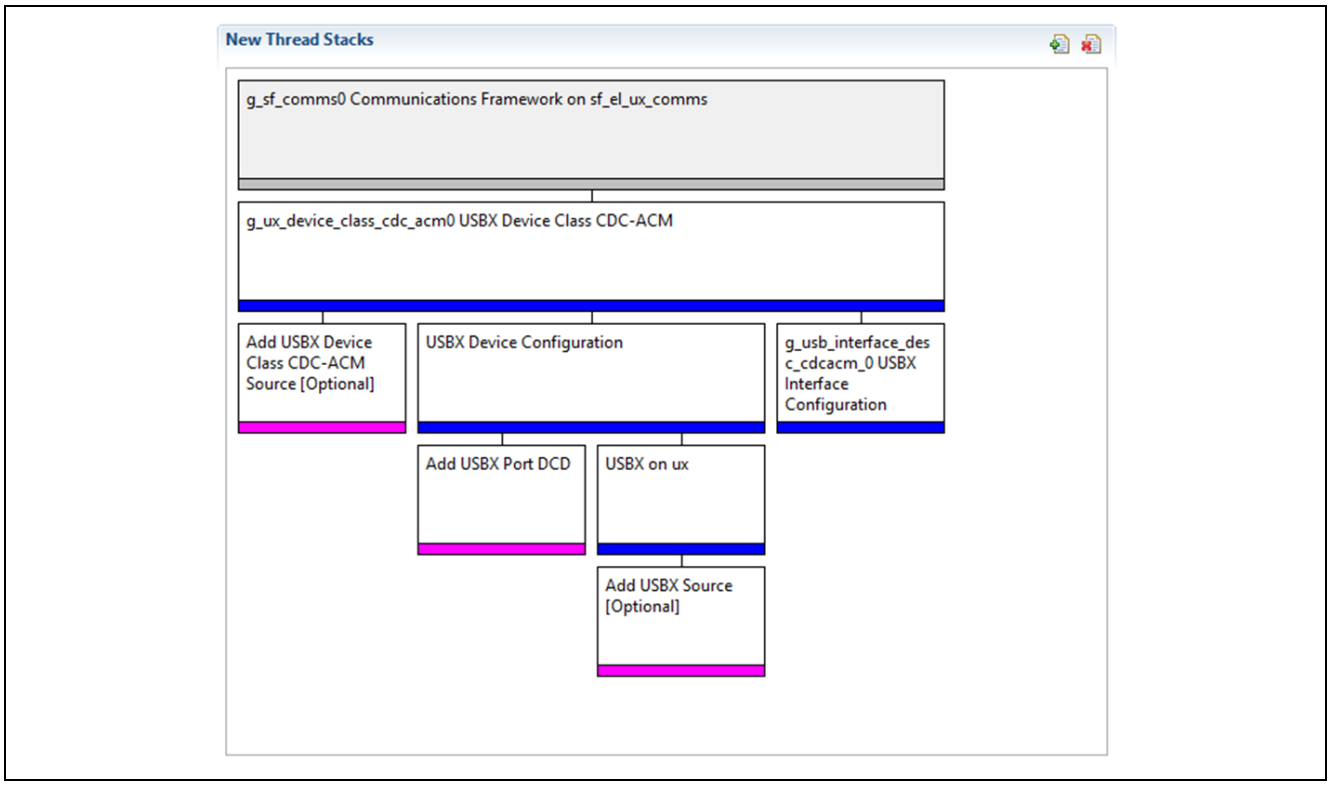

Figure 2 Communications Framework on USBX Module Stack

5. Configuring the Communications Framework on USBX Module

You must configure the communications framework on USBX module for the desired operation. The SSP configuration window will automatically identify (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. Also, only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and are not available for changes, and 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 are shown 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 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. Also note that the interrupt priorities listed in the Properties window in the ISDE 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 4 Configuration Settings for the Communications Framework on USBX Module on sf_el_ux_comms

Parameter	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enables or disables the parameter checking.
Read Input Buffer Size (Bytes)	128	Maximum number of bytes that can be received at a time in the read API.

Parameter	Value	Description
Timeout in ticks	1000	Timeout value to suspend a USBX CDC instance creation in the open API.
Name	g_sf_comms0	Module name.
Name of the sf_comms initialization function	sf_comms_init0	Name of helper function to initialize Communications Framework. The function will be presented in the auto-generated code in the `<xxx_thread>.c`, where `<xxx_thread>` is the name of your thread symbol given to the Thread property. The function is to be called in the auto-generated code if Auto sf_comms Initialization property is enabled. If disabled, the function can be called in the user application.
Auto sf-comms Initialization	Enable, Disable (Default: Enable)	Auto Initialization support of communications framework. The helper function above is called in the auto-generated code if this configuration is enabled. Otherwise, the function is not called automatically and the user can call it sometime later.

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

In some cases, settings other than the default values for stack modules can be desirable. For example, it might be useful to select different buffer sizes depending on the application. The configurable properties for the lower-level stack modules are given in the following sections for completeness and as a reference.

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

5.1 Configuration Settings for the Communications Framework on USBX Lower-Level Drivers

Table 5 Configuration Settings for the USBX Device Class CDC-ACM

ISDE Property	Value	Description
Name	g_ux_device_class_cdc_acm0	Module name
USBX CDC-ACM instance_activate Function Callback	ux_cdc_device0_instance_activate	USBX CDC-ACM instance_activate Function Callback selection
USBX CDC-ACM instance_deactivate Function Callback	ux-cdc_device0_instance_deactivate	USBX CDC-ACM instance_deactivate Function Callback selection

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

Table 6 Configuration 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	Device, Communications (CDC), HID, Mass Storage, Miscellaneous, Vendor Specific (Default: Communications) (CDC)	Class Code selection

ISDE Property	Value	Description
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 (Modify this value only for Vendor-specific Class, otherwise set zero) selection
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero) selection
Self-Powered	Enable, Disable (Default: Enable)	Self-Powered selection
Remote Wakeup	Enable, Disable (Default: Disable)	Remote Wakeup selection
Maximum Power Consumption (in 2 mA units)	50	Maximum Power Consumption (in 2 mA units) 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 (US English is applied if zero is set) selection

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

Table 7 Configuration Settings for the USBX Interface Configuration

ISDE Property	Value	Description
Name	g_usb_interface_desc_cdcacm_0	Module name
Interface Number of Communications Class interface	0x00	Interface Number of Communications Class interface selection
Interrupt Transfer endpoint to use for Communications Class	Endpoint 1-9 (Default: Endpoint 3)	Interrupt Transfer endpoint to use for Communications Class selection
Polling period for Interrupt Endpoint (in ms/125 µs units for FS/HS)	0x0F	Polling period for Interrupt Endpoint (in ms/125 µs units for FS/HS) selection
Interface Number of Data Class interface	0x01	Interface Number of Data Class interface selection
Bulk In Transfer endpoint to use for Data Class	Endpoint 1-9 (Default: Endpoint 1)	Bulk In Transfer endpoint to use for Data Class selection

ISDE Property	Value	Description
Bulk Out Transfer endpoint to use for Data Class	End (Default: Endpoint 2)	Bulk Out Transfer endpoint to use for Data Class selection
Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface)	0x00	Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface) selection
Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface)	0x00	Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface) selection

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

Table 8 Configuration Settings for the USBX Port DCD 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.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	USB controller selection.

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

Table 9 Configuration Settings for the USBX Port DCD 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.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	USB controller selection.

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

Table 10 Configuration Settings for the USBX on ux

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Module name.
USBX Pool Memory Size	18,432	USBX pool memory size selection.
User Callback for Host Event Notification (Only valid for USB Host)	(Default: NULL)	Callback function to handle Host Event Notification.

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

5.2 Communications Framework on USBX Module Clock Configuration

The USB peripheral module uses UCLK as the driver and it should be set to 48 MHz in the Clocks tab of the SSP configurator.

5.3 Communications Framework on USBX 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 table illustrates an example selection for the USBFS0 pins.

Note: The USBHS pins are similar to those shown in the following table, so they are not included.

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

Table 11 Pin Selection for USBFS0

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

Note: The selection sequence assumes USBFS0 is the desired hardware target for the driver.

Table 12 Pin Configuration Settings for USBFS0

Property	Settings	Description
Operation Mode	Disabled, Custom, Device, Host, OTG (Default: Disabled)	Select Device for Communications Framework
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None, P204, P205 (Default: None)	OVRCURB pin
OVRCURA	None, P205, P501 (Default: None)	OVRCURA pin
VBUSEN	None, P206, P500 (Default: None)	VBUSEN pin
VBUS	None, P406 (Default: None)	VBUS pin
EXICEN	None, P409, P503 (Default: None)	EXICEN pin
ID	None, P408, P504 (Default: None)	ID pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

6. Using the Communications Framework on USBX Module in an Application

The typical steps in using the Communications Framework on USBX in an application are:

1. Initialize the Communications Framework on USBX using the `open` API.
2. Lock the channel for continuous communications using the `lock` API if needed.
3. Receive data using the `read` API.
4. Send data using the `write` API.
5. Unlock the channel from continuous communication using the `unlock` API if needed.
6. Close the channel using the `close` API.

These steps are illustrated in a typical operational flow diagram in the following figure:

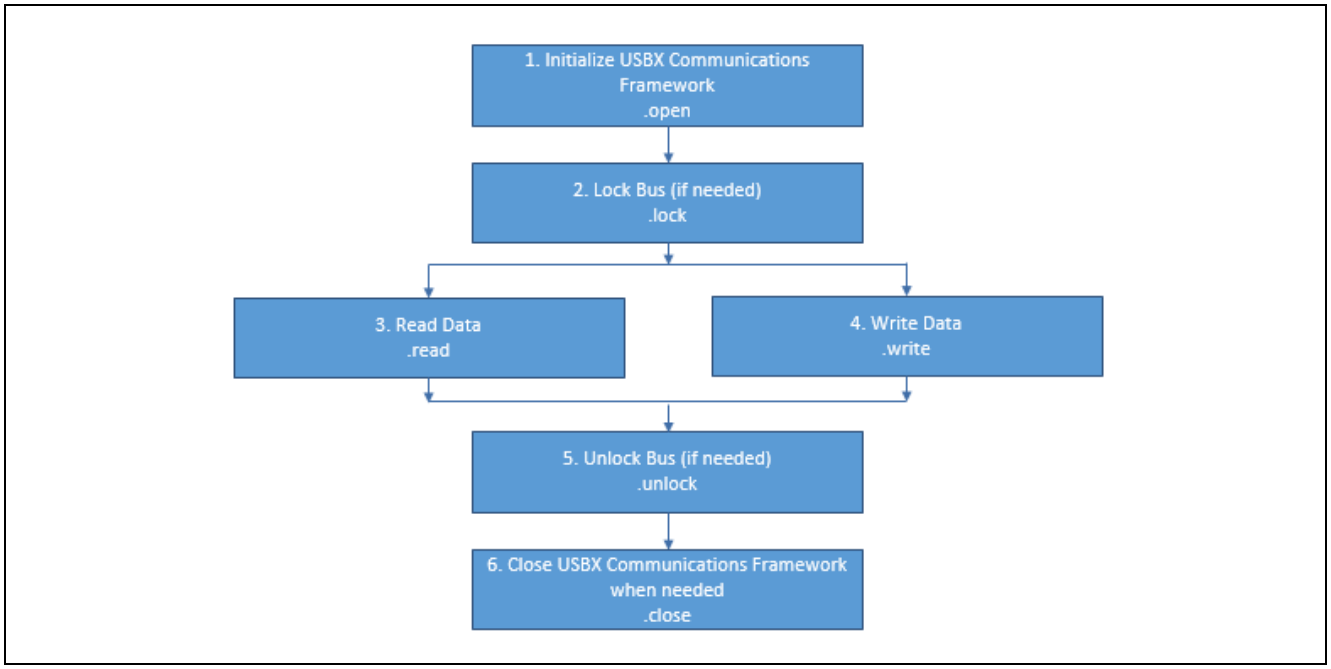

Figure 3 Flow Diagram of a Typical Communications Framework on USBX Application

7. The Communications Framework on USBX Module Application Project

The application project associated with this module guide demonstrates the aforementioned steps in a full design. The project can be found using the link provided in the References section at the end of this document. You may want to import and open the application project within the ISDE and view the configuration settings for the Communications Framework module. You can also read over the code (in `usb_thread_entry.c`) used to illustrate the Communications Framework APIs in a complete design.

The application project demonstrates the typical use of the Communications Framework APIs. The application project USBX thread entry initializes the Communications Framework on USBX, sends a welcome message through the framework, and listens for user input. Each time the user presses a proper key (1, 2, or 3), a corresponding LED toggles, and all LED statuses are sent in a message over the framework. A terminal application is required to handle communication with the board and display messages, such as the Tera Term terminal, for instance.

Table 13 Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	5.3.1 or later	Integrated Solution Development Environment
IAR EW for Synergy	7.71.2 or later	IAR Workbench IDE for Synergy platform
SSP	1.2.0 or later	Synergy Software Platform
SSC	5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following figure shows a simple flow diagram of the application project:

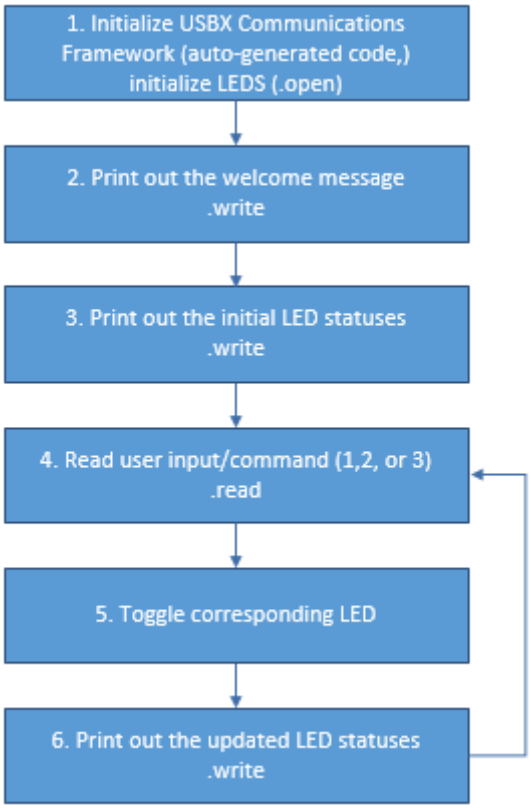

Figure 4 Communications Framework Application Project Flow Diagram

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 provided to help identify key uses of APIs.

The first section of `usb_thread_entry.c` has the header files which reference the Communications Framework instance structure and a code section with function prototypes (for toggling LED pin levels and generating messages sent through the framework) and global variables (LED pin levels). The next section is the entry function for the main program-control section. At first, the LED information structure is initialized and all LEDs are turned off. Then, there is the 1000-tick thread sleep to make sure the board as the USB device is properly set up. After the sleep, the application sends the welcome message and the LED status message through the Communications Framework using the `write` API. The program enters an infinite loop afterwards and in that loop, a key is read using the Communications Framework `read` API. The input is then parsed and translated into a command to toggle LED 1, 2, or 3 in response to receiving keys 1, 2 or 3, respectively. After an LED is toggled, the application sends an LED status message using the `write` API.

The next sections are the functions to toggle LED levels, update LED pin levels, and generate and send LED status messages.

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. 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 14 Communications Framework Configuration Settings for the Application Project

ISDE Property	Value Set
Parameter Checking	Default (BSP)
Read Input Buffer Size (Bytes)	128
Timeout in ticks	1000
Name	g_sf_comms
Name of the sf comms initialization function	sf_comms_init
Auto sf comms Initialization	Enable

Table 15 USBX Device Class CDC-ACM Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_ux_device_class_cdc_acm0
USBX CDC-ACM instance_activate Function Callback	ux_cdc_device0_instance_activate
USBX CDC-ACM instance_deactivate Function Callback	ux_cdc_device0_instance_deactivate

Table 16 USBX Device Configuration Settings for the Application Project

ISDE Property	Value Set
Vendor ID	0x045B
Product ID	0x0238 (Windows 7 and Windows 8 — RFP driver installed) 0x0000 (Windows 10 — RFP driver installed) 0x0238 (Windows 10 — No RFP driver installed) See note following this table.
Device Release Number	0x0000
Index of Manufacturing String Descriptor	0x00
Index of Product String Descriptor	0x00
Index of Serial Number String Descriptor	0x00
Class Code	Miscellaneous. See the note following this table.
Index of String Descriptor describing this configuration	0x00
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00
Self-Powered	Enable
Remote Wakeup	Disable
Maximum Power Consumption (in 2 mA units)	50
Supported Language Code	0x0409
Name of USBX String Framework	NULL
Total index number of USB String Descriptors in USB String Framework	0
Name of USBX Language Framework	NULL
Number of Languages to support (US English is applied if zero is set)	0

Note: On Microsoft® Windows® 7 and Microsoft® Windows® 8 machines, a signed driver is required. The Renesas Flash Programmer (RFP) driver is used for this purpose. Download this driver from the Renesas website. Set the class code to miscellaneous.

On a Microsoft® Windows® 10 machine, a signed driver is not required as Windows uses the class and sub-class to determine which driver to use. In this case, set the product ID to 0x0238. If the RFP driver is installed, set the product ID to 0x0000, so the RFP driver does not take precedent over the Windows 10 Inbox driver.

Table 17 USBX Interface Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_usb_interface_desc_cdcacm_0
Interface Number of Communications Class interface	0x00
Interrupt Transfer endpoint to use for Communications Class	Endpoint 3
Polling period for Interrupt Endpoint (in ms/125 µs units for FS/HS)	0x0F
Interface Number of Data Class interface	0x01
Bulk In Transfer endpoint to use for Data Class	Endpoint 1
Bulk Out Transfer endpoint to use for Data Class	Endpoint 2
Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface)	0x00
Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface)	0x00

Table 18 USBX Port DCD on sf_el_ux for USBFS Configuration Settings for the Application Project

ISDE Property	Value Set
Full Speed Interrupt Priority	Priority 2
Name	g_sf_el_ux_dcd_fs_0
USB Controller Selection	USBFS

Table 19 USBX on ux Configuration Settings for the Application Project

ISDE Property	Value Set
USBX Pool Memory Name	g_ux_pool_memory
USBX Pool Memory Size	18,432
User Callback for Host Event Notification (Only valid for USB Host)	NULL

8. Customizing the Communications Framework on USBX Module for a Target Application

As an alternative, USBHS can be used. Also, the Class Code, Vendor and Product IDs may be changed accordingly to your project's needs. A locking mechanism could be used in case the user would need two or more threads to call `read` and `write` APIs.

9. Running the Communications Framework on USBX Module Application Project

To run the USBX Communications Framework application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug.

Note: The following steps provide 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* available as described in the References section at the end of this document.

1. Refer to the *Importing a Renesas Synergy Project* (11an0023eu0116-synergy-ssp-import-guide.pdf) included in this package for instructions on importing the project into e^2 studio ISDE or the IAR EW for Synergy and building/running the application
2. Connect to the host PC through a micro USB cable to J19 on the SK-S7G2 board.
3. Connect to the host PC through a micro USB cable to J5 on the SK-S7G2 board.
4. Start to debug the application.
5. Start the Tera Term application and create a connection on the appropriate serial port (COM5, for example). Keep the connection open in Tera Term for further debug sessions. During very first debug session, the welcome message may not be displayed.
6. Press 1, 2 or 3 on the PC keyboard. The output can be viewed in the Tera Term application as shown in the following figure.

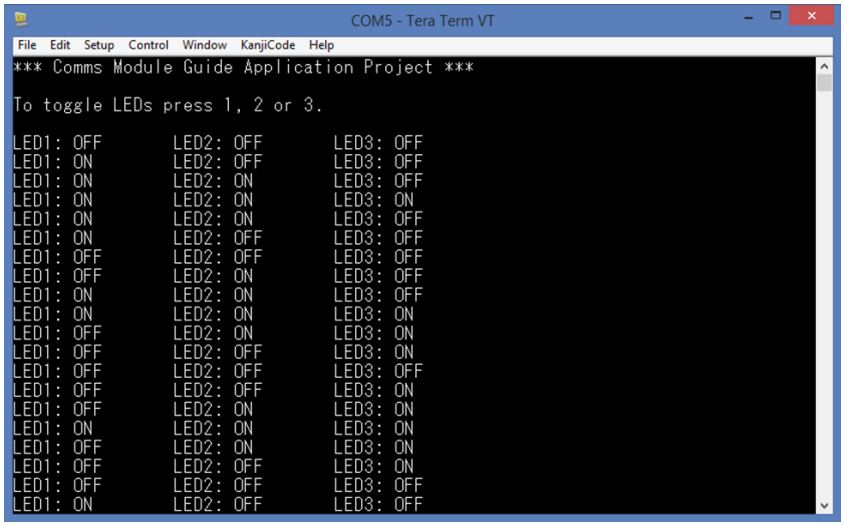

Figure 5 Example Output from Communications Framework on USBX Application Project

10. Communications Framework on USBX Module Conclusion

This module guide has provided 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. Communications Framework on USBX Module Next Steps

After you have mastered a simple Communications Framework on USBX project, you may want to review alternative Communications Framework implementations on NX or UART.

12. Communications Framework on USBX 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 sf_el_ux_comms module reference materials and resources are available on the Synergy Knowledge Base: https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/SF_EL_UX_Comms_Module_Guide_Resources.

Website and Support

Support: https://synergygallery.renesas.com/support

Technical Contact Details:

- America: https://renesas.zendesk.com/anonymous_requests/new
- Europe: https://www.renesas.com/en-eu/support/contact.html
- Japan: https://www.renesas.com/ja-jp/support/contact.html

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Aug 31, 2017		Initial version

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 disputes 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, drawing, chart, program, algorithm, 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 otherwise misappropriate 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, copy or otherwise misappropriation of Renesas Electronics products.
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; and 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.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the 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 radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use 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 not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
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.

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

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

(Rev.3.0-1 November 2016)

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

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

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

Colophon 6.0