

Renesas Synergy™ Platform

I²C Slave HAL 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 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 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 I²C slave on RIIC Slave HAL Module provides a high-level API for I²C slave applications and is implemented on `r_riic_slave`. The I²C slave on RIIC Slave Module uses the RIIC peripheral on the Synergy MCU. Callbacks are provided for transmit complete and receive complete notification.

Contents

1. I ² C Slave HAL Module Features	2
2. I ² C Slave HAL Module APIs Overview	2
3. I ² C Slave HAL Module Operational Overview	3
3.1 I ² C Slave HAL Module Important Operational Notes and Limitations	3
3.1.1 I ² C Slave HAL Module Operational Notes	3
3.1.2 I ² C Slave HAL Module Limitations	3
4. Including the I ² C Slave HAL Module in an Application	3
5. Configuring the I ² C Slave HAL Module	4
5.1 I ² C Slave HAL Module Clock Configuration	5
5.2 I ² C Slave HAL Module Pin Configuration	5
6. Using the I ² C Slave HAL Module in an Application	6
7. The I ² C Slave HAL Module Application Project	6
8. Customizing the I ² C Slave HAL Module for a Target Application	9
9. Running the I ² C Slave HAL Module Application Project	10
10. I ² C Slave HAL Module Conclusion	10
11. I ² C Slave HAL Module Next Steps	11
12. I ² C Slave HAL Module Reference Information	11

1. I²C Slave HAL Module Features

- Support for I²C Slave operations
- Support transactions with a I²C master device
 - Read
 - Write
- Callback support
 - Transmit complete (number of bytes transmitted provided)
 - Receive complete (number of bytes received provided)

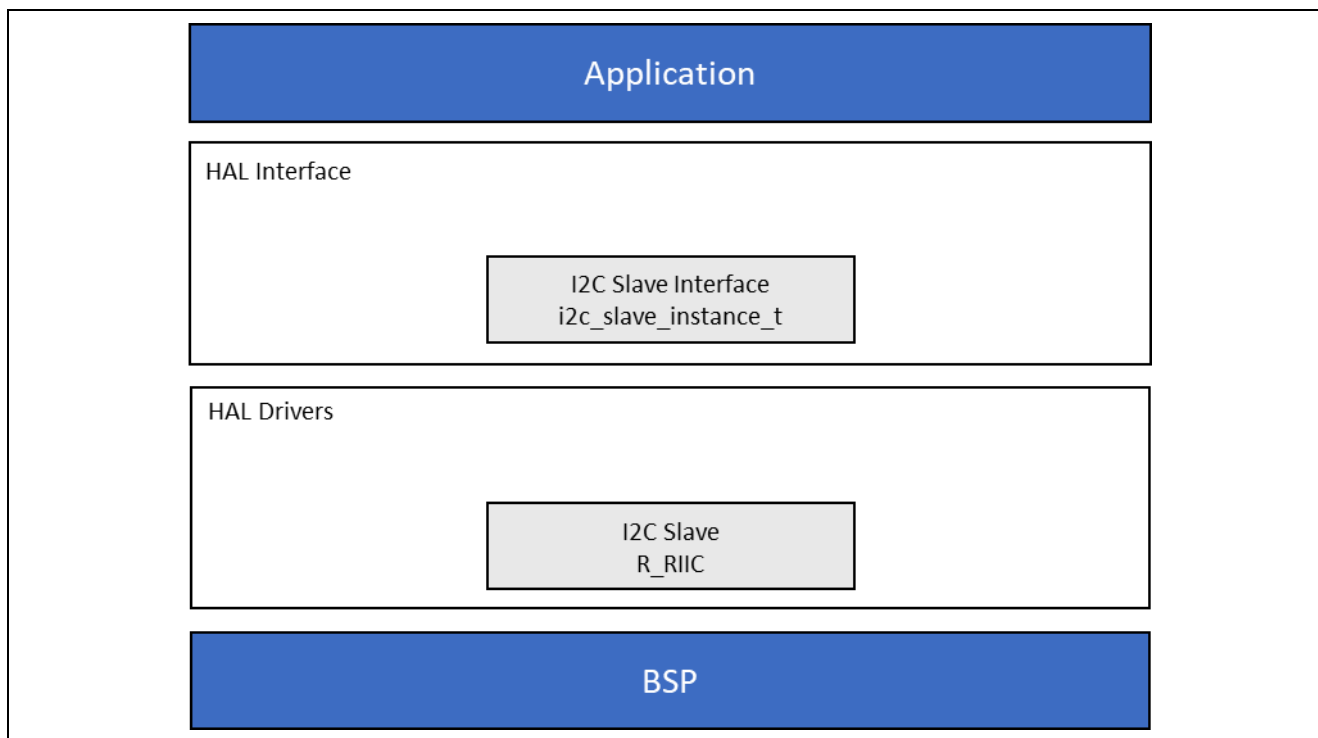


Figure 1. I²C Slave HAL Module Block Diagram

2. I²C Slave HAL Module APIs Overview

The I²C RIIC Slave HAL Module defines APIs for reading and writing to a master I²C device. The following table includes a complete list of the available APIs, an example API call and a short description of each API. Following the API summary table is a table of status return values.

Table 1. I²C Slave HAL Module API Summary

Function Name	Example API Call and Description
<code>.open</code>	<code>g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg);</code> Open the instance and initialize the hardware.
<code>.close</code>	<code>g_i2c.p_api->close(g_i2c.p_ctrl);</code> Closes the driver and releases the I ² C device.
<code>.masterWriteSlaveRead</code>	<code>g_i2c.p_api->masterWriteSlaveRead(g_i2c.p_ctrl, &destination, bytes);</code> Performs a read operation on an I ² C device.
<code>.masterReadSlaveWrite</code>	<code>g_i2c.p_api->masterReadSlaveWrite(g_i2c.p_ctrl, &source, bytes, restart);</code> Performs a write operation on an I ² C device.
<code>.versionGet</code>	<code>g_i2c.p_api->versionGet(&version);</code> Retrieve the API version with the version pointer.

Note: For detail 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	API Call Successful
SSP_ERR_INVALID_POINTER	Pointer is NULL
SSP_ERR_IN_USE	Attempted to open an already open device instance.
SSP_ERR_ABORTED	Device was closed while a transfer was in progress.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value

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. I²C Slave HAL Module Operational Overview

The I²C slave on the RIIC Slave HAL Module supports transfers to an I²C Master device. Callbacks are provided to interrupt the CPU when a data transmit or receive have been completed.

3.1 I²C Slave HAL Module Important Operational Notes and Limitations

3.1.1 I²C Slave HAL Module Operational Notes

- The RIIC Error (EEI), Receive Buffer Full (RXI), Transmit Buffer Empty (TXI) and Transmit End (TEI) interrupts for the selected channel used must be enabled in the Board Support Package (BSP), even when the user wants to use callbacks.
- Setting the interrupts to different priority levels could result in improper operation.

3.1.2 I²C Slave HAL Module Limitations

This is the initial version of I²C RIIC Slave Driver with only basic functionality implemented. The following limitations are known:

The Driver locks up the I²C bus when any of the following operations occur:

When Master is Reading **M** bytes and Slave has **N** bytes available to write. In this case ($M < N$) or ($M > N$).

When Master and Slave are both writing to the bus at same time.

When Master and Slave are both reading from the bus at same time.

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

4. Including the I²C Slave HAL Module in an Application

This section describes how to include the I²C Slave HAL 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, see the *SSP User's Manual* to learn how to manage each of these important steps in creating SSP-based applications.

To add the I²C Slave 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 I²C RIIC HAL Module is `g_i2c0`. This name can be changed in the associated Properties window.

Table 3. I²C Slave HAL Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I ² C Slave Driver on r_riic_slave	Threads	New Stack> Driver> Connectivity> I2C Slave Driver on r_riic_slave

The following image shows when the I²C Slave HAL Module on `r_riic_slave` is added to the Thread Stack, the configurator automatically adds any needed lower level drivers. In this case, none are needed.

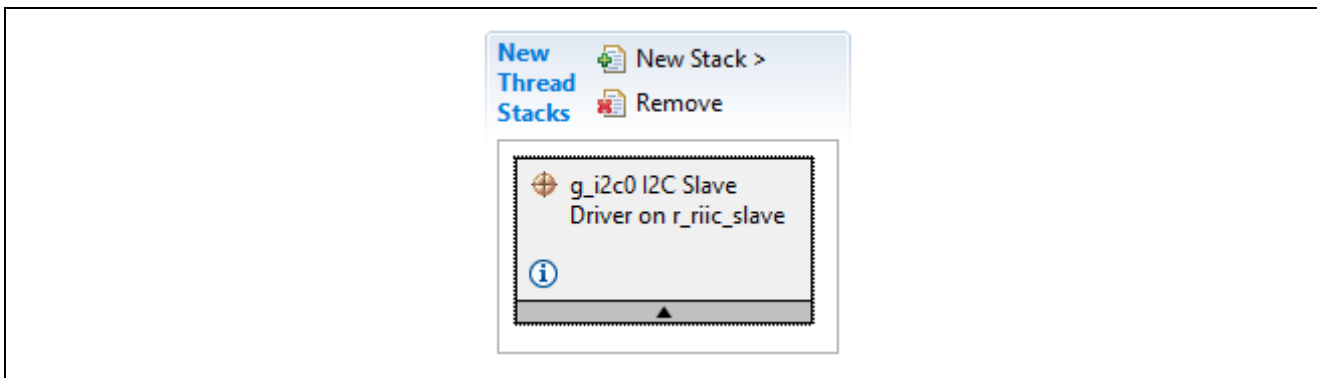


Figure 2. I²C Slave HAL Module Stack

5. Configuring the I²C Slave HAL Module

You must configure the I²C Slave HAL module 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 to enable successful operations. Only properties that can be changed without causing conflicts are available for modification. Other properties are **locked** and not available for changes. These 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 are listed in a table in this document for easy reference.

One of the properties most often identified as requiring a change is the interrupt priority. This configuration setting is available with the Properties window of the associated module. Simply select the indicated module to view it in the Properties window. The interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Note that the interrupt priorities listed in the Properties window in the ISDE include an indication as to the validity of the setting based on the MCU targeted (CM4 or CM0+.) This level of detail is not included in configuration properties tables, but is visible in 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 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 I²C Slave HAL Module on r_riic_slave

ISDE Property	Value	Description
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	Selects if code for parameter checking is to be included in the build
Name	Default: g_i2c0	Module name
Channel	0 – 2 (Default: 0)	Specify the IIC channel to be used with this configuration
Rate	Standard, Fast-mode, Fast-mode plus (Default: Standard)	Transfer rate to which the IIC peripheral should be configured to operate
Slave Address	0x00	Slave address
Address Mode	7-bit, 10-bit (Default: 7-bit)	Address mode

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t
Receive Interrupt Priority	Priority 0 (highest)-15 (lowest), Disabled (Default: 2)	Receive Interrupt priority
Transmit Interrupt Priority	Priority 0 (highest)-15 (lowest), Disabled (Default: 2)	Transmit Interrupt priority
Error Interrupt Priority	Priority 0 (highest)-15 (lowest), Disabled (Default: 2)	Error Interrupt priority

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

In some cases, settings other than the defaults for stack modules can be desirable. For example, it might be useful to select different slave addresses or address mode.

5.1 I²C Slave HAL Module Clock Configuration

The RIIC peripheral module uses PCLKB as its clock source.

5.2 I²C Slave HAL Module Pin Configuration

The RIIC 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 lists the method for selecting the pins within the SSP configuration window and the subsequent table has an example of pin selection.

Note: For some peripherals, the operation mode selection determines what peripheral signals are available and thus what MCU pins are required.

Table 5. Pin Selection Sequence for I²C Slave HAL Module

Resource	ISDE Tab	Pin selection Sequence
IIC	Pins	Select Peripherals > Connectivity: IIC > IIC0

Note: The selection sequence assumes IIC0 is the desired hardware target for the driver.

Table 6. Pin Configuration Settings for I²C Slave HAL Module

Property	Value	Description
Pin Group Selection	_A only, _B only, Mixed (Default: _A only)	Pin group selection
Operation Mode	Enabled, Disabled (Default: Disabled)	Enable or disable peripheral module
SDA	None, P401, P407 (Default: None)	SDA Pin
SCL	None, P400, P204 (Default: None)	SCL Pin

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

6. Using the I²C Slave HAL Module in an Application

The typical steps in using the I²C RIIC Slave HAL Module in an application are:

1. Initialize and open the I²C Slave HAL Module using the `open` API
2. Transfer data to the master using the `masterReadSlaveWrite` API
3. Receive data from the master using the `masterWriteSlaveRead` API
4. Close the channel using the `close` API

The following figure illustrates common steps used to communicate with a slave device in a typical operational flow.

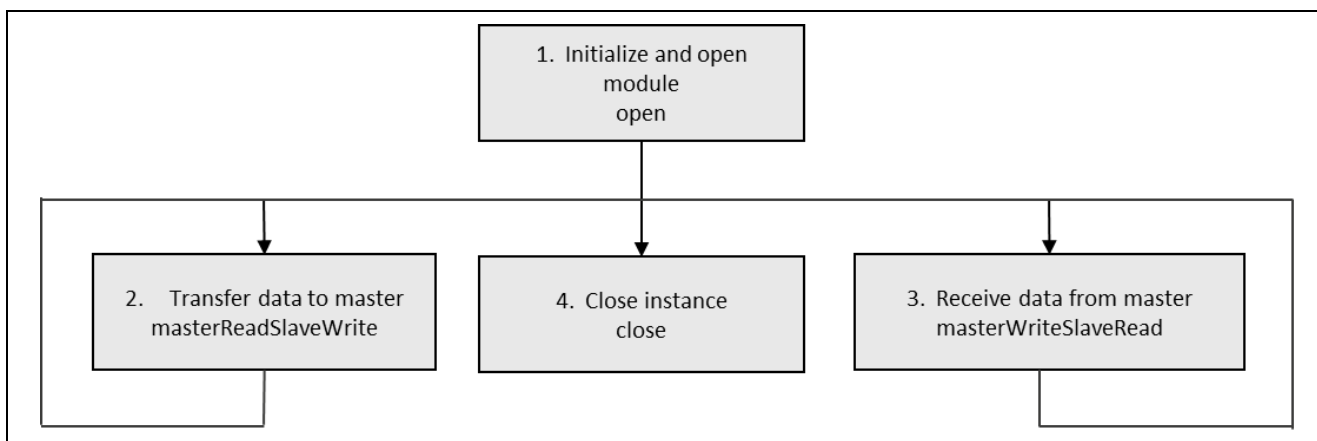


Figure 3. Typical Operational Flow for the I²C Slave HAL Module Application

7. The I²C Slave HAL Module Application Project

The Application Project demonstrates the typical use of the I²C RIIC slave HAL Module APIs. This application project uses the `r_riic_slave` module and channel 0 for I²C communication. The pin setting for the I²C Slave is P400 for SCL and P401 for SDA. Because there must be a master to initiate the data transfer, this application project also contains a `r_riic` master module that uses channel 2. The pin settings for the I²C Master are P512 for SCL and P511 for SDA.

Note that the I²C bus requires pull-up resistors on the clock and data lines. Without the pull-up resistors, the operation of the I²C bus is undetermined. Pins P512 and P511 have the required pullups on S7G2-SK board ensuring the correct I²C bus configuration. Connect the corresponding clock and data lines, P400 to P512 and P401 to P511 before running the included application project.

Table 7. S7G2-SK Board Setup

I ² C Slave	I ² C Master	Description
SCL P400 (J21)	SCL P512 (J22)	I ² C Clock
SDA P401 (J21)	SDA P511 (J22)	I ² C Data

It can be helpful to open the application project in the ISDE and locate these settings from the Pin Configuration tab. These signals can also be located on the SK-S7G2 MCU board schematic as a check on the validity of the selected pins for the I²C signals.

Table 8. Software and Hardware Resources used by the Application Project

Resource	Revision	Description
e ² 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

A simple flow diagram of the application project is provided in the following figure:

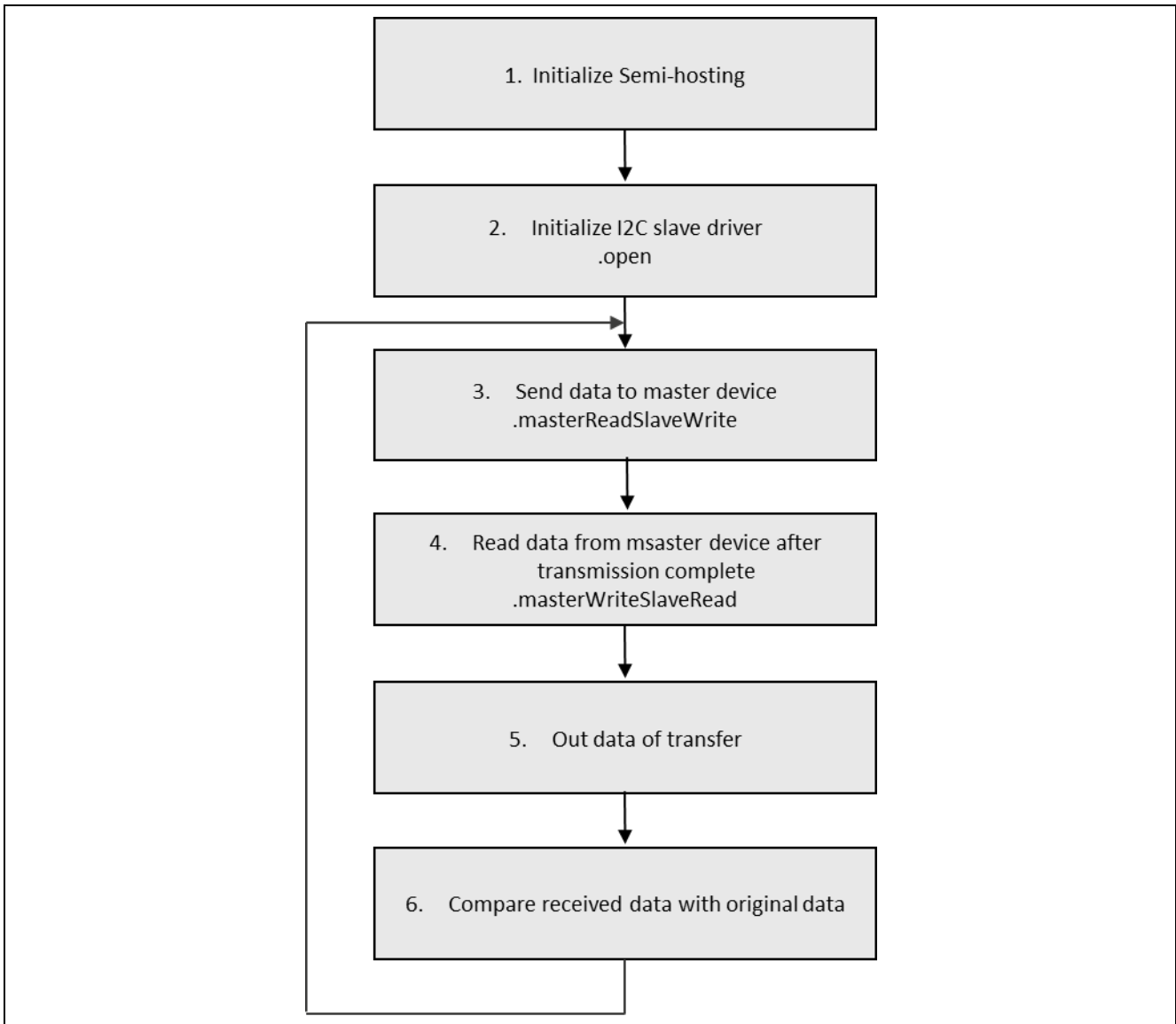


Figure 4. Detailed flow chart of I²C Slave HAL Module Application Project

The Application Project first calls `hal_entry()` and then it calls the `i2c_riic_slave_hal_mg()` in `i2c_riic_slave_hal_mg.c` file. The first section of the `i2c_riic_slave_hal_mg.c` has semi hosting definitions to display results using `printf()` function. The Application Project turns off all LEDs and opens the master and slave I²C SSP instances. Then it initializes the master receive, slave receive and slave transmit buffers. If the semi hosting is turned on, the contents of the buffers are printed out. The project uses `masterReadSlaveWrite` API for slave device on channel 0 to write data to master device. To complete the data transfer project also uses `read()` API for master device on channel 2.

The slave reads the data it just sent with `masterWriteSlaveRead` API. Again master imitates the data transfer with `write()` API sending the data it received from the slave. The project prints out the buffers after the data transfer if semi hosting is enabled. Then the project compares the transmitted and received data by the slave. The LEDs on the SK-S7G2 board blinks indicating that there are no errors and project continuously writes to master and reads back from it. In case of an SSP API error or a mismatch of the transmitted data to the received, red LED turns on and the project enter into a `while(1)` loop.

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

Table 9. I²C RIIC Slave HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_i2c_slave
Channel	0
Rate	Standard
Slave Address	0x01
Address Mode	7-bit
Callback	i2c_slave_callback
Receive Interrupt Priority	Priority 2
Transmit Interrupt Priority	Priority 2
Error Interrupt Priority	Priority 2

For accessing a channel or pin, the I²C 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 I²C pin.

Table 10. Pin Selection Sequence for I²C RIIC Slave HAL Module

Resource	ISDE Tab	Pin selection Sequence
I ² C	Pins	Select Peripherals > Connectivity: IIC > IIC0

Table 11. Pin Configuration Settings for I²C RIIC Slave HAL Module on r_riic_slave

Pin Configuration Property	Value
Pin Group Selection	_A only
Operation Mode	Enabled
SDA	P401
SCL	P400

Note: The example values are for a project using the S7G2 Synergy MCU Group and the SK-S7G2 Kit. Other Synergy MCUs and Kits may have different pin configuration settings available. Also, P401 is set to CAN0_CTS of Connectivity: CAN->CAN0 by default, and P400 is set to GPIO. Disable CAN0 operation mode and P400 GPIO mode, and then reassign these pin as I²C. The following figure shows how these pins are used.

Master configuration settings are shown in the tables below.

Table 12. I²C RIIC Master HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_i2c_master
Channel	2
Rate	Standard
Slave Address	0x01
Address Mode	7-bit
Timeout Mode	Short Mode
Callback	i2c_master_callback
Receive Interrupt Priority	Priority 2
Transmit Interrupt Priority	Priority 2
Transmit End Interrupt Priority	Priority 2
Error Interrupt Priority	Priority 2

Table 13 Pin Selection Sequence for I²C RIIC Master HAL Module

Resource	ISDE Tab	Pin selection Sequence
I ² C	Pins	Select Peripherals > Connectivity: IIC > IIC2

Table 14. Pin Configuration Settings for I²C RIIC Master HAL Module on r_riic_slave

Pin Configuration Property	Value
Pin Group Selection	_A only
Operation Mode	Enabled
SDA	P511
SCL	P512

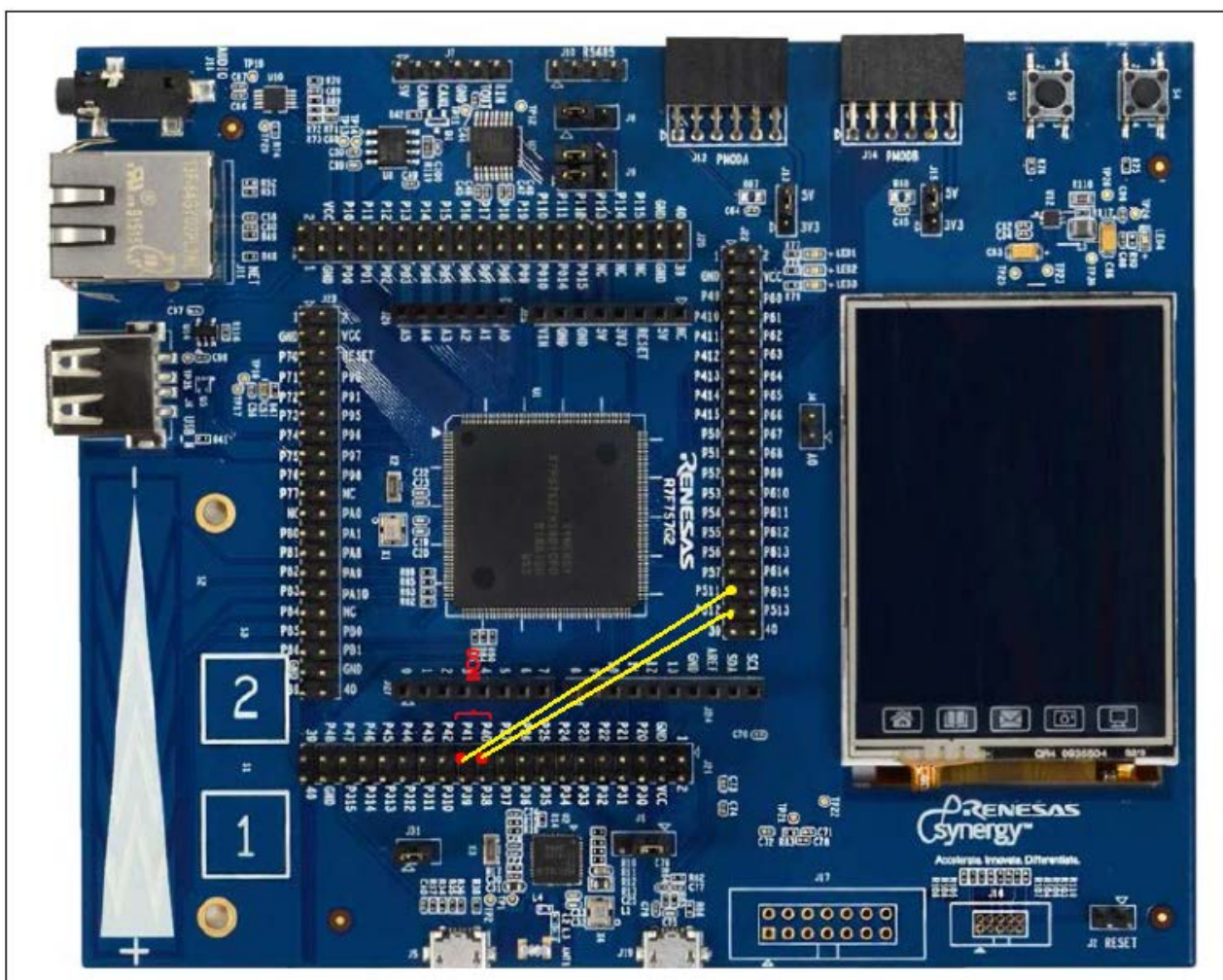


Figure 5. Hardware connection picture

8. Customizing the I²C Slave HAL Module for a Target Application

Some configuration settings can be changed by the developer from those normally shown in the application project. For example, you can easily change the I²C clock configuration settings for the by updating the PCLKB in the **Clock** tab. You can also change the IIC port pins to select the desired port. These changes can be done using the **Pins** tab in the configurator. The I²C RIIC Slave application project uses channel 0, and it can also be connected to an external assistant master device to establish I²C communication.

9. Running the I²C Slave HAL Module Application Project

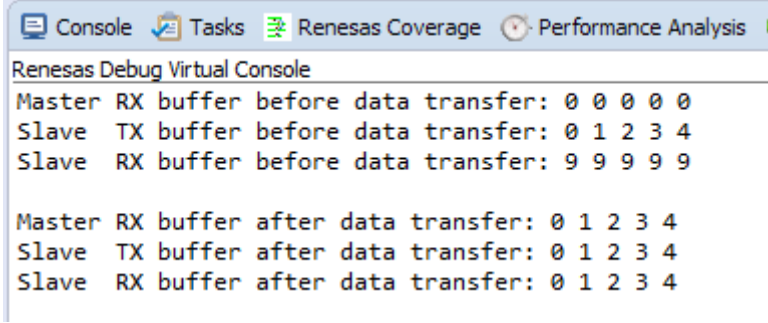
To run the I²C Slave HAL module application project and to see it executed on a target kit, you can simply import it into your ISDE, compile it, and then run the debug operation. See **Importing a Renesas Synergy Project**, in the *SSP Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf, that is included in this package) for instructions on importing the project into Synergy e² studio ISDE or IAR EW for Synergy used to build and run the application.

To implement the I²C Slave HAL module application in a new project, use the follow steps to define, configure, and auto-generate files, as well as add code, compile and debug the target kit. This **hands-on** approach can help make the development process with the SSP more practical, while just reading over the 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, see the *SSP User's Manual* for a description of how to accomplish these steps.

To create and run the I²C Slave HAL module application project, follow these steps:

1. Create a new Renesas Synergy™ project for the S7G2-SK Synergy MCU Group called **I²C_RIIC_SLAVE_HAL_MG_AP**.
2. When creating a project, select BSP in the Project Template Selection page, and then finish a new project setup.
3. Select the **Threads** tab -> **HAL/Common**.
4. Add the I²C Slave Driver to the HAL/Common stack.
5. Configure the parameters.
6. Click on the **Generate Project Content** button.
7. Add the code from the supplied project file `i2c_riic_slave_hal_mg.c`, `i2c_riic_slave_hal_mg.h`, and `hal_entry.c`, or copy over the files.
8. Compile the project.
9. Connect to the host PC using the USB cable (use J19 DEBUG_USB connector).
10. Start to debug the application.
11. The output can be viewed in the Renesas Debug Virtual Console.



```
Console Tasks Renesas Coverage Performance Analysis
Renesas Debug Virtual Console
Master RX buffer before data transfer: 0 0 0 0 0
Slave TX buffer before data transfer: 0 1 2 3 4
Slave RX buffer before data transfer: 9 9 9 9 9

Master RX buffer after data transfer: 0 1 2 3 4
Slave TX buffer after data transfer: 0 1 2 3 4
Slave RX buffer after data transfer: 0 1 2 3 4
```

Figure 6. Example Output from I²C Slave HAL Module Application Project

The output above is for e² studio. To verify the project with IAR EW for Synergy, break to execution of the code and examine the master and slave buffers.

10. I²C Slave HAL Module Conclusion

This module guide has provided all the background information needed to select, add, configure, and use the components in an example application 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 common errors, like conflicting configuration settings or the incorrect selection of lower level drivers. The use of high-level APIs (as demonstrated in the application project) shows 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. I²C Slave HAL Module Next Steps

After you have mastered a simple I²C RIIC Slave Module project you may want to review a more complex example. Another module guide named [I²C SCI HAL Module Guide](#) is also provided for using a master device on `r_sci_i2c` to realize I²C function. You also may find the I²C Framework, which supports the SCI and IIC peripherals and is implemented as [sf_i2c](#).

12. I²C Slave HAL Module Reference Information

SSP User's Manual: Available in html format in the SSP distribution package and as a pdf from the www.renesas.com/synergy/ssp.

Links to all the most up-to-date [r_riic_slave](#) module reference materials and resources are available on the Synergy Knowledge Base.

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	Apr.27.17	—	Initial Release
1.01	Sep.12.17	—	Update to Hardware and Software Resources Table
1.02	Oct.24.17	—	Editing and release
1.03	Oct.16.18	—	Update to SSP 1.5.0
1.04	Apr.29.19	—	Updated for SSP 1.6.0

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/.