Renesas Synergy™ Platform

UART HAL Module Guide

R11AN0085EU0101
Rev.1.01
Sep 15, 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 an efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are included in this document, and should be valuable resources for creating more complex designs.

The UART HAL Module is a high-level API for UART applications and is implemented on r_sci_uart. The UART HAL module uses the SCI peripherals on the Synergy MCU. A user-defined callback can be created to manage hardware-handshakes and data operation, if needed.

Contents

1. UART HAL Module Features

The UART HAL module supports the standard UART protocol. The UART HAL module used in concert with the SCI peripheral in UART mode (UART on SCI) supports the following features (in addition to the standard UART protocol):

- Full-duplex UART communication
- Simultaneous communication with multiple channels
- Interrupt-driven data transmission and reception
- Invoking the user-callback function with an event code in the argument
- Baud-rate change at run-time
- Hardware resource locking during UART transaction
- CTS/RTS hardware flow control (with an associated IOPORT pin and supported by user-defined callback function)
- Integration with the DTC transfer module

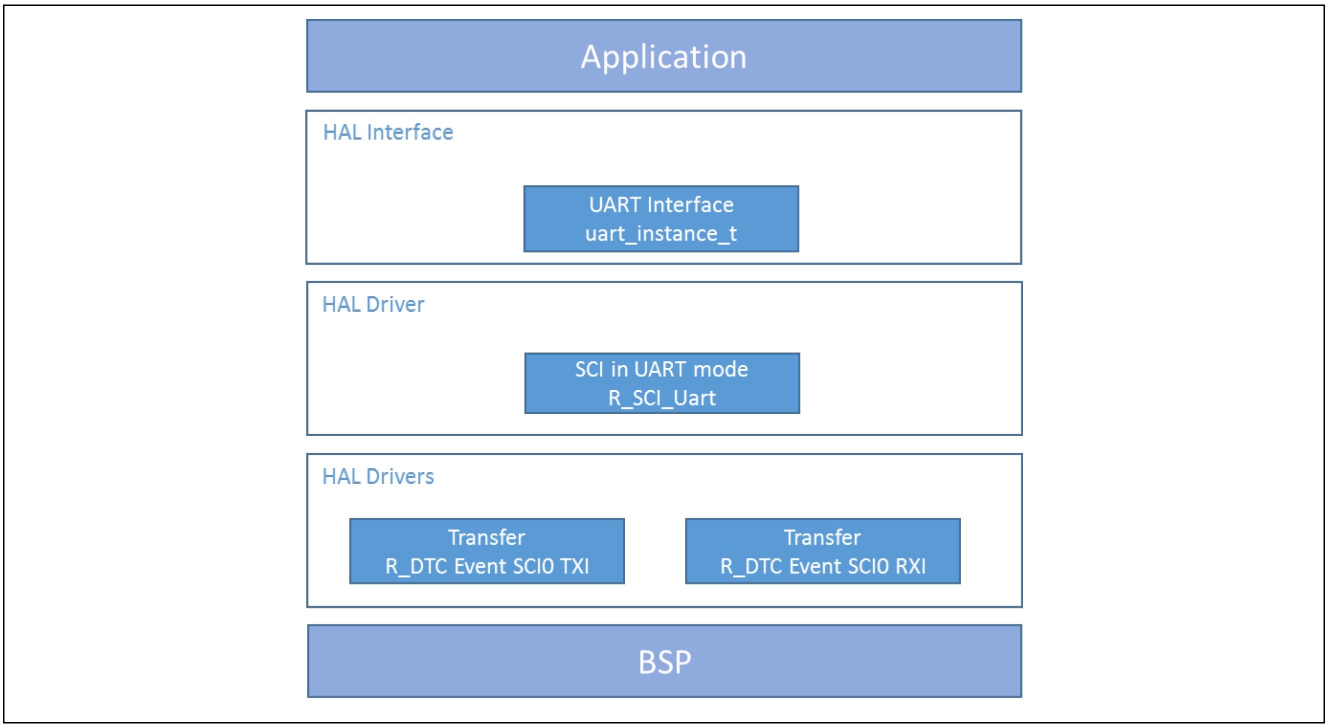

Figure 1 UART HAL Module Block Diagram

2. UART HAL Module APIs Overview

The UART HAL module interface defines APIs for key features such as opening, closing, reading, writing and setting the baud rate. 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.

Table 1 UART HAL Module API Summary

Function Name	Example API Call and Description
.open	`g_uart0.p_api->open(g_uart0.p_ctrl, g_uart0.p_cfg);` Open UART device.
.read	`g_uart0.p_api->read(g_uart0.p_ctrl, uart0_buf, uart0_rcv_num);` Read from UART device. If a transfer instance is used for reception, the received bytes are stored directly in the read input buffer, uart0_buf. When a transfer is complete, the callback is called with event UART_EVENT_RX_COMPLETE. Bytes received outside an active transfer are received in the callback function with event UART_EVENT_RX_CHAR.
.write	`g_uart0.p_api->write(g_uart0.p_ctrl, uart0_buf, uart0_send_num)` Write to UART device. The write buffer is used until write is complete. Do not overwrite write buffer contents until the write is finished. When the write is complete (all bytes are fully transmitted on the wire), the callback called with event UART_EVENT_TX_COMPLETE.
.baudSet	`g_uart0.p_api->baudSet(g_uart0.p_ctrl, (uint32_t)9600);` Change baud rate.
.infoGet	`g_uart0.p_api->infoGet(g_uart0.p_ctrl, &uart_info);` Get the driver specific information.
.close	`g_uart0.p_api->close(g_uart0.p_ctrl);` Close UART device.
.versionGet	`g_uart0.p_api->versionGet(&uart_version);` Retrieve the API version with the version pointer.

Note: For more complete descriptions of 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 operates successfully.
SSP_ERR_IN_USE	Control block has already been opened or channel is being used by another instance.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL or configuration structure is NULL.
SSP_ERR_HW_LOCKED	Channel is locked.
SSP_ERR_INVALID_MODE	Channel is used for non-UART mode or illegal mode is set.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter setting found in the configuration structure. Or source/destination address or data size is invalid against data length.
SSP_ERR_NOT_OPEN	The control block has not been opened.
SSP_ERR_UNSUPPORTED	SCI_UART_CFG_RX_ENABLE is set to 0.

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. UART HAL Module Operational Overview

The UART HAL Module manages data flow using the standard UART protocol. The high-level APIs are used to read, write, and set the baud rate for the UART interface. In addition, interrupts are typically used to simplify the management of low-level activities.

Note: Interrupts need to be enabled for the following functions to operate successfully.

3.1 UART on SCI RXI interrupt

The RXI interrupt controls the flow of data received from the UART port. When the amount of received data reaches the expected read length, the interrupt service routine (ISR) invokes a user-defined callback (`p_callback`) with the argument `uart_callback_args_t` to indicate that the received data is complete. When the External RTS Operation option is enabled, the ISR invokes the UART callback function for the RTS external pin control twice: once at the top of ISR and once at the bottom. You can use the callback function to emulate the RTS function (see the UART on SCI hardware flow-control section). This interrupt is activated in the `open` API as long as the reception is enabled in the SCI_UART_CFG_RX_ENABLE configuration parameter.

3.2 UART on SCI TXI interrupt

The TXI interrupt handles consecutive transmissions of data to the UART port as requested by the `write` API. When no data is left in the transmit circular buffer, the ISR deactivates the TXI interrupt and activates the TEI interrupt to handle the last sequence in the data transmission. This interrupt is activated in the `write` API as long as the transmission is enabled by the SCI_UART_CFG_TX_ENABLE configuration parameter.

3.3 UART on SCI TEI interrupt

The TEI interrupt handles the last data transmission to the UART port requested by `write` API. This interrupt is activated by TXI ISR and deactivates itself. The ISR invokes a user-defined callback (`p_callback`) with the argument `uart_callback_args_t` to indicate that the end of data was transmitted.

3.4 UART on SCI ERI interrupt

The ERI interrupt handles errors that occur in the UART reception. This interrupt is activated in the `open` API as long as the reception is enabled by the SCI_UART_CFG_RX_ENABLE configuration parameter. The ISR invokes a user-defined callback (`p_callback`) with the argument `uart_callback_args_t` to indicate the uart_event_t cause of an error.

3.5 UART HAL Module Important Operational Notes and Limitations

3.5.1 UART HAL Module Operational Notes

UART on SCI Hardware Flow Control

The SCI hardware module supports hardware flow control for only one of the RTS or CTS signals at a time. CTS and RTS are multiplexed on the CTSn/RTSn pin so that one of the hardware flow-control signals can be used exclusively depending on the use case. The UART HAL module expands this specification and allows control of both the CTS and the RTS signal by enabling an additional pin for the RTS signal. To enable this mode, set the UART on SCI configurations as follows:

- Set SCI_UART_CFG_EXTERNAL_RTS_OPERATION to Enable.
- Set ctsrts_en to CTS (true).
- Specify a user-callback function name to “Name of UART callback function for the RTS external pin control” in p_extpin_ctrl.

The UART on SCI HAL module invokes the user callback function from the RXI ISR at the top and at the end of processing.

The callback function argument “level” refers to the signal level on the RTS pin for the selected SCI channel.

Note: The HAL module does not handle the GPIO pin initialization or control it. Instead, you need to initialize the GPIO pin before starting the UART reception.

The following figure shows the timing diagram of CTS/RTS hardware flow-control with an external GPIO pin used as the RTS signal:

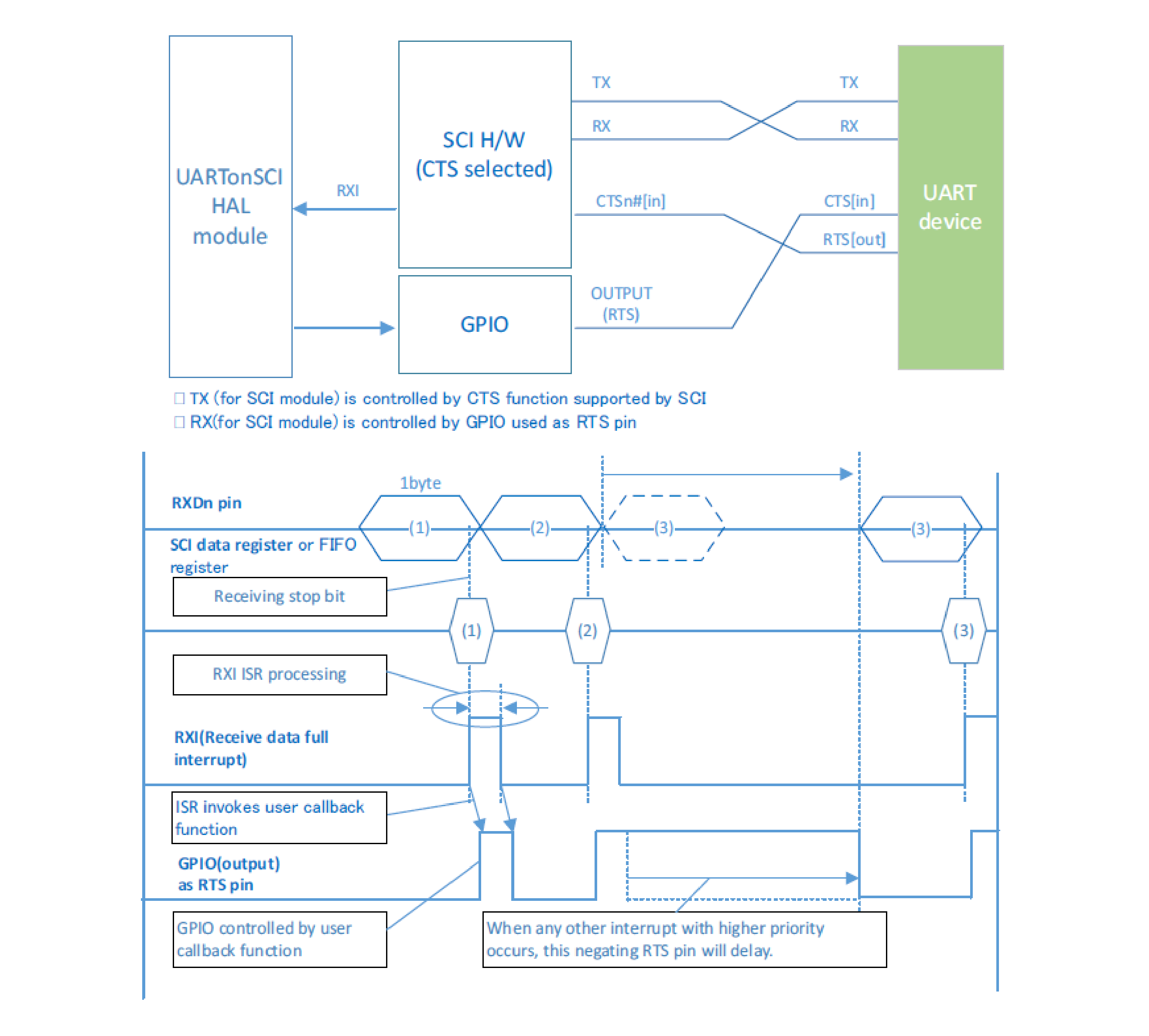

Figure 2 CTS/RTS Hardware Control with an External GPIO

Note: The UART on SCI module on the SK-S7G2 board uses PORT8 pin0 (pin P800) and J8 to activate the RS232C port on the RS-232C transceiver. Connect pin 1 and pin 2 of J8. Configure pins P800 as IOPORT pins and set its level for the desired operation.

3.5.2 UART HAL Module Limitations

- The module supports interrupt-based operation but does not support a polled UART mode.
- The module does not support non-buffered UART mode.
- The module does not support Event Link functionality.
- There is a 64k limit to the block size that can be sent to the r_sci_uart driver if the DTC is being used. This limit can be retrieved using the `infoGet` API.
- Refer to the most recent SSP Release Notes for any additional operational limitations for this module.

4. Including the UART HAL Module in an Application

This section describes how to include the UART HAL 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 UART Driver to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the UART HAL is g_uart0. This name can be changed in the associated Properties window.)

Table 3 UART Driver Stack Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_uart0 UART on r_sci_uart	Threads->HAL/Common Stacks	Highlight **Threads > HAL/Common Stacks and select New Stack > Driver > Connectivity > UART Driver on r_sci_uart**

When the UART HAL module on r_sci_uart is added to a thread as shown in the following figure, the configurator automatically adds the needed lower-level drivers. Any drivers that need additional configuration information will be 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 only be added once and can be used by multiple stacks.

When the mouse hovers over the Red position, the required operations for correcting the configuration will display. Please follow the instructions to enable the SCI Receive Interrupt (RXI), SCI Transmit Interrupt (TXI), and SCI Transmit End Interrupt (TEI) in the Properties window to complete a valid configuration.

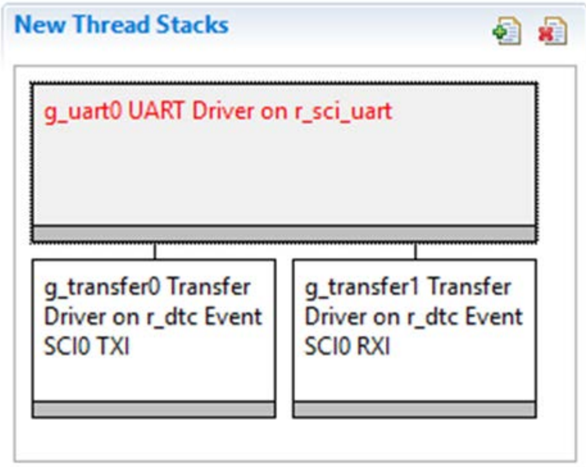

Figure 3 UART HAL Module Stack

5. Configuring the UART HAL Module

The UART HAL module must be configured by the user 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. Furthermore, only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and 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.

Note: You may want to open your ISDE, create the UART HAL module and explore the property settings in parallel with looking over the following configuration table settings. This will help 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 UART HAL Module on r_sci_uart

ISDE Property	Value	Description
External RTS Operation	Enable, Disable Default: Disable	Enable an IOPORT pin to be used as RTS signal. For RTS functionality, set this configuration parameter to **Enable** and specify the configuration **Name of UART callback function for the RTS external pin control**.
Reception	Enable, Disable Default: Enable	Enable or disable UART reception for all UART channels on SCI. Setting this configuration parameter to **Disable** reduces code size because the portion of code for UART reception is not compiled. You cannot set this parameter for individual UART channels.
Transmission	Enable, Disable Default: Enable	Enable or disable UART transmission for all UART channels on SCI. Setting this configuration to **Disable** reduces code size because the portion of code for UART transmission is not compiled. However, you can only set this configuration to **Disable** if no other SCI channels which work as UART ports are transmitting.
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_uart0	The name to be used for UART on SCI module control block instance. This name is also used as the prefix of the other variable instances.
Channel	0-9	SCI channel number.
Baud Rate	9600	Baud rate selection.
Data Bits	7 bits, 8, bits, 9 bits Default: 8 bits	UART data bits.
Parity	None, Odd, Even Default: None	UART parity bits.
Stop Bits	1 bit, 2 bits Default: 1 bit	UART stop bits.
CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin), RTS (CTS is disabled) Default: RTS (CTS is disabled)	Select CTS or RTS for the CTSn/RTSn pin of SCI channel n. The SCI hardware supports either the CTS or the RTS control signal on this pin but not both. For an application that uses both CTS and RTS, select **CTS** for this configuration parameter and enable the configuration **External RTS Operation** specifying the configuration **Name of UART callback function for the RTS external pin control**.

Name of UART callback function to be defined by user	user_uart_callback	Name must be a valid C symbol.
Name of UART callback function for the RTS external pin control to be defined by user	NULL	Name must be a valid C symbol.
Clock Source	Internal Clock, External Clock 8x baudrate, External Clock 16x baudrate Default: Internal Clock	Selection of the clock source to be used in the baud-rate clock generator block.
Baudrate Clock Output from SCK pin	Enable, Disable Default: Disable	Optional setting to output the baud-rate clock on the SCKn pin for the selected channel n.
Start bit detection	Falling Edge, Low Level Default: Falling Edge	Start bit detection mode in the reception, usually set **Falling Edge** to this configuration.
Noise Cancel	Enable, Disable Default: Disable	Enable the digital noise cancellation on RXDn pin. The digital noise filter block in SCI consists of two-stage flip-flop circuits. For details, refer to the Noise cancellation section in the Renesas Synergy hardware manual.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Bit rate modulation enable selection.
Receive 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	Receive interrupt priority selection.
Transmit 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	Transmit interrupt priority selection.
Transmit End 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	Transmit end interrupt priority selection.

Error 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	Error interrupt priority selection.

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

In some cases, settings other than the defaults for lower-level modules can be desirable. For example, it might be useful to select different noise cancellation settings. 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 fairly intuitive and usually can be determined by inspection of the associated Properties window from the SSP configurator.

5.1 Configuration Settings for the UART HAL Module Lower 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 will be locked to prevent user modification. The following table identifies all the settings within the properties section for the module:

Table 5 Configuration for the Transfer Driver on r_dtc Event SCI0 TXI

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Set start mode
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section setting
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode)	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency 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)	Event SCI0 TXI	Activation source selection

Auto Enable	True, False Default: True	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event 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	ELC Software Event interrupt priority selection

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 6 Configuration for the Transfer Driver on r_dtc Event SCI0 RXI

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Set start mode
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section setting
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 RXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ELC Software Event 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 Thread X, CM0: invalid), Disabled Default: Disable	Interrupt priority for ELC SW event

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.

When the UART with CTS and RTS function are used simultaneously, the transfer driver cannot be used. Please delete all transfer drivers on the low level. After being deleted, the optional transfer driver will display in pink, meaning the driver is recommended but optional, as in the following figure:

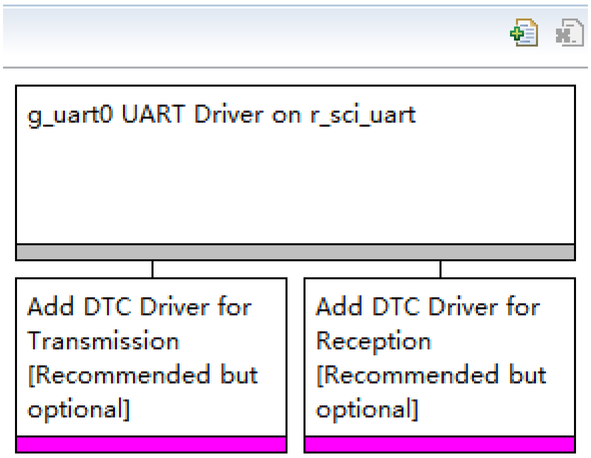

Figure 4 UART Stack with the CTS and RTS Functions

5.2 UART HAL Module Clock Configuration

The SCI UART peripheral uses PCLKA as its clock source (PCLKB for S124) or an external clock from the SCKn pin for the selected channel n.

5.3 UART HAL Module Pin Configuration

The SCI UART peripheral 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 the UART pins.

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

Table 7 Pin Selection Sequence for UART HAL Module on SCI

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select **Peripherals > Connectivity: SCI > SCI0**

Table 8 Pin Configuration Settings for UART HAL Module on SCI

Pin Configuration Property	Value	Description
Pin Group Selection	Mixed, _A Only, _B Only (Default: Mixed)	Pin grouping selection
Operation Mode	Disabled, Custom, Asynchronous UART, Simple SPI, Simple I2C, Synchronous UART, SmartCard (Default: Simple SPI)	Select Operation Mode for UART on SCI
TXD_MOSI	None, P411, P101 (Default: P411)	TXD Pin
RXD_MISO	None, P410, P100 (Default: P410)	RXD Pin
SCK	None, P412, P102 (Default: P412)	SCK Pin
CTS_RTS_SS	None, P413, P103 (Default: None)	CTS Pin
SDA	Disabled	SDA Pin (when Simple I2C is used)
SCL	Disabled	SCL Pin (when Simple I2C is used)

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

6. Using the UART HAL Module in an Application

Once the module has been configured and the files generated, the UART HAL module is ready to be used in an application. The typical steps in using the UART HAL module in an application are:

1. Initialize the UART HAL Module using the `open` API.
2. Set Baud Rate with the `baudSet` API (if needed).
3. Read and Write data as needed using the `read` and `write` APIs and callbacks.
4. Close the UART HAL module using the `close` API as needed.

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

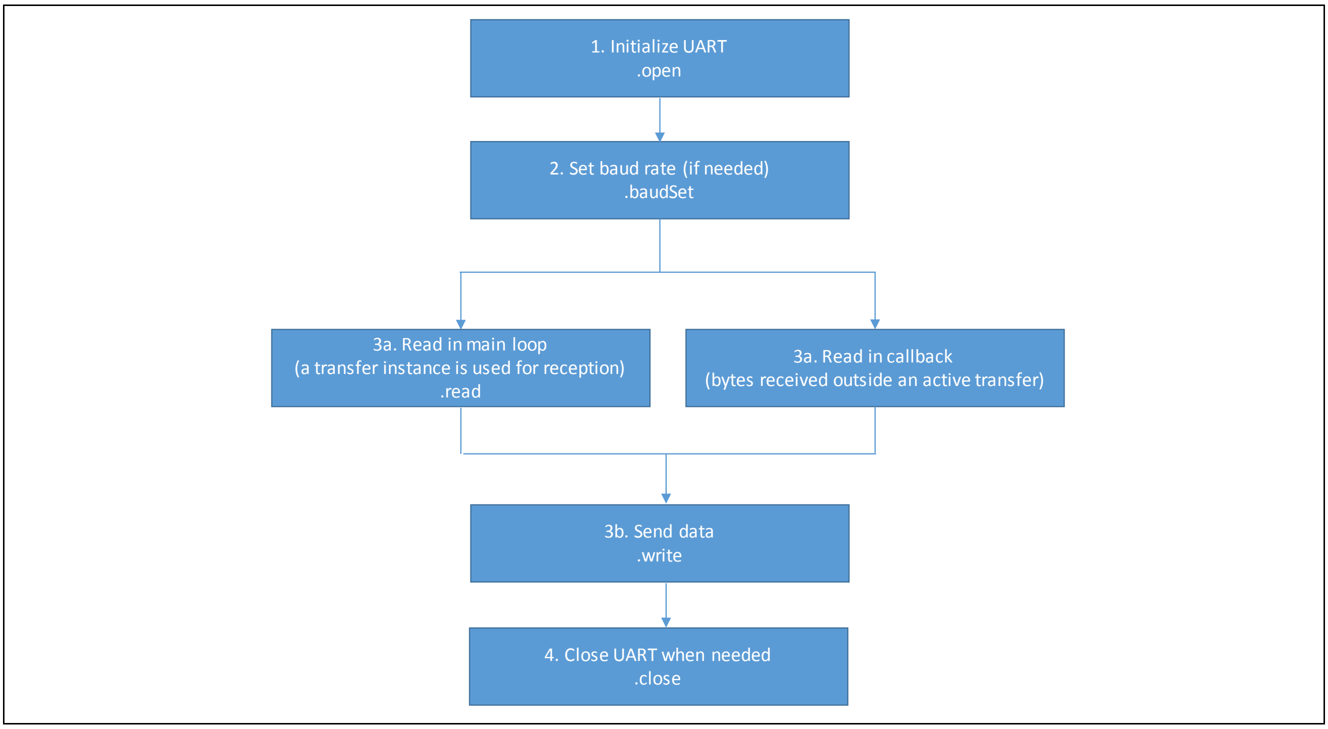

Figure 5 Flow Diagram of a Typical UART HAL Module Application

7. The UART HAL Module Application Project

The application project associated with this module guide demonstrates the aforementioned steps in a full design. You may want to import and open the application project within the ISDE and view the configuration settings for the UART HAL module. You can also read over the code (in `uart_hal_mg.c`) which illustrates the UART APIs in a complete design.

The application project demonstrates two uses of the UART APIs, one for generic operation and the other for CTS/RTS hardware flow-control using an external GPIO pin as the RTS signal. For the flow control example, a second board is required to send the specified bytes to the SK-S7G2.

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

Table 9 Software and Hardware Resources Used by the Application Project

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

Important Note: Two SK-S7G2 kits are required to demonstrate the flow control example.

The generic operation flow is as follows (for UART4):

- Initialize the UART HAL module using the `open` API.
- Initialize data buffer for storing received data from external device using the `read` API.
- Set flag in interrupt callback function when data reception is completed (when event code of UART_EVENT_RX_COMPLETE is set.)
- Operate on the received data as needed by the application when the received completion flag is set.
- Transmit data to external device according to the application.
- Wait for the completion of transmission. In the interrupt callback function, after the last data is sent, the event code of UART_EVENT_TX_COMPLETE is set.
- Close the UART HAL module using the `close` API after the data transmission is finished.

The specific operation flow of CTS/RTS hardware flow-control using external GPIO pin as the RTS signal is as follows (for UART0):

- Initialize the UART HAL module using the `open` API.
- Change baud rate as needed using the `baudSet` API.
- Store received data into user-defined buffer in the interrupt callback function (when the event code of UART_EVENT_RX_CHAR is set).
- Set the flag in the interrupt callback function when data reception is completed.
- Operate on the received data as needed by the application when the received completion flag is set.
- Transmit data to external device according to the application.
- Wait for the completion of transmission. In the interrupt callback function, after the last data is sent, the event code of UART_EVENT_TX_COMPLETE is set.
- Close the UART HAL Module using the `close` API after the data transmission is finished.

A simple flow diagram of the application project is given in the following figures:

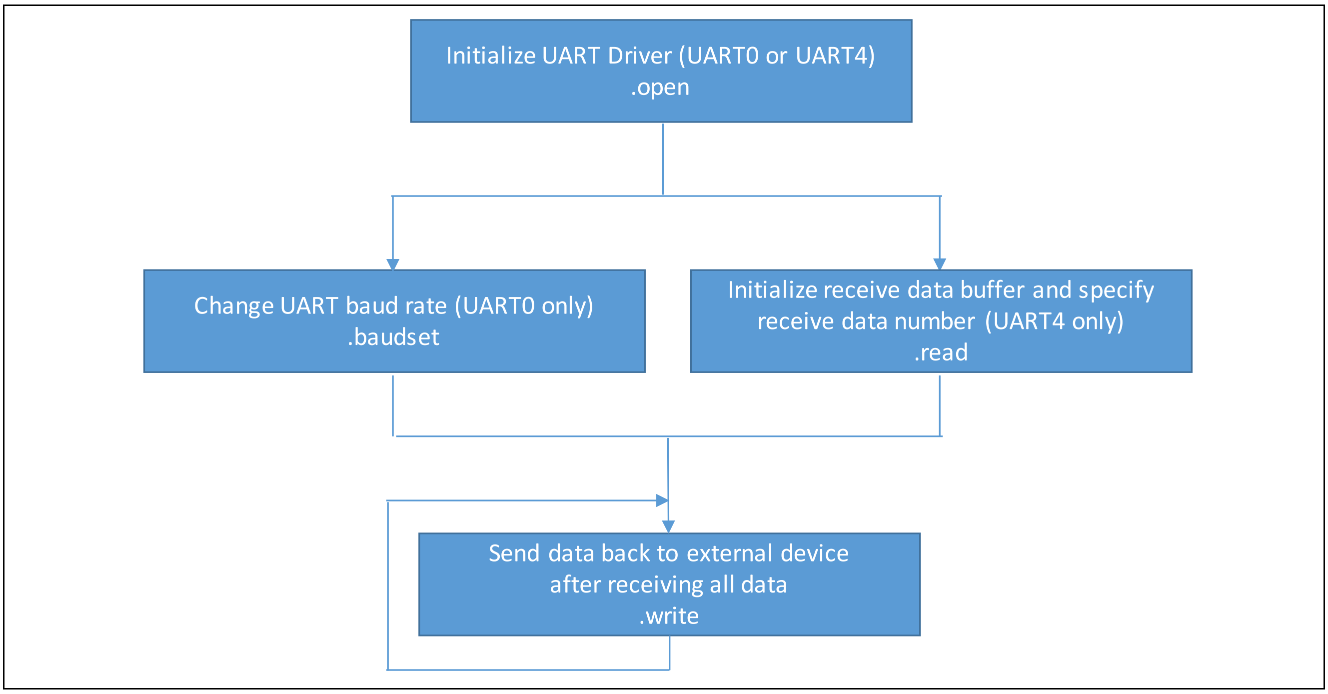

Figure 6 UART Application Project Flow Diagram in Main Loop

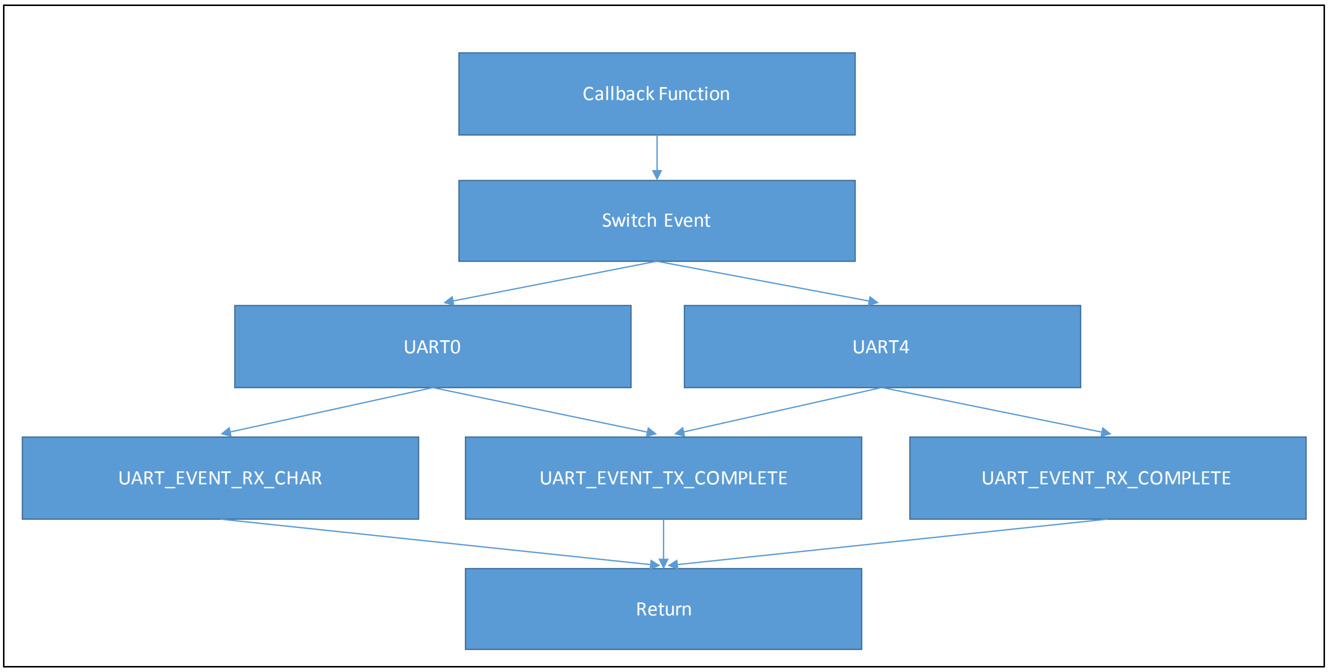

Figure 7 Detailed Flow Chart of the Callback Function

The `uart_hal_mg.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 `uart_hal_mg.c` has the header files which reference the UART instance structure and external function declaration. The `hal_entry.c` calls the `uart_hal_demo()` function in `uart_hal_mg.c`. There are two types of operation in this demo project, the generic operation (UART4) and the specific operation (UART0).

- Multiple data (16 bytes) can be stored into a user-defined buffer automatically by setting the byte's number using the `read` API in the generic mode. In this condition, the DTC is used to receive data successively. However, each data needs to be stored into a user-defined buffer manually in the callback function in the specific mode.
- Data transaction status for the generic mode can be found from the callback function with event enumeration UART_EVENT_RX_COMPLETE and UART_EVENT_TX_COMPLETE. While for the specific mode, it can be found with event enumeration UART_EVENT_RX_CHAR and UART_EVENT_TX_COMPLETE.
- For specific operation, a callback function named `user_rts_callback` is also used to specify which pin that RTS signal uses.

If the semi-hosting function is enabled, the `printf` function will output all received data and send data to the Debug Virtual Console.

Note: This description assumes you are familiar with using `printf()` with 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" 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 tables. You can also open the application project and view these settings in the Properties window as a hands-on exercise.

UART0 in specific operation:

Table 10 UART HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
External RTS Operation	Enable
Name	g_uart0
Channel	0
Baud Rate	115200
CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin)
Name of UART callback function to be defined by user	user_uart0_callback
Name of UART callback function for the RTS external pin control to be defined by user	user_rts_callback
Receive Interrupt Priority	Priority 2
Transmit Interrupt Priority	Priority 2
Transmit End Interrupt Priority	Priority 2
Error Interrupt Priority	Priority 2

UART4 in generic operation:

Table 11 UART HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
External RTS Operation	Enable
Name	g_uart4
Channel	4
Baud Rate	9600
CTS/RTS Selection	RTS (CTS is disabled)
Name of UART callback function to be defined by user	user_uart4_callback
Receive Interrupt Priority	Priority 2
Transmit Interrupt Priority	Priority 2
Transmit End Interrupt Priority	Priority 2
Error Interrupt Priority	Priority 2

Note: The mode is defined through “#define UartNormal” in the application project. If the specific mode is used, please comment out this sentence.

To access a particular channel or pin, the SCI pin must be set in the Pins tab of the ISDE.

The following tables illustrate the method for selecting the pins within the SSP configuration window and example selections for the SCI pin.

UART0 in specific operation:

Table 12 Pin Selection Sequence for UART HAL Module on SCI

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select Peripherals > Connectivity: SCI > SCI0

Table 13 Pin Configuration Settings for UART HAL Module on SCI

Pin Configuration Property	Value
Pin Group Selection	Mixed
Operation Mode	Custom
TXD_MOSI	P411
RXD_MISO	P410
SCK, SDA, SCL	None
CTS_RTS_SS	P413

Note: In order to set the P413 to CTS_RTS_SS, the port setting of P413 as a GPIO output mode (initial low) must be disabled.

Table 14 Pin Selection Sequence for External RTS pin

Resource	ISDE Tab	Pin selection Sequence
GPIO	Pins	Select Ports > P1 > P100

Table 15 Pin Configuration Settings for External RTS pin

Pin Configuration Property	Value
Mode	Output mode (Initial Low)
Pull up	None
IRQ	None
Driver Capacity	Low
Output type	CMOS

Note: In order to set the P100 to output mode, the operation mode setting of SPI0 must be disabled.

UART4 in generic operation:

Table 16 Pin Selection Sequence for UART HAL Module on SCI

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select Peripherals > Connectivity: SCI > SCI4

Table 17 Pin Configuration Settings for UART HAL Module on SCI

Pin Configuration Property	Value
Pin Group Selection	Mixed
Operation Mode	Asynchronous UART
TXD_MOSI	P512
RXD_MISO	P511

Note: The example values are for a project using the Synergy S7G2 and the SK-S7G2 Kit. Other Synergy Kits and other Synergy MCUs may have different available pin configuration settings.For pin using, please refer to the following figure shows.

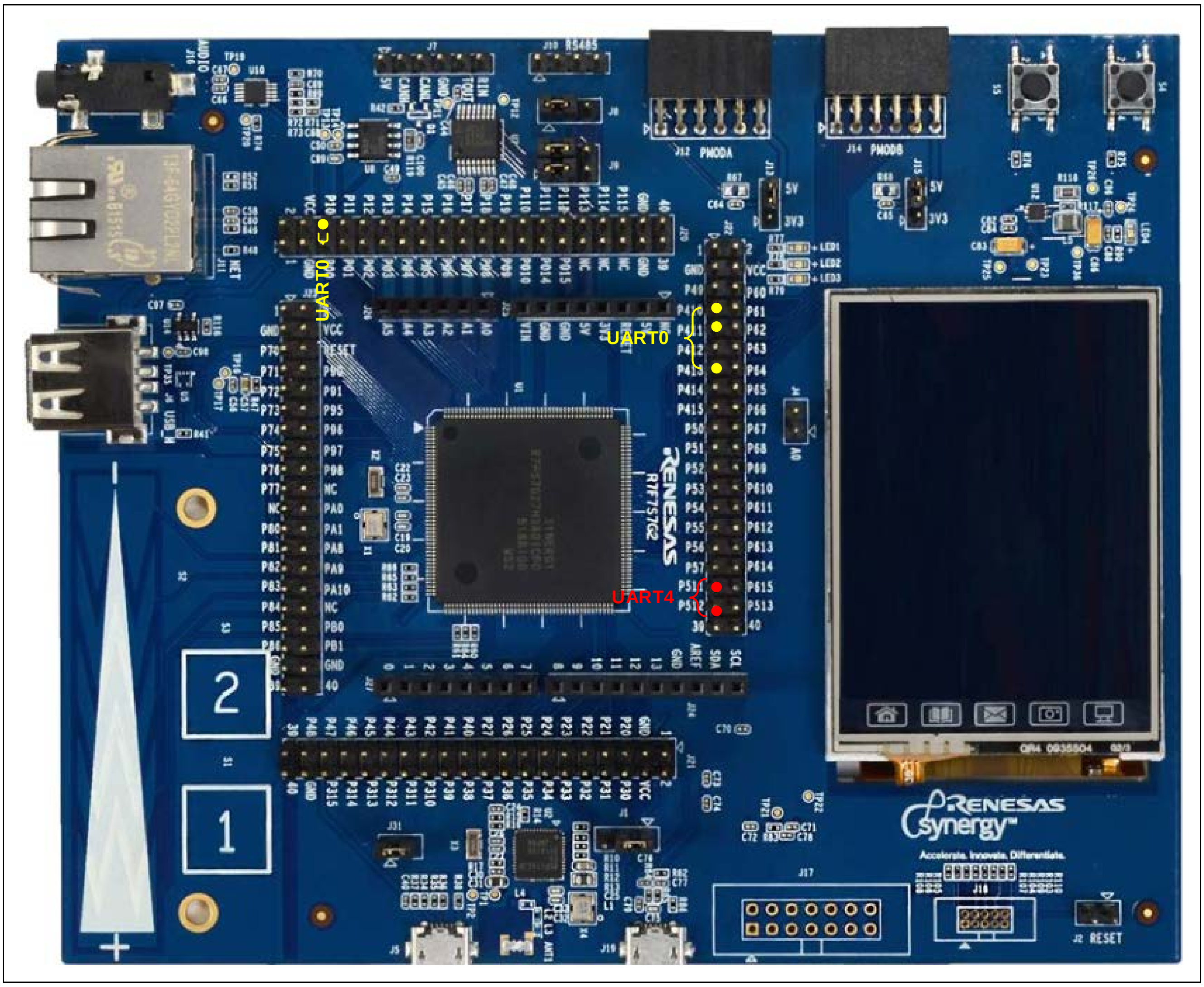

Figure 8 Hardware connection picture

8. Customizing the UART HAL Module for a Target Application

Some configuration settings will normally be changed by the developer from those shown in the application project. For example, the user can easily change the configuration settings for the UART clock by updating the PCLKA/PCLKB in the Clocks tab. The user can also change the UART port pins to select the desired input; this can be done using the **Pins** tab in the configurator. The CTS/RTS function cannot be used for UART mode based on SCI.

9. Running the UART HAL Module Application Project

To run the UART HAL 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 (r11an0023eu0116-synergy-ssp-import-guide.pdf, included in this package) for instructions on importing the project into e^2 studio or IAR embedded workbench and building/running the application.

To implement the UART HAL 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, while just reading over this guide will tend to be more theoretical.

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 UART HAL module application project, simply follows these steps:

1. Create a new Renesas Synergy project for the SK-S7G2 board (S7G2-BSP) called “UART_HAL”.
2. Select the BSP in Project Template Selection page when creating a project. Then finish a new project setup.
3. Select the **Threads** tab -> **HAL/Common**.
4. Add the UART HAL module to the HAL/Common stack.
5. Configure the parameters contains enabling the interrupt.
6. Click on the **Generate Project Content** button.
7. Add the code from the supplied project files `art_hal_mg.c`, `uart_hal_mg.h`, and `hal_entry.c`.
8. Compile the project.
9. Connect to the host PC via a micro USB cable to J19 on SK-S7G2.
10. Use another board with SCI generic and specific functions to communicate with the SK. For UART0, the RxD, TxD and external RTS pin (P100) must be connected with another SK-S7G2. For UART4, only the RxD and TxD need to be connected.
11. Use “UartNormal” (defined in the project) to select generic operation (UART4) or specific operation (UART0.)
12. Start to debug the application. 16 bytes of data sent from another board will be input into the SCI, and after receiving all bytes, these bytes will be sent back to the opposite side.
13. If semi-hosting is enabled, the output can be viewed in the Renesas Debug Virtual Console. The first picture is the output value of the UART0 (specific operation) and the second is the output value of the UART4 (generic operation).

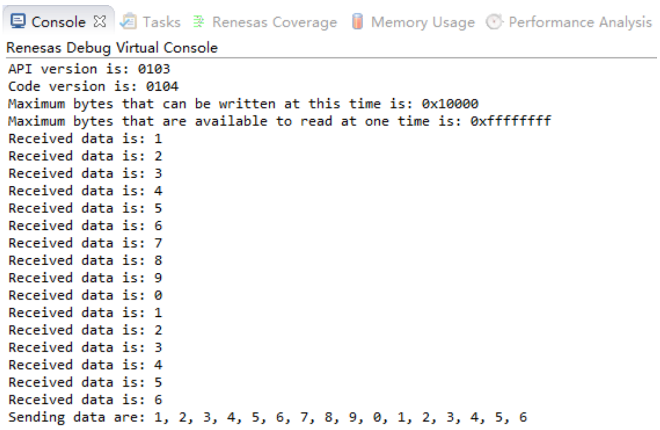

Figure 9 Example Output from UART0

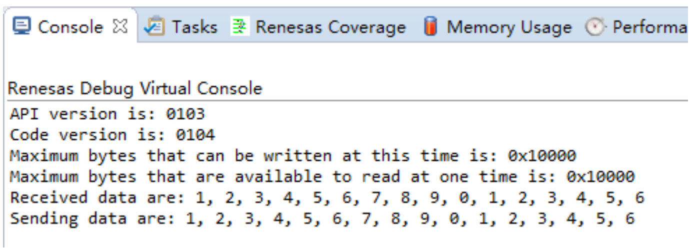

Figure 10 Example Output from UART4

14. The output can also be viewed through an oscilloscope.

Figure 11 UART0 Waveform (not using semi-hosting)

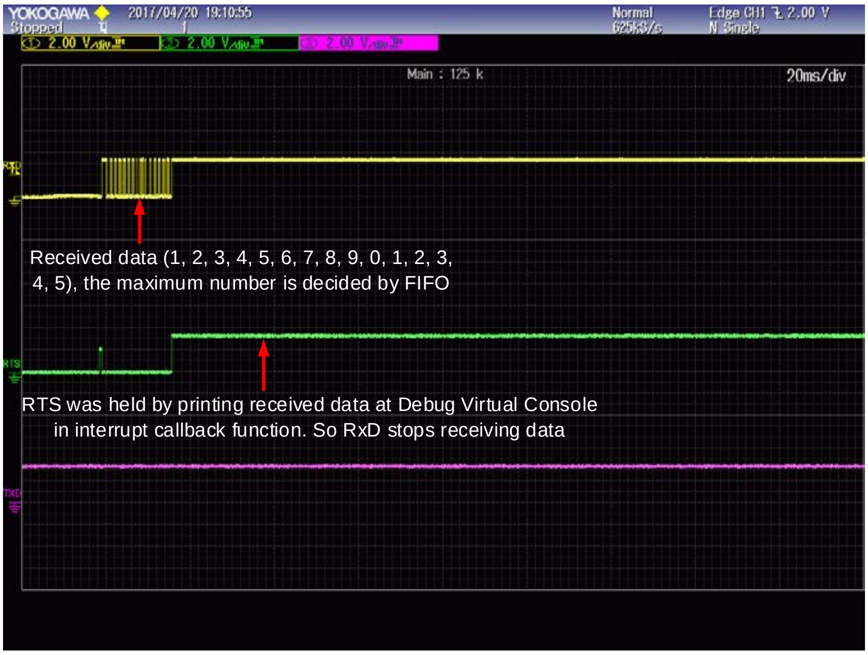

Figure 12 UART0 Waveform (using semi-hosting) – the first frame

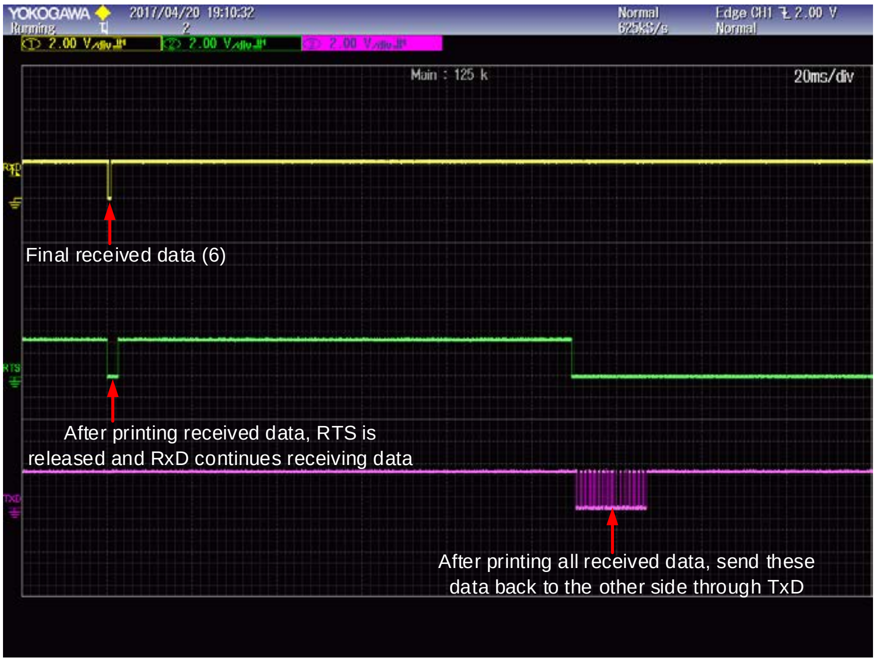

Figure 13 UART0 Waveform (using semi-hosting) – the second frame

Figure 14 UART4 Waveform (not using semi-hosting)

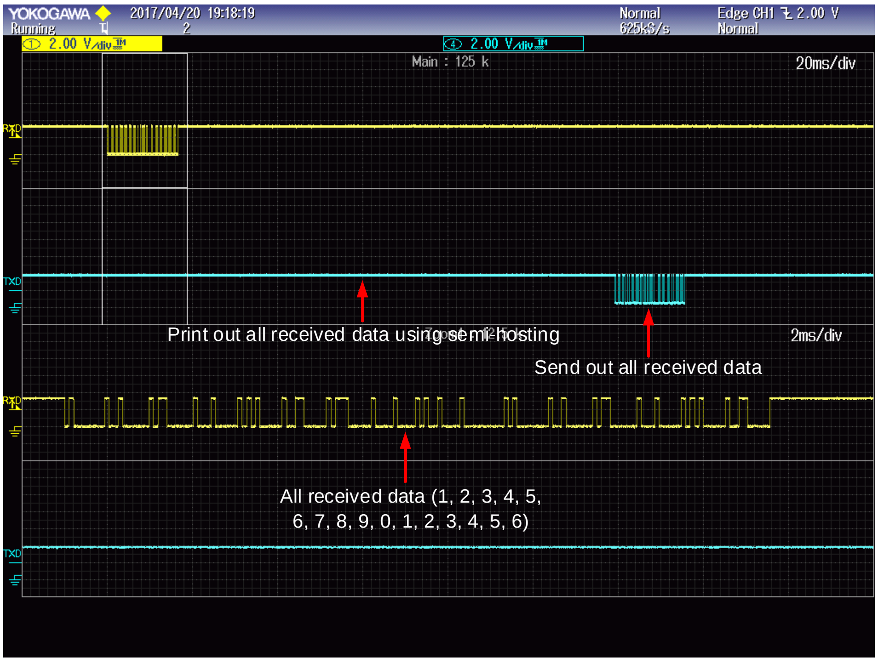

Figure 15 UART4 Waveform (using semi-hosting)

Note: When semi-hosting is used to display received data on the Renesas Debug Virtual Console, the RTS will be held by the `printf` function used in the `user_uart_callback` function when the ISR is called. Not using semi-hosting can reduce the time in ISR. A high level on RTS will forbid the opposite side sending data continuously.

10. UART HAL Module Conclusion

This module guide has provided all the background information needed to select, add, configure, and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy Platform makes these steps much less time consuming and removes the common errors, like conflicting configuration settings or incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrate 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. UART HAL Module Next Steps

After you have mastered a simple UART HAL module project, you may want to review a more complex example. For example, using RS-485. This module guide includes an additional application project demonstrating RS-485. This example uses the DK-S7G2 kit.

The UART on SCI module on the DK-S7G2 board uses PORT9 pin14 (pin P914), S101 and S102 to activate the RS-485 port on the dual protocol RS-232/RS-485 transceiver.

On the main board with the S5 DIP switch, set JTAG to the ON position and set PBS to the ON position to enable PORT8 pin7 (pin P807) and pin10 (pin P810) to light the LED with green and red, and enabling S1 to trigger master to send data.

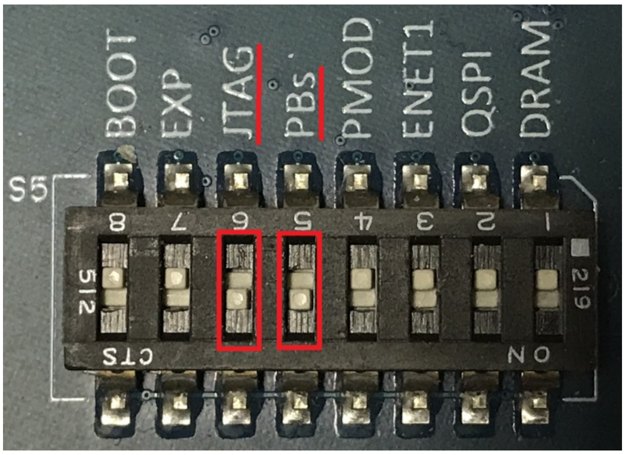

Figure 16 DIP Switch S5 configuration

On the base board, DIP switch S101, RS, is in the ON position. On DIP switch S102, set 232 to the OFF position, then the slew rate of transceiver can be selected using SLEW and SPB (Please refer to transceiver's datasheet to get more information). All other switches on S101 and S102 are OFF.

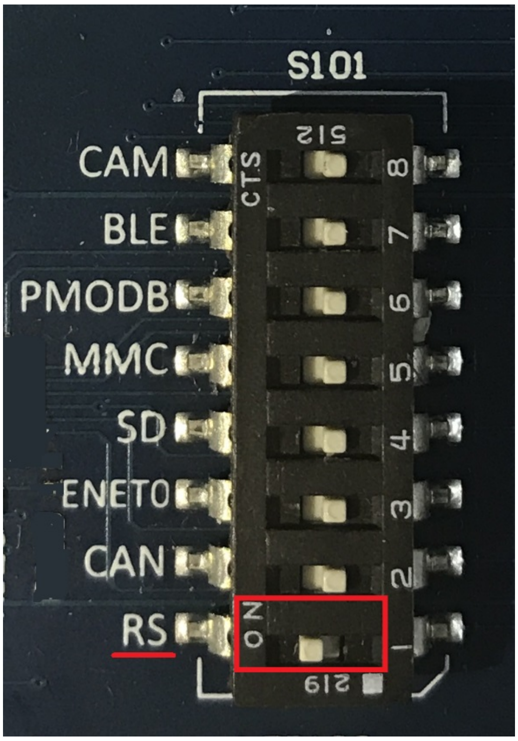

Figure 17 DIP Switch S101 configuration

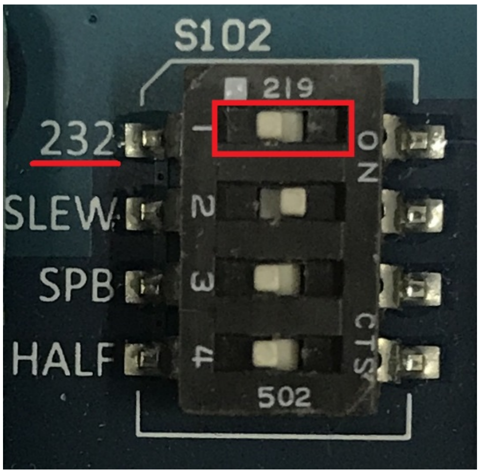

Figure 18 DIP Switch S102 configuration

J112 on the base board is output connector for RS-485. Please connect with the additional kit according to the below picture.

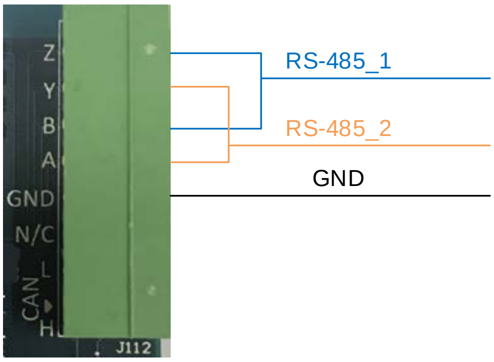

Figure 19 Connection Example for J112 on the base board

RS-485 communication uses generic operation mode on SCI1. Active high driver output enable and active low receiver output enable of transceiver are connected and P914 is used to control the direction of reception and transmission. Click button S1 will trigger data transfer from master device. When slave device receives correct data from master device, “Received data is right!” will be back and LED2 ligh up GREEN. Otherwise, “Received data is wrong!” will be back and LED2 light RED.

A simple flow diagram of RS-485 application project is given in the figures below.

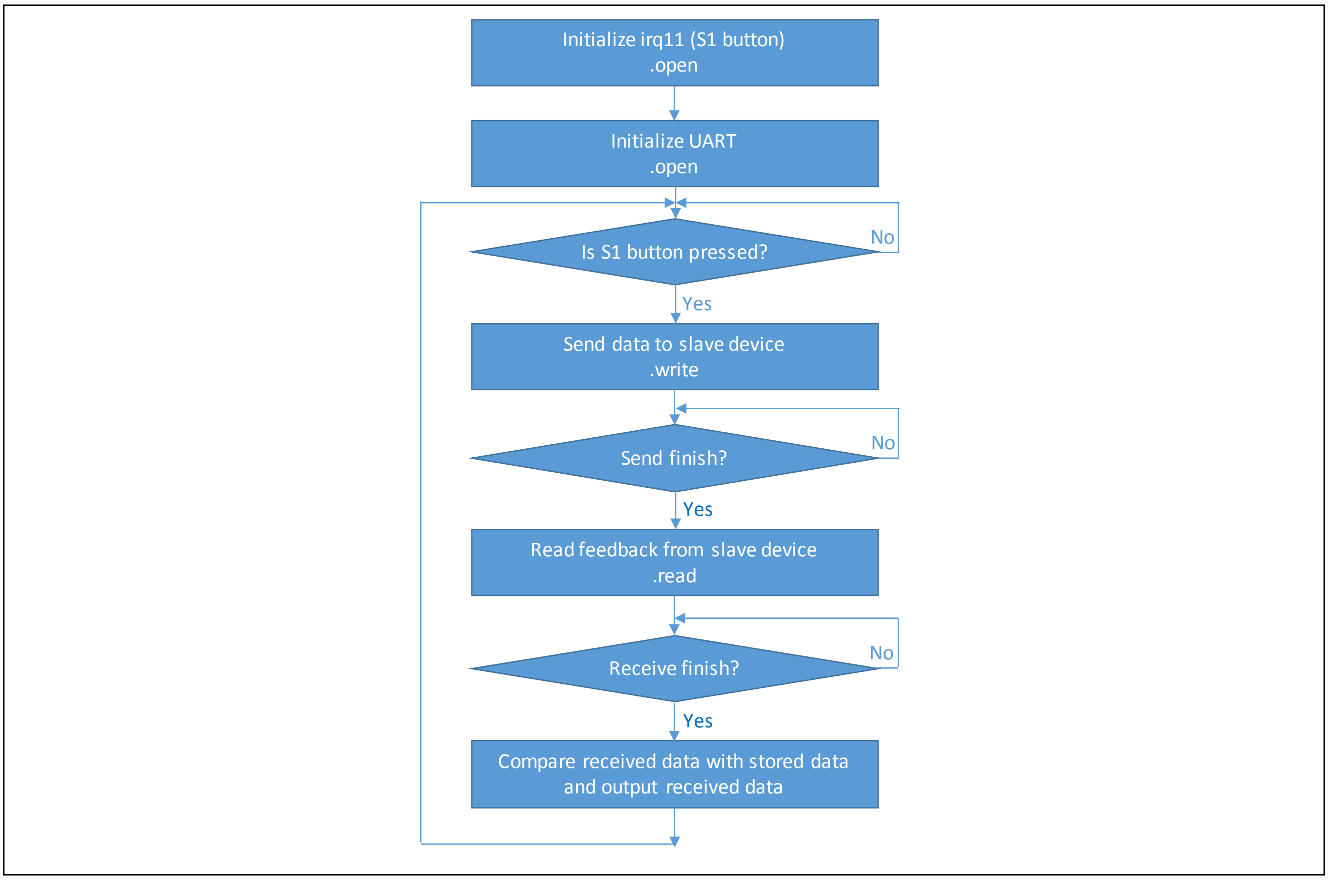

Figure 20 RS-485 Master Project Flow Diagram

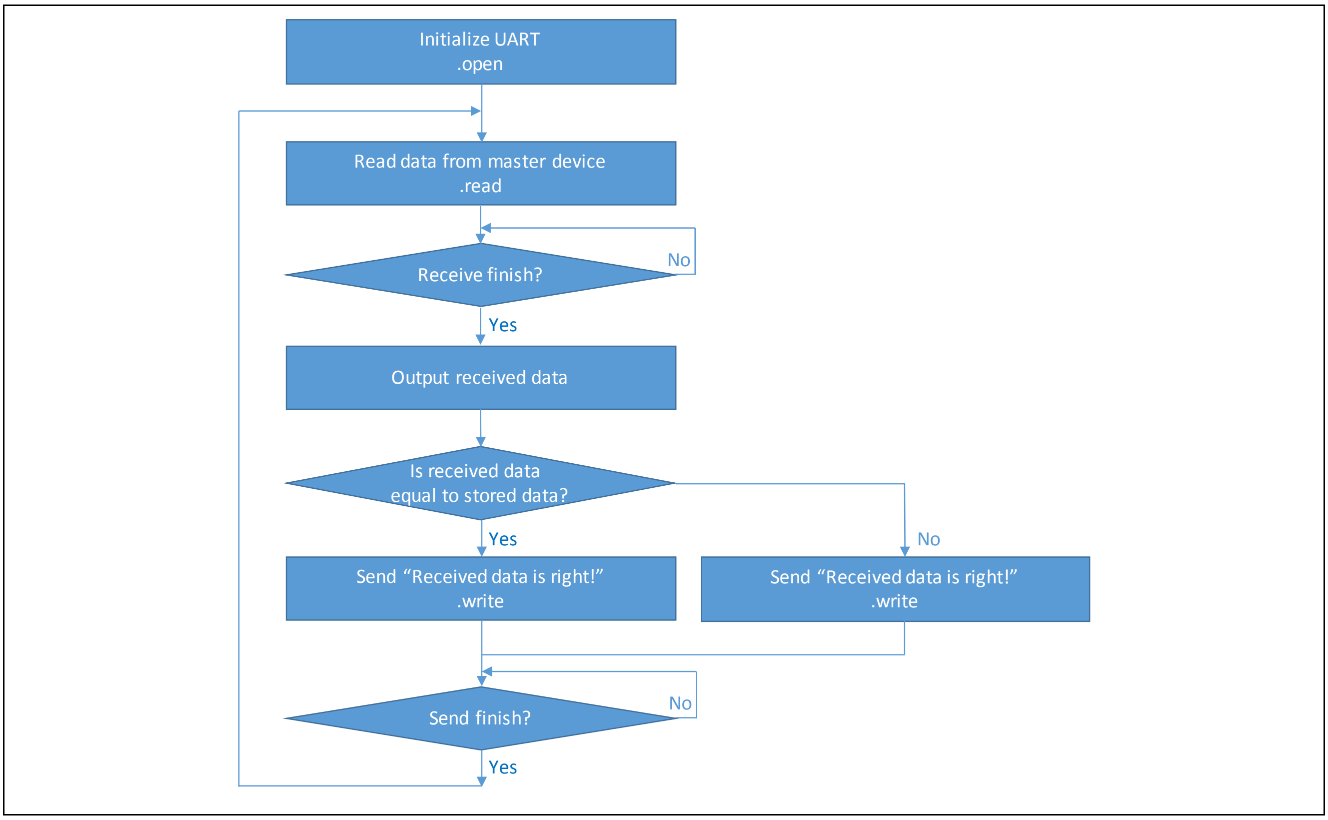

Figure 21 RS-485 Slave Project Flow Diagram

Note: When using master project, `uart_hal_mg_RS485_slave.c` should be excluded from building, and vice versa. Right click "*.c" file and select exclude from build to exclude unnecessary files. `#define MASTER` in `hal_entry.h` is used for the master project. Comment it out when a slave project is used.

The output can be viewed in the Renesas Debug Console. The first picture is output value of master and the second is output value of slave.

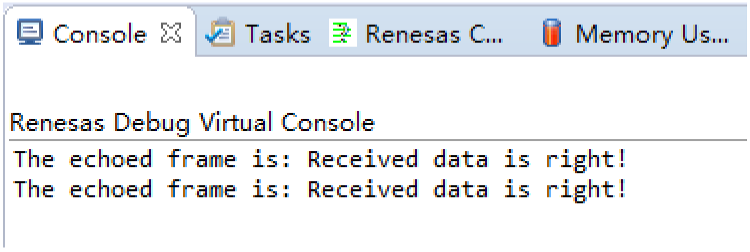

Figure 22 Example Output from RS-485 Master Project when received data is right

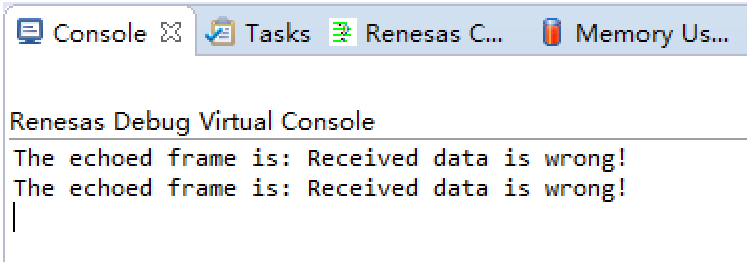

Figure 23 Example Output from RS-485 Master Project when received data is wrong

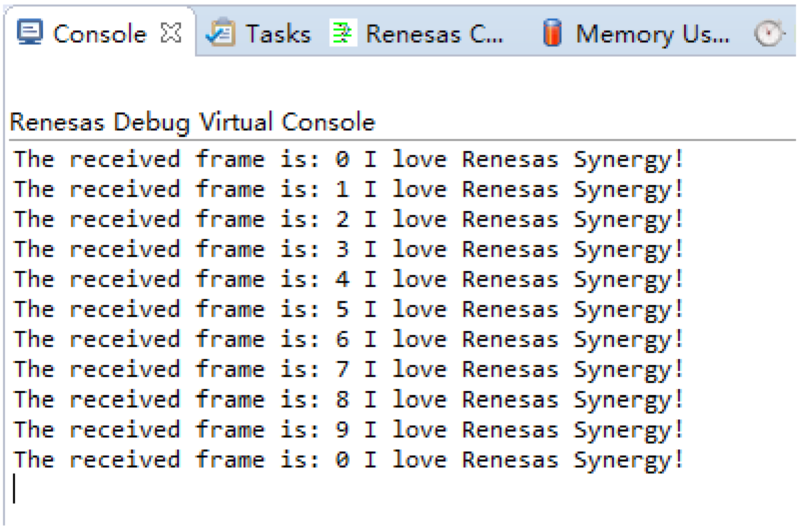

Figure 24 Example Output from RS-485 Slave Project

The output can be also viewed through an oscilloscope.

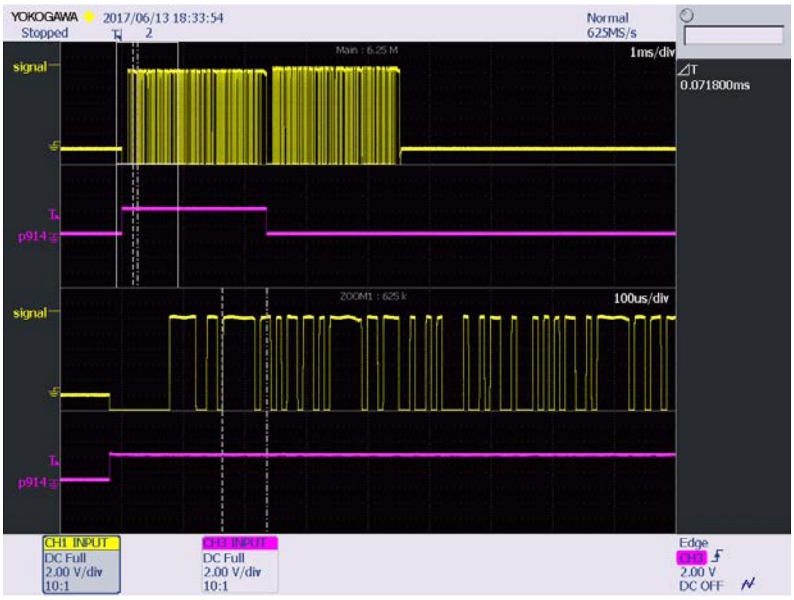

Figure 25 Differential Waveform

Besides RS-485 communication, you also may find that the Communications Framework is a good fit for your application. The Communications Framework can be implemented on the UART framework interface and uses at least one UART HAL module to create a UART application. It is a generic API for communications applications using the ThreadX RTOS. The Console Framework is a general API for console command-line interface (CLI) applications using the ThreadX RTOS. The implementation can also use a UART as the communications interface. References to these and other useful resources can be found as described in the References section of this document.

12. UART HAL 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 r_sci_uart 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/R_SCI_UART_Module_Guide_Resources.

Website and Support

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

Technical Contact Details:

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb 24, 2017	-	Initial Release
1.01	Sep 15, 2017	14	Update to Hardware and Software Resources Table

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