Renesas Synergy™ Platform

I2C Master HAL Module Guide

R11AN0148EU0101
Rev. 1.01
Sep 5, 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 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 Master on RIIC HAL module is a high-level API for I2C Master applications and is implemented on `r_riic`. The I2C Master RIIC module uses the IIC peripheral on the MCU Synergy Group. Callbacks are provided for transmit complete and receive complete event notification.

The intended audience are developers who want to develop an application that uses USBX Mass Storage Class (Device) to enable quick and easy file transfer between devices using MCU Synergy Groups.

Contents

1. I2C Master HAL Module Features

- Support for I2C RIIC operations
  - Standard (100 kHz)
  - I2C fast-mode (400 kHz)
  - I2C fast-mode plus (1 MHz on channel 0 of S7G2 and S5D9 MCU Families)
- Initialization of the RIIC module
- Read from a slave device
- Write to a slave device
- Reset the I2C peripheral
- Set the address of the slave device
- Callback support
  - Transfer aborted
  - Transmit complete (number of bytes transmitted provided)
  - Receive complete (number of bytes received provided)

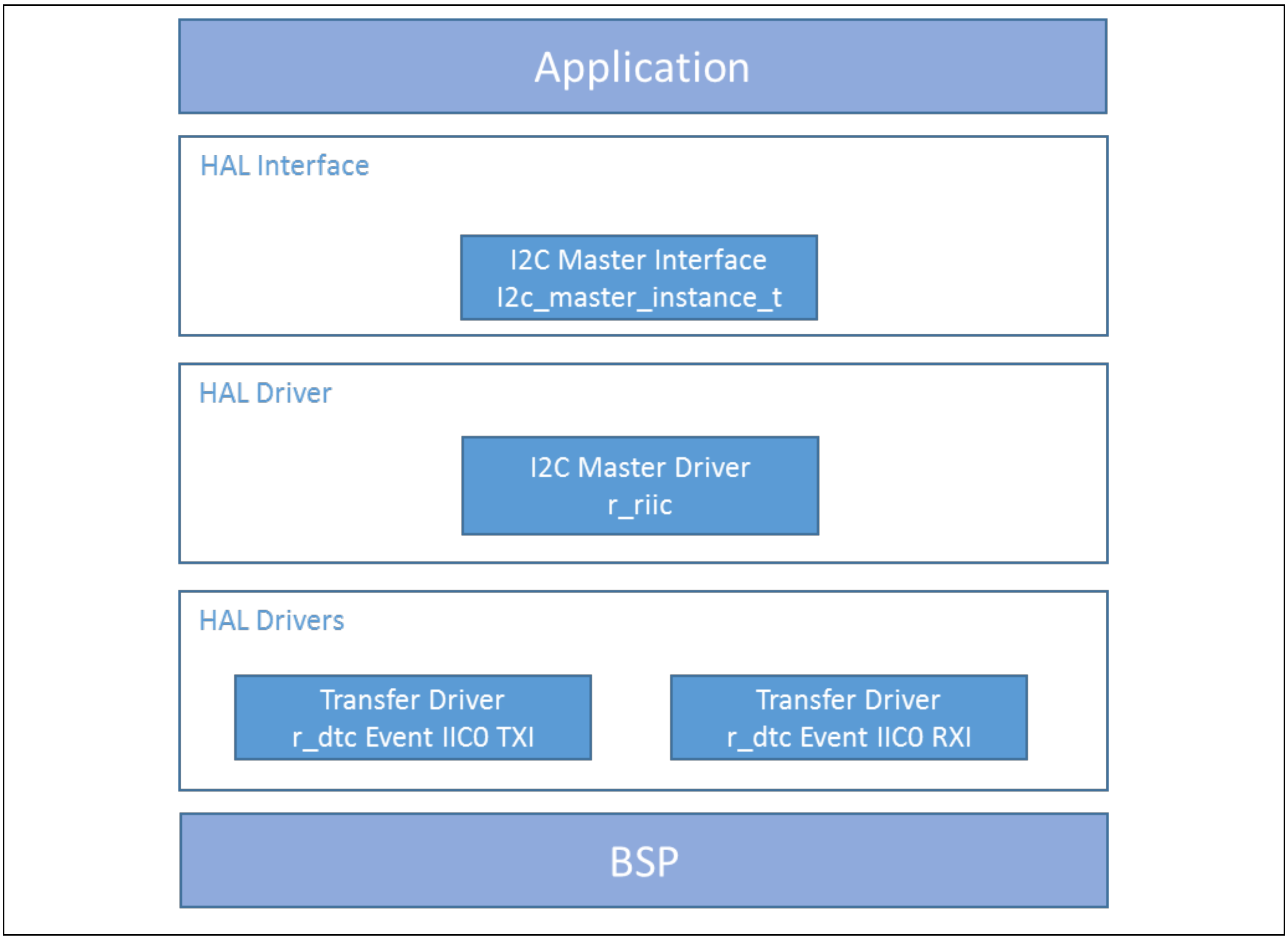

Figure 1. I2C Master HAL Module Block Diagram

2. I2C Master HAL Module APIs Overview

The I2C Master on RIIC (I2C RIIC) HAL module defines APIs including reading and writing using a master I2C device. 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. I2C Master HAL Module API Summary

Function Name	Example API Call and Description
.open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); Open the instance and initialize the hardware.
.close	g_i2c.p_api->close(g_i2c.p_ctrl); Closes the driver and releases the I2C device.
.read	g_i2c.p_api->read(g_i2c.p_ctrl, &destination, bytes, restart); Performs a read operation on an I2C device.
.write	g_i2c.p_api->write(g_i2c.p_ctrl, &destination, bytes, restart); Performs a write operation on an I2C device.
.reset	g_i2c.p_api->reset(g_i2c.p_ctrl); Reset the peripheral.
.versionGet	g_i2c.p_api->versionGet(&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. I2C Master HAL Module API Summary

Function Name	Example API Call and Description
.open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); Open the instance and initialize the hardware.
.close	g_i2c.p_api->close(g_i2c.p_ctrl); Closes the driver and releases the I2C device.
.read	g_i2c.p_api->read(g_i2c.p_ctrl, &destination, bytes, restart); Performs a read operation on an I2C device.
.write	g_i2c.p_api->write(g_i2c.p_ctrl, &destination, bytes, restart); Performs a write operation on an I2C device.
.reset	g_i2c.p_api->reset(g_i2c.p_ctrl); Reset the peripheral.
.versionGet	g_i2c.p_api->versionGet(&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 3. 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
SSP_ERR_INVALID_RATE	The requested rate cannot be set

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 Master HAL Module Operational Overview

The I2C master on RIIC HAL module supports transactions with an I2C Slave device. Callbacks are provided to interrupt the CPU when a transmission or receive has been completed. The RIIC HAL module invokes the callback with the argument `i2c_callback_args_t`, indicating the number of received or transmitted bytes in buffer, pointer to user provided context, and the event `i2c_event_t`.

3.1 I2C Master HAL Module Important Operational Notes and Limitations

3.1.1 I2C Master HAL Module Operational Notes

Interrupts

- 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 properties of the selected device irrespective of whether the user wants to use callbacks.
- Setting the interrupts to different priority levels could result in improper operation.

IIC Rate Calculation

- The I2C Master module calculates the internal baud-rate setting based on the configured transfer rate and passed to open. The closest possible baud-rate that can be achieved (less than or equal to the requested rate) at the current PCLKB settings is calculated and used.
- If a valid clock rate could not be calculated, an error is returned.

Triggering DMAC/DTC with the IIC

- DTC transfer support added by default in the configurator. This can be removed for CPU transfer cases. The DTC is configured in the module. No user configuration is required for this.
- DMA transfer is not supported.

Triggering ELC Events with the IIC

- The I2C Master module can trigger the start of other peripherals. See events and peripheral definitions in the *ELC User's Guide* for further information.

Multiple Devices on the Bus

- If multiple devices are connected on the same bus, only one device can be opened at a time.

3.1.2 I2C Master HAL Module Limitations

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

4. Including the I2C Master HAL Module in an Application

This section describes how to include the I2C RIIC 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 I2C Master 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 4. I2C Master HAL Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I2C Master Driver on r_riic	Threads	New Stack> Driver> Connectivity> I2C Master Driver on r_riic

Note: When the I2C RIIC HAL module on `r_riic` is added to the thread stack as shown in the following figure, the configurator automatically adds any needed lower-level drivers.

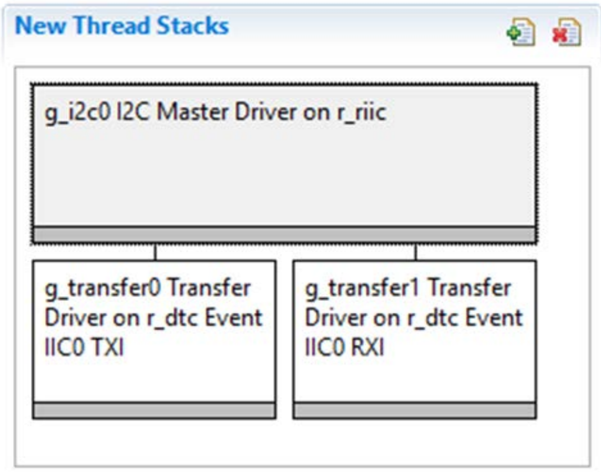

Figure 2. I2C Master HAL Module Stack

5. Configuring the I2C Master HAL Module

The I2C RIIC HAL module must be configured by you 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 operations. 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 table later 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 and then view the **Properties** window. The interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Note that the interrupt priorities listed in the Properties window in the ISDE will include an indication as to the validity of the setting based on the MCU targeted (CM4 or CM0+). This level of detail is not included in the following configuration Properties tables, but is easily visible with the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE, create the module and explore the property settings in parallel with looking over the following configuration table 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 5. Configuration Settings for the I2C Master HAL Module on r_riic

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_i2c0	Module name.
Channel	0, 1, or 2	Specify the IIC channel to be used with this configuration.
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	Standard, Fast, and Fast-plus. (See IIC Rate Calculation.)
Slave Address	0x00	Set the address of the slave device the I2C master will be communicating with.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
Callback	NULL	A user callback function can be registered in i2c_api_master_t::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. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
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: Priority 2	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: Priority 2	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)	Transmit end interrupt priority selection.

	Default: Priority 2	
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: Priority 2	Error 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 6. Configuration Settings for the DTC HAL Module on r_dtc Event IIC0 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	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Normal	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	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 IICI0 TXI	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 ThreadX, CM0+: invalid) Default: Disabled	ELC software event 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 7. Configuration Settings for the DTC HAL Module on r_dtc Event IIC0 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	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	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 IIC0 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 ThreadX, CM0+: invalid) Default: Disabled	ELC software event 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.

In some cases, settings other than the defaults for the module can be desirable. For example, it might be useful to select different slave addresses or address modes. The configurable properties for the lower-level stack modules are given in the following sections for completeness and as a reference.

5.1 I2C Master HAL Module Clock Configuration

The IIC peripheral module uses PCLKB as its clock source. The actual I2C transfer rate will be calculated and set internally by the driver depending on the selected transfer rate. If the PCLKB is configured in such a manner that the selected internal rate cannot be achieved, an error will be returned when initializing the driver.

5.2 I2C Master HAL Module Pin Configuration

The IIC peripheral module uses pins on the MCU to communicate to external devices. I/O pins must be selected and configured as required by the external device. The following table illustrates the method for selecting the pins within the SSP configuration window and the subsequent table illustrates an example selection for the pins.

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

Table 8. Pin Selection Sequence for I2C Master 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 9. Pin Configuration Settings for I2C Master HAL Module

Pin Configuration 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 and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

6. Using the I2C Master HAL Module in an Application

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

1. Initialize and open the I2C RIIC HAL Module using the open API
2. Transfer data to the slave using the write API
3. Receive data from the slave using the read API
4. Reset the instance with the reset API (if needed)
5. Close the channel using the close API

Note: If the application wants to switch the device without opening and closing the bus, use the `slaveAddressSet` API where `g_i2c.p_ctrl` is the same control instance that was used in the last opened device. The module will use the same bus configuration to communicate with the new device. In this case, you can use the same control instance to communicate with different slave devices by setting the new slave address and calling the read or write APIs.

These common steps to communicate with a slave device are illustrated in a typical operational flow in the figure below:

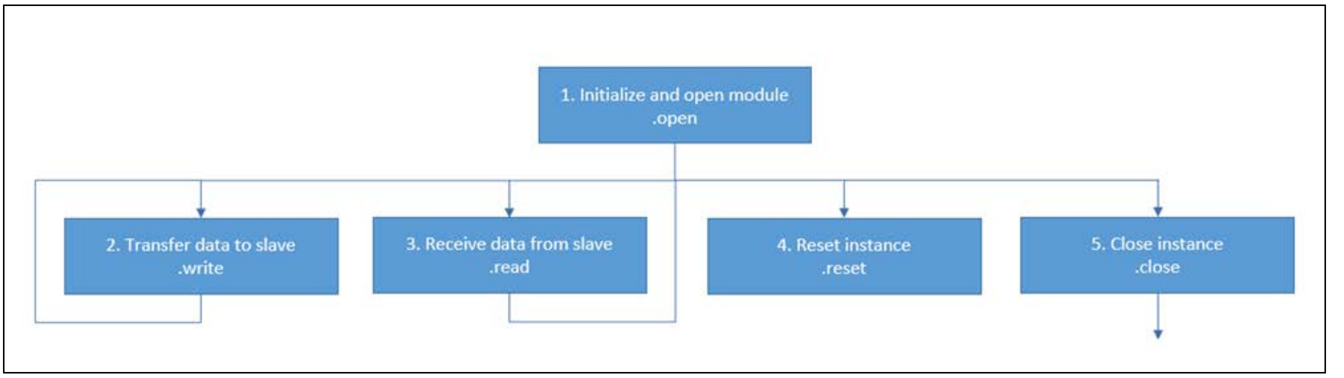

Figure 3. Flow Diagram of a Typical I2C Master HAL Module Application

7. The I2C Master HAL Module Application Project

The application project associated with this module guide demonstrates the steps in an example application. You may want to import and open the application project within the ISDE and view the configuration settings for the I2C HAL module.

Table 10. Application Project Configuration Settings (Changed from the Defaults)

Resource	ISDE Property	Property / Configuration Setting
g_i2c I2C Master Driver on r_riic_i2c	Name	g_i2c
	Slave Address	0x48
	Channel	2
	Receive Interrupt Priority	Priority 3
	Transmit Interrupt Priority	Priority 3
	Transmit End Interrupt Priority	Priority 3
	Error Interrupt Priority	Priority 3
Pins tab > Pin Selection > Peripherals > Connectivity: IIC > IIC2	Pin Group Selection	A only
	Operation Mode	Enable
	SDA	P511
	SCL	P512
Pins tab > Pin Selection > Ports > P4 > P511 & Pins tab > Pin Selection > Ports > P4 > P512	Mode	Peripheral Mode
	Pull up	None
	IRQ	None
	Drive Capacity	Low
	Output Type	n-ch open drain
Pins tab > Pin Selection > Ports > P6 > P609	Mode	Output Mode (Initial High)
	Pull up	None
	Drive Capacity	Low
	Output Type	CMOS

The application project demonstrates the typical use of the I2C HAL module APIs. The configuration settings in the application project need to be customized for the specifics of the target kit and MCU. The application project uses the `r_riic` module and uses channel 2 for I2C communication. The output pins for I2C communication are selected to conform to the signal connections from the touch controller (P512 for SCL and P511 for SDA.) It can be helpful to open the application project in the ISDE and locate these settings in the PIN configuration tab. These signals can also be located on the schematic for the SK-S7G2 board as a check on the validity of the selected pins for the I2C signals. The external slave reset signal is connected to GPIO pin P609 and must be enabled and configured for proper operation. All these application projects' specific settings are given in the preceding table.

Table 11. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	5.3.1 or later	Integrated Solution Development Environment
SSP	1.2.0 or later	Synergy Software Platform
IAR EW for Synergy	7.71.2 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

Once the I2C HAL module application project has been successfully added and configured, it can be used by the application program. The I2C application project implements steps similarly to those shown for the general case; the key difference is that the read and write functions implement specific program functions to initialize, configure, and read data from the I2C slave device. The overall program flow is illustrated in the figure below:

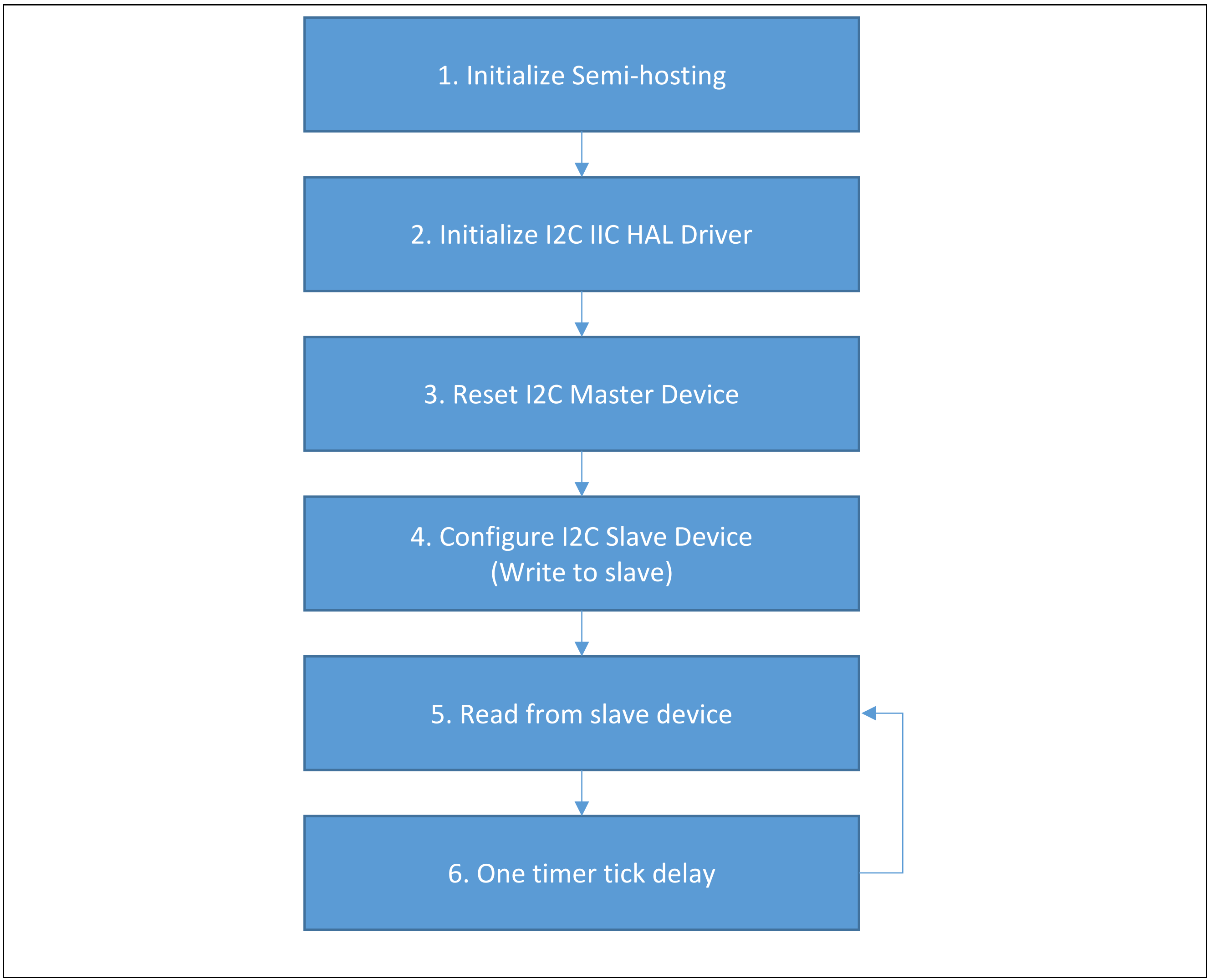

Figure 4. Detailed flow chart of I2C Master HAL Module Application Project

After importing the application project into the ISDE, you can read through the code in `i2c_hal.c` to follow along with the flow outlined in the figure above. The first section of `i2c_hal.c` are the header files which reference the generated I2C instance structure.

The next section contains macro definitions and the semi hosting support for `printf()`function. Then the project opens the I2C interface and resets and configures the I2C slave device. The read operation is called continuously in the main loop to read a status register from the I2C slave device. You can see the read values change on the Renesas Debug Console if you touch the LCD panel. If there are any errors from SSP API calls, the red LED is turned on and the project goes into a `while(1)` infinite loop.

Semi-hosting is a common technique used to display results using `printf()`. The application project supports semi-hosting if semi-hosting macro is uncommented in the `i2c_hal.c` file.

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

8. Customizing the I2C Master 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, you can easily change the configuration settings for the I2C rate. You can also add more slaves to the I2C bus and use different instance of I2C HAL drivers to address that slave by just changing the slave address and instance name. You can also use APIs to change the slave-address in at the run time with the same bus configuration and control data structure. I2C HAL configuration also provides flexibility to use 7 bit or 10 bit addressing mode and callback functions for user-defined interrupt handling.

9. Running the I2C Master HAL Module Application Project

To run the I2C 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 *SSP Import Guide* (11an0023eu0117-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 I2C HAL module application in a new project, follow the steps below for defining, configuring, auto-generating files, adding code, compiling and debugging on the target kit. This is a hands-on approach that can help make the development process with 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, 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 I2C RIIC HAL application project, simply follow these steps:

1. Import the attached application project I2C_HAL to e2 studio or IAR embedded workbench.
2. Compile the application without errors or warnings.
3. Connect to the host PC via a micro USB cable to J19 on SK-S7G2 board.
4. Start to debug the application.
5. LED1-3 will blink when communication is ongoing. If the semi-hosting macro is uncommented in `i2c_hal.c` , the output can be viewed in the Renesas Debug Console as seen in the following figure. Upon touching the touch screen, values of the received data in the console will change.

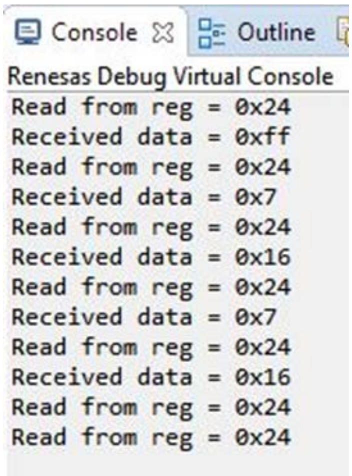

Figure 5. Example Output from I2C Master HAL Module Application Project

10. I2C Master 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 the 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) 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. I2C Master HAL Module Next Steps

After you have mastered a simple I2C HAL Driver application project, you may want to review a more complex example. The I2C Framework is a set of ThreadX®-aware Framework APIs. The I2C Framework handles the integration and synchronization of multiple I2C peripherals on the I2C bus. With the I2C Framework, you can create one or more I2C buses and connect multiple I2C peripherals to each I2C bus. The I2C Framework uses a single interface to access both SCI I2C and RIIC drivers. You can learn more about the I2C Framework by reading the associated module guide listed in the References section at the end of this document.

12. I2C Master HAL Module 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 `r_riic_master` module reference materials and resources are available on the Renesas Synergy Knowledge Base: https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/R_RIIC_Master_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	May 15, 2017	-	Initial Release
1.01	Sep 5, 2017	-	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