Renesas Synergy™ Platform

USBX™ Host Class CDC-ACM Module Guide

Introduction

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

The USBX Host Class CDC-ACM module is a high-level API for USBX Host Class CDC-ACM applications and is implemented on **g_ux_host_class_cdc_acm**. The USBX Host Class CDC-ACM module configures the USBX Host Class CDC-ACM Source, USBX Host Configuration, USBX Source, USBX Port HCD, and a transfer driver. The USBX Host Class CDC-ACM module uses the DMAC/DTC and USB Host Class peripherals on the Synergy MCU.

This overview covers the key elements related to the USBX Host Class CDC-ACM module implementation on the Renesas Synergy™ Platform. Its primary focus is the addition and configuration of the USBX Host Class CDC-ACM module to a Renesas Synergy™ Platform project. For more details on the operation of this module, consult the *USBX User's Guide* for the Renesas Synergy Platform document. This user's guide is part of X-Ware™ Component Documents for Renesas Synergy™ zip file available from the Renesas Synergy Gallery (www.renesas.com/synergy/ssp).

Contents

1. Features 2
2. Overview 2
3. Operational Overview 3
3.1 Important Operational Notes and Limitations 4
4. USBX Host Class CDC-ACM Module in an Application 4
5. Configuring the USBX Host Class CDC-ACM Module 5
5.1 Configuration Settings for the Low-Level Modules 6
5.2 Clock Configuration 9
6. Using the USBX Host Class CDC-ACM Module in an Application 9
7. The USBX Host Class CDC-ACM Module Application Project 10
8. Customizing for a Target Application 12
9. Running the USBX Host Class CDC-ACM Module Application Project 12
10. Conclusion 13
11. Next Steps 13
12. Reference Information 13

1. Features

The CDC-ACM class uses a composite device framework to group interfaces (control and data). As a result, care should be taken when defining the device descriptor. The USBX relies on the IAD descriptor to know internally how to bind interfaces. The IAD descriptor should be declared before the interfaces and contain the first interface of the CDC-ACM class and how many interfaces are attached. The CDC-ACM class also uses a union functional descriptor which performs the same function as the newer IAD descriptor. Although a union functional descriptor must be declared for historical reasons and compatibility with the host side, it is not used by the USBX.

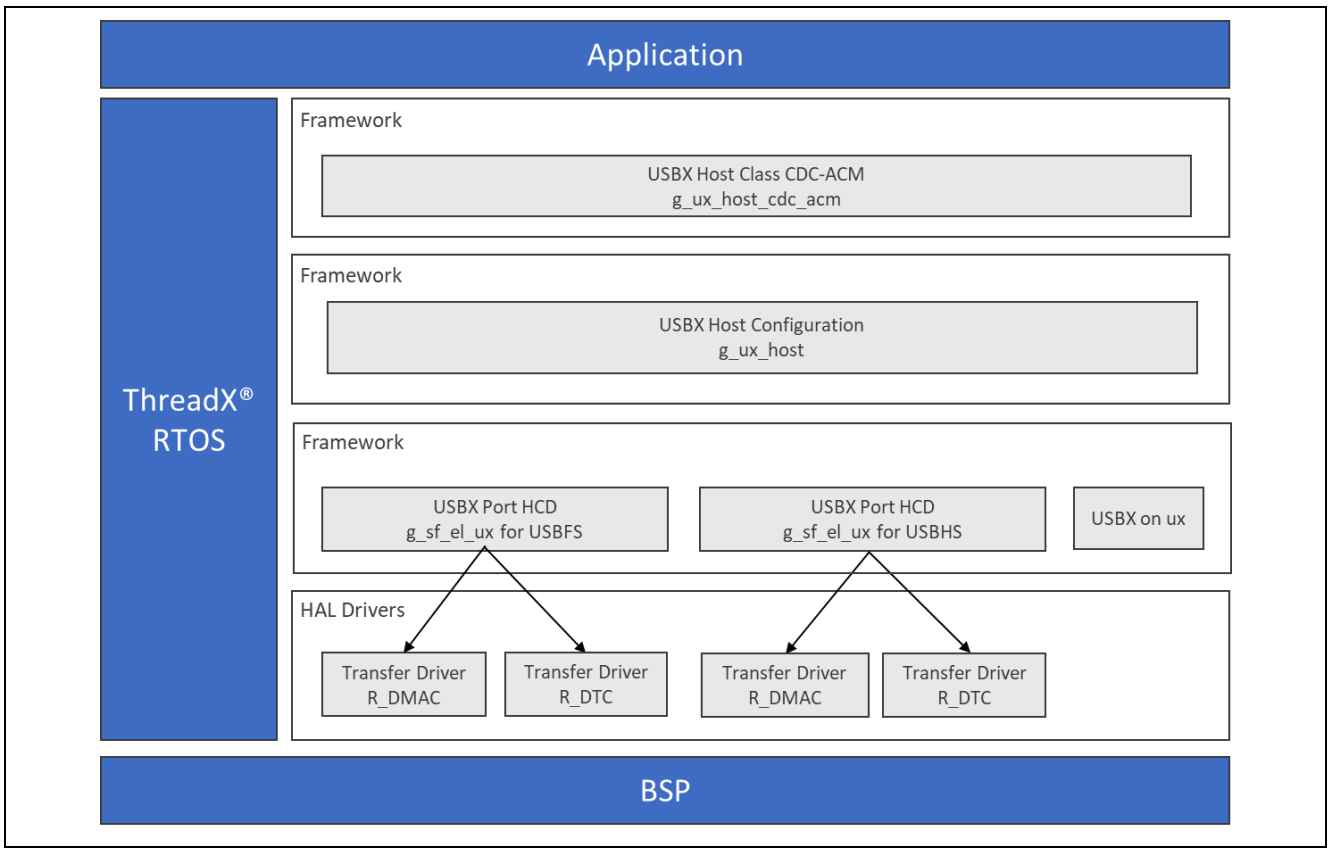

Figure 1. USBX Host Class CDC-ACM Module Organization, Options and Stack Implementations

2. Overview

The USBX Host Class CDC-ACM Module defines APIs for reading, writing, and IOCTL. A complete list of the available APIs, an example API call, and a short description of each can be found in the following table. A table of status return values follows the API summary table below.

Table 1. USBX Host Class CDC-ACM Module API Summary

Function Name	Example API Call and Description
ux_host_class_cdc_acm_read	`status = ux_host_class_cdc_acm_read(cdc_acm, data_pointer, requested_length,&actual_length);` This function reads from the cdc_acm interface. The call is blocking and only returns when there is either an error or when the transfer is complete.
ux_host_class_cdc_acm_write	`status = ux_host_class_cdc_acm_write(cdc_acm, data_pointer,requested_length,&actual_length);` This function writes to the cdc_acm interface. The call is blocking and only returns when there is either an error or when the transfer is complete.

Function Name	Example API Call and Description
ux_host_class_cdc_acm_ioctl	`status = ux_host_class_cdc_acm_ioctl(cdc_acm, ioctl_function, &parameter_p);` This function performs a specific IOCTL function to the cdc_acm interface. The call is blocking and only returns when there is either an error or when the command is completed.

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures and function variables review the associated *Express Logic User's Manual* accessible as described in the References section later in this document.

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 *SSP User's Manua*l API References for the associated module and refer to the definition of all relevant status return values.

3. Operational Overview

Initialization of USBX resources

USBX has its own memory manager. The memory needs to be allocated to USBX before the host or device side of USBX is initialized. 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 Host class is USB CDC ACM.

Definition of Host Classes

It is required to define one or more host classes with 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 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 initialization of the interface.

3.1 Important Operational Notes and Limitations

3.1.1 Operational Notes

- Use a class container for the USBX Host Class CDC-ACM obtained by auto-generated code to get a CDC-ACM instance.
- Poll the flag **ux_host_class_cdc_acm_state** in the instance and make sure the status is live.
- Check if a DATA class interface is available in the CDC-ACM instance.
- Set up the CDC-ACM reception if required.
- Perform CDC-ACM communication with USB device.

3.1.2 Limitations

- The module needs the interrupt of a USB Controller to be enabled.
- See *SSP Release Notes* (sf_el_ux sections) for known limitations.
- The module uses the interrupt of a USB Controller. Set the appropriate interrupt priority level in Synergy Configuration tool. By default, it is disabled, and in that state does not work.
- The module uses the interrupt of a Transfer module (implemented as DMAC or DTC) if it is used. Set appropriate priority level in the Synergy Configuration tool. The level must be higher than a USB Controller to work properly.
- A zero-length packet does not contain the received data, but the application must perform the receive operation.
- Refer to the most recent *SSP Release Notes* for any additional operational limitations for this module.

4. USBX Host Class CDC-ACM Module in an Application

This section describes how to include the USBX Host Class CDC-ACM module in an application using the SSP configurator.

Note: This section assumes 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 CDC-ACM 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 CDC-ACM module is `g_ux_host_class_cdc_acm0`. This name can be changed in the associated Properties window.)

Table 3. USBX Host Class CDC-ACM Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_device_class_cdc_acm0 USBX Host Class CDC-ACM	Threads	New Stack> X-Ware> USBX> Device> Classes > CDC-ACM > USBX Host Class CDC-ACM

When the USBX Host Class CDC-ACM module added to the thread stack as shown in the following figure, the configurator automatically adds the needed lower-level drivers. Any drivers that need additional configuration information are 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), and need to be added only once, since they can be used by multiple stacks. Modules with a Pink band can require the selection of lower-level drivers. Sometimes these modules are optional (or recommended), and they are indicated in the block with the inclusion of this text. If lower-level drivers need to be added, the module description includes "Add" in the text. Clicking on any Pink banded modules brings up the "New" icon and shows alls the possible choices.

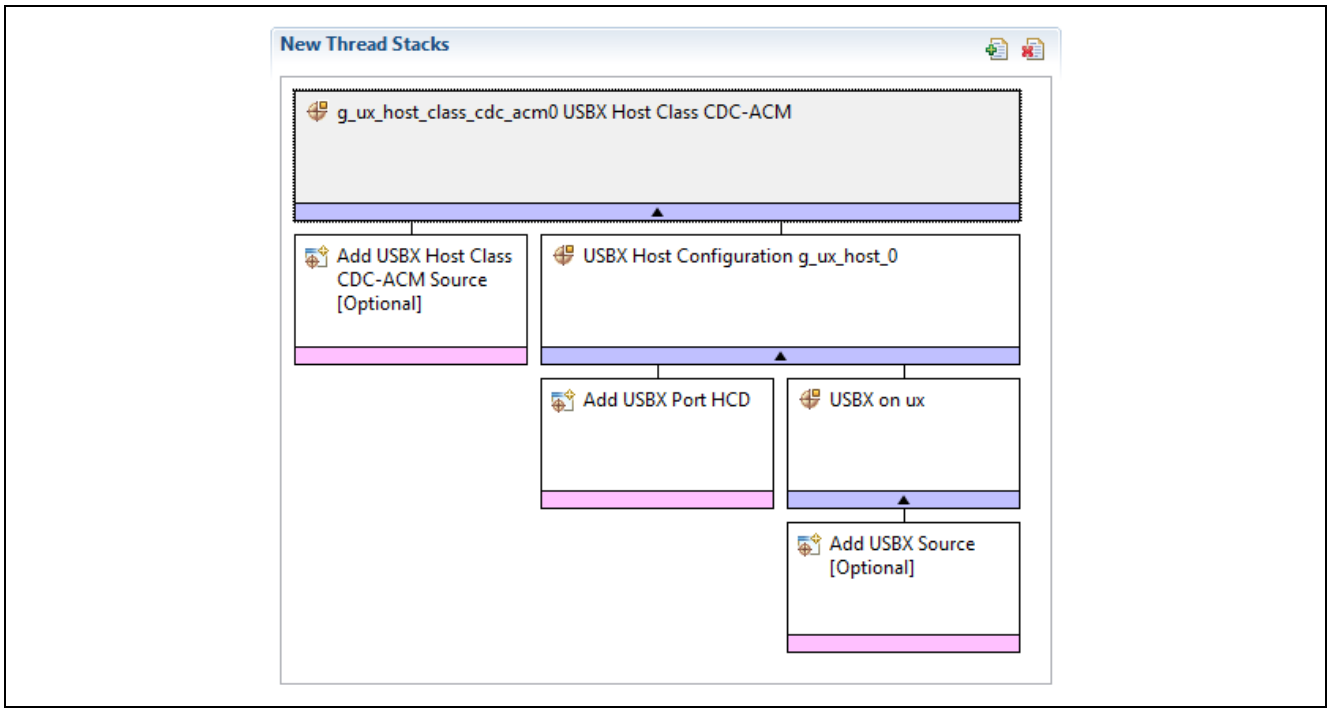

Figure 2. USBX Host Class CDC-ACM Module Stack

5. Configuring the USBX Host Class CDC-ACM Module

The USBX Host Class CDC-ACM 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 not available for changes. Locked properties 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 easy reference.

One of the properties most often identified as requiring a change is the interrupt priority; this configuration setting is available in the **Properties** window of the USBx Port DCD for **sf_el_ux** for USBHS module. Simply select the indicated module and then view the **Properties** window. Note that the interrupt priorities listed in the ISDE **Properties** window indicate the validity of the setting, based on the MCU targeted (CM4 or CM0+). This level of detail is not included in the following configuration properties tables but is easily visible within the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE, create the USBX Host Class CDC-ACM module and explore the property settings, while looking over the following configuration table settings. 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 USBX Host Class CDC-ACM Module

ISDE Property	Value	Description
Name	g_ux_host_class_cdc_acm0	Name of the Instance

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

In some cases, settings other than the defaults for lower-level modules may be desired. For example, it might be useful to select different channels for the data-transfer driver or adjust the size of the USBX pool memory. The configurable properties for the lower-level stack modules are given in the following sections as a complete 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 Low-Level Modules

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

Table 5. Configuration Settings for the USBX Host Configuration Driver

ISDE Property	Value	Description
Name	g_ux_host_0	Name of the instance

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

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

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15 (lowest, not valid if using Thread X), Disabled Default: Disabled	Set the interrupt priority for the USBFS peripheral for the value range for target CPU core. Note that the interrupt priority of DMAC or DTC has to be greater than this setting if Transfer Module is used for data transfer.
VBUSEN pin Signal Logic	Active High, Active Low Default: Active High	Specify the VBUSEN pin control logic. The logic level specified here is outputted to the pin.
Name	g_sf_el_ux_hcd_fs_0	USB Synergy Port module instance. Must be a valid C symbol.
USB Controller Selection	USBFS	It is fixed setting that users cannot change.

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

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

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14, 15 (lowest, not valid if using Thread X), Disabled Default: Disabled	Set the interrupt priority for USBHS peripheral for the value range for target CPU core. Note that the interrupt priority of DMAC or DTC has to be greater than this setting if Transfer Module is used for data transfer.
FIFO size for Bulk Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	Specify the FIFO size for Bulk-IN and Bulk-OUT Pipes. Setting larger value contributes to get the USB data throughput improved.
VBUSEN pin Signal Logic	Active High, Active Low Default: Active High	Specify the VBUSEN pin control logic. The logic level specified here is outputted to the pin. In case “S7G2-SK” is selected as target board on the BSP tab in Synergy Configuration tool, the logic level is forced Low.
Enable High Speed	Enable, Disable Default: Disable	Normally set Enable. Only change the setting to Disable if forcing the USBHS peripheral run at Full-Speed.
Name	g_sf_el_ux_hcd_hs_0	USB Synergy Port module instance. Must be a valid C symbol.
USB Controller Selection	USBHS	It is fixed setting and users cannot change the setting.

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

Table 8. Configuration Settings for the USBX on ux

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.
USBX Pool Memory Size	18KB	Required memory size for CDC ACM Class.
User Callback for Host Event Notification (only valid for USB Host)	NULL	User callback for host event notification selection
Name of generated initialization function	ux_common_init0	Initialization function name
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function

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

Table 9. Configuration Settings for the TX Transfer Driver on r_dmac Software Activation

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

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

Table 10. Configuration Settings for the TX Transfer Driver on r_dtc Software Activation 1

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in	Source	Repeat area selection

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

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

Table 11. Configuration Settings for the RX Transfer Driver on r_dmac Software Activation

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

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

Table 12. Configuration Settings for the RX Transfer Driver on r_dtc Software Activation 1

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

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

5.2 Clock Configuration

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

6. Using the USBX Host Class CDC-ACM Module in an Application

The configurator initiates processing to create and register the USBX Host Class CDC-ACM module; communication must be done after the device is connected to the host.

The typical steps in using the USBX Host Class CDC-ACM module in an application are:

1. Get the first instance of the connected device with `ux_host_stack_class_instance_get` API.
2. Wait for the device status to become live.
3. Check that the class of the device is CDC Data Class.
4. If there is the next device, check the status again.
5. For received data reading, use the `ux_host_class_cdc_acm_read` API.
6. For data sending, use the `ux_host_class_cdc_acm_write` API.

These common steps are shown in the following operational flow diagram:

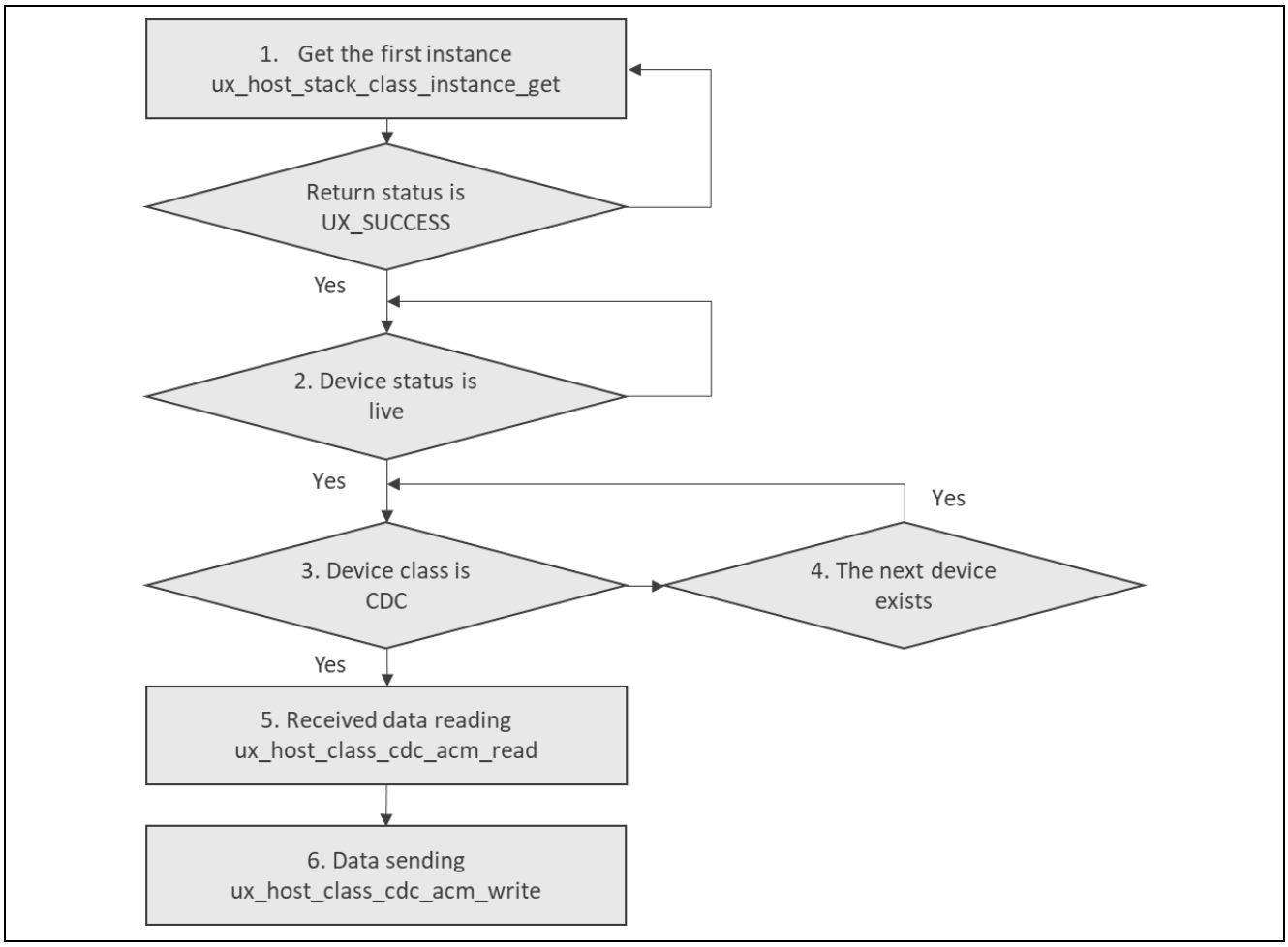

Figure 3. Flow Diagram of a Typical USBX Host Class CDC-ACM Module Application

7. The USBX Host Class CDC-ACM Module Application Project

The application project associated with this module guide demonstrates the steps in a full design. The project can be found as described 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 USBX Host Class CDC-ACM module. You can also read over the code (in `usb_cdc_acm_host_thread_entry.c`) which is used to illustrate the USBX Host Class CDC-ACM module APIs in a complete design. And, the USBX Host Class CDC-ACM communicates with USB device CDC-ACM. A typical device that can be used to test the application is a USB modem, but you can use other boards running Synergy USBX Device CDC-ACM.

The application project demonstrates the typical use of the USBX Host Class CDC-ACM module APIs. The application project main thread entry waits for the connection from the callback function, sends **AT** string to device. After this, the application receives an echo back, response, and zero length packet from device. Sends from host and receives from device are repeated until the device is unplugged. The user-callback function obtains a pointer of device instance and notifies the main thread of the plugged or unplugged state. 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
e² studio	7.3.0 or later	Integrated Solution Development Environment
SSP	1.6.0 or later	Synergy Software Platform
IAR EW for Synergy	8.23.3 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	7.3.0 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following illustration shows a simple flow diagram of the application project.

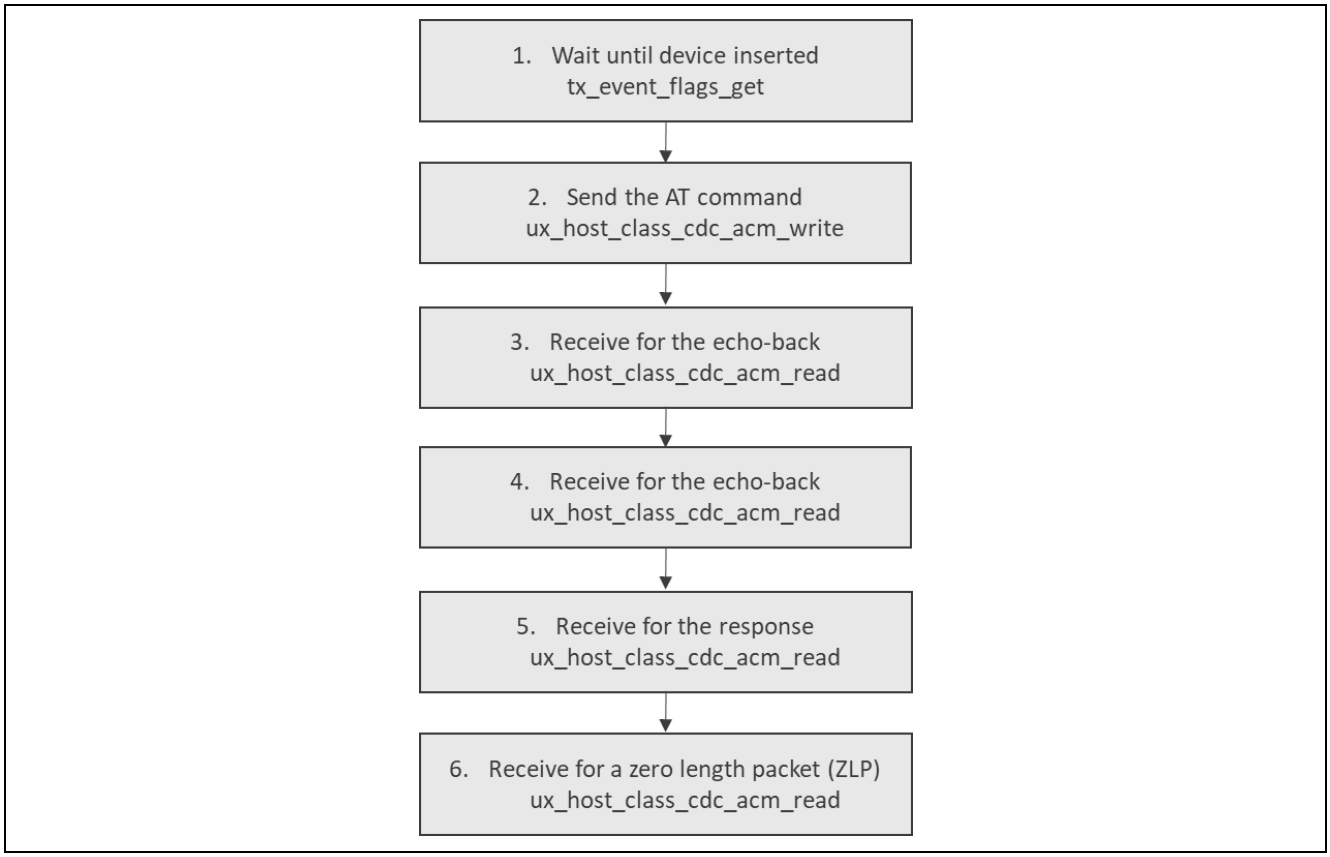

Figure 4. USBX Host Class CDC-ACM Module Application Project Flow Diagram

The first section of usb_cdc_acm_host_thread_entry.c has a header file for the user thread and the define which allows semi-hosting to display results using printf(). The next section has the definitions and variables used in the application, followed by a callback function used when the USB device is connected or disconnected. The last section is the user thread to send and receive to the USB CDC ACM device.

When connected to USB device, `ux_host_usr_event_notification` is called. This callback function determines the connected device from the argument, saves the pointer of CDC ACM device instance, and notifies the user thread. This callback function is also called when the USB device was unplugged. The user thread receives notification from the callback function and starts access to the USB CDC ACM device. The user thread sends "AT" string to device and waits for receives of echo-back string and response string. Finally, wait for zero length packet reception. The printf function displays the short message and the string from CDC-ACM device.

Note: It is assumed that you are familiar with using `printf()` the Debug Console in the Synergy Software Package. If you are unfamiliar with this, refer to the *How do I Use Printf() with the Debug Console in the Synergy Software Package,* a Knowledge Base article, available as described in the References section at the end of this document. Alternatively, you can see results via the watch variables in the debug mode.

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 table. You can also open the application project and view these settings in the Properties window as a hands-on exercise.

Table 14. USBX Host Class CDC-ACM Module Configuration Settings for the Application Project

ISDE Property	Value Set
Add USBX Port HCD	Name: g_sf_el_ux_hcd_hs_0
USBX Port HCD on sf_el_ux for USBHS High Speed Interrupt Priority	Priority 3
VBUSEN pin Signal Logic	Active Low
USBX on ux USBX Pool Memory Size	32768
User Callback for Host Event Notification (Only valid for USB Host)	ux_host_usr_event_notification
Add the event flag to application thread	Name: CDCACM Activate Flags Symbol: g_cdcacm_activate_event_flags0

8. Customizing for a Target Application

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

9. Running the USBX Host Class CDC-ACM Module Application Project

To run the USBX Host Class CDC-ACM module application project and see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug.

Note: The below 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 application project simply follow these steps:

1. Import and build the example project included with this module guide according to the Synergy Project Import Guide r11an0023eu0121-synergy-ssp-import-guide.pdf.
2. If you use a device that returns response like a USB modem as a test device, please define 'RECEIVE_RESPONSE' and build it.
3. Connect to the host PC using the USB cable. (For SK-S7G2 board which performs the functionality of host, use J19 DEBUG_USB connector)
4. If running on the DK-S3A7, set the USBF switch to OFF on the S6 dip switch.
5. Start to debug the application.
6. Load USB CDC-ACM device application on second synergy board (e.g. SK-S7G2, DK-S3A7 etc.). This board acts as device. (Use USBX_CDCACM_Device application for any board available from synergy gallery)
7. Connect the USB CDC-ACM device board to host board using the USB cable. (e.g. For SK-S7G2 board, use J6 USB connector)
8. The output can be viewed in the ISDE Debug Console. (Here is when USBX Device CDC-ACM Module Guide application used as device)

```
send_count: 1
echo-back : AT

send_count: 2
echo-back : AT

send_count: 3
echo-back : AT

send_count: 4
echo-back : AT

send_count: 5
echo-back : AT
```

Figure 5. Example Output from USBX Host Class CDC-ACM Module Application Project

10. 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 the incorrect selection of lower-level modules. 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. Next Steps

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

12. Reference Information

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

Links to all the most up-to-date USBX Host Class CDC-ACM module reference materials and resources are available on the Renesas Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977570.

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 Release
1.01	Jun.08.18	—	Fixed the pin configuration for the DK-S3A7 board
1.02	May.06.19	—	Updated for SSP v1.6.0

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/.