

I²C Framework Module Guide

Introduction

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

The I²C Framework module is a ThreadX-aware high-level API for I²C Framework applications and is implemented on `sf_i2c`. The I²C HAL module configures the I²C peripheral to enable serial communication to be used by the framework. The I²C Framework module uses the I²C and SCI peripherals on the Synergy MCU.

Contents

1. I ² C Framework Module Features	2
2. I ² C Framework Module APIs Overview	2
3. I ² C Framework Module Operational Overview	4
3.1 I ² C Framework Module Important Operational Notes and Limitations	4
3.1.1 I ² C Framework Module Operational Notes.....	4
3.1.2 I ² C Framework Module Limitations.....	5
4. Including the I ² C Framework Module in an Application	5
5. Configuring the I ² C Framework Module	6
5.1 Configuration Settings for the I ² C Framework Lower-Level Drivers.....	7
5.2 I ² C Framework Module Clock Configuration	14
5.3 I ² C Framework Module Pin Configuration	14
5.4 I ² C Framework Module Other Settings.....	14
6. Using the I ² C Framework Module in an Application.....	15
7. The I ² C Framework Module Application Project	16
8. Customizing the I ² C Framework Module for a Target Application.....	18
9. Running the I ² C Framework Module Application Project	18
10. I ² C Framework Module Conclusion	19
11. I ² C Framework Module Next Steps	19
12. I ² C Framework Module Reference Information	19

1. I²C Framework Module Features

- ThreadX-aware framework
- Handles integration and synchronization of multiple I²C peripherals on the I²C bus
- Provides a single interface to access both SCI I²C and RIIC drivers
- The I²C framework module configures I²C communication in master mode
- The I²C framework module supports three data rates: 100 kHz, 400 kHz, and 1 MHz
- The I²C framework module supports both 7-bit addressing and 10-bit addressing
- The I²C framework module also provides support for callbacks internally. User defined callback is not used. The callback functions are called with the following events `i2c_event_t`:
 - Transfer aborted
 - Transmit complete
 - Receive complete.
- The callback structure `i2c_callback_args_t` also provides the number of bytes that were sent or received
- Implemented by:
 - Simple I²C on SCI
 - RIIC.

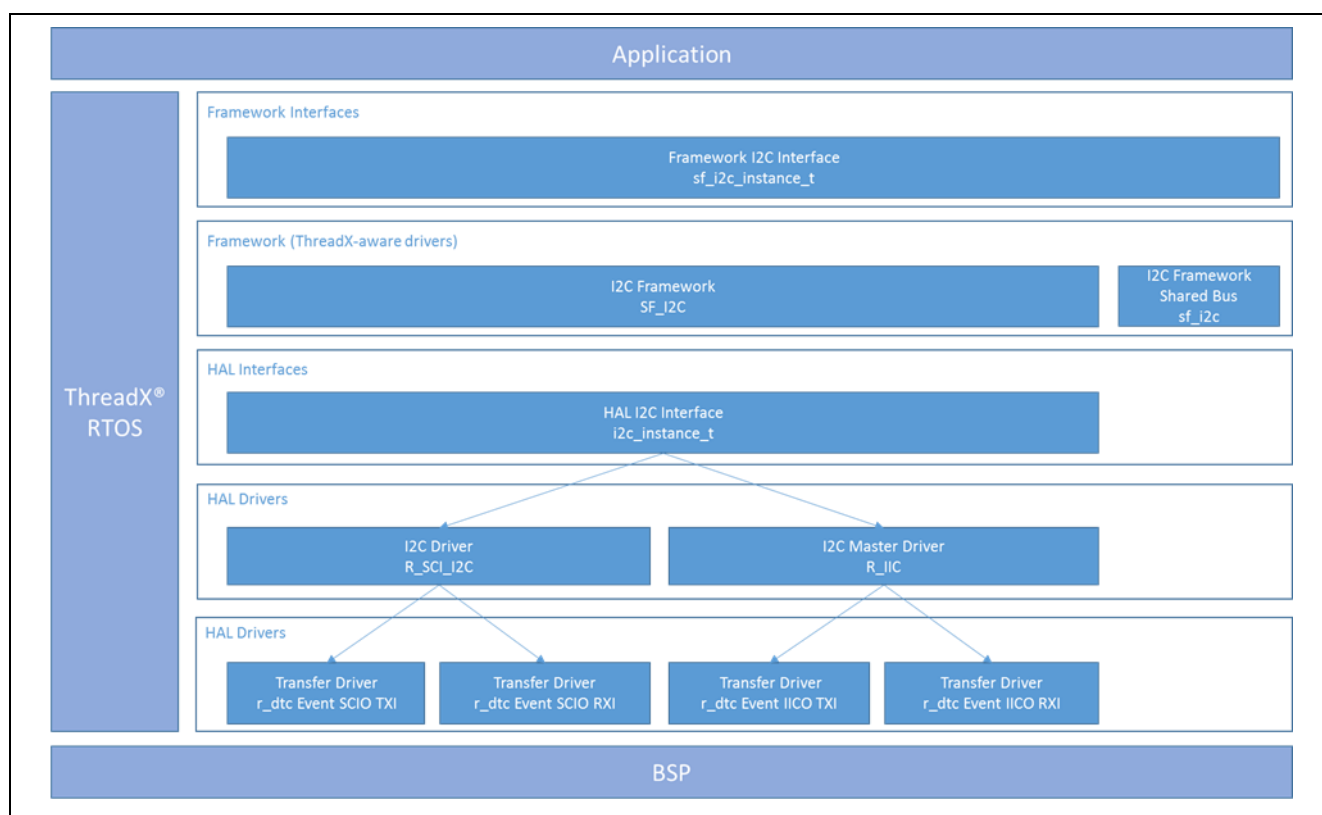


Figure 1 I²C Framework Module Block Diagram

2. I²C Framework Module APIs Overview

The I²C Framework interface defines APIs for opening, closing, reading, writing, locking, unlocking, and resetting the bus using the I²C Framework. 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 Framework Module API Summary

Function Name	Example API Call and Definition
.open	<pre>g_sf_i2c_device.p_api->open(g_sf_i2c_device.p_ctrl, g_sf_i2c_device.p_cfg)</pre> <p>Opens a designated I²C device on the bus.</p>
.close	<pre>g_sf_i2c_device.p_api->close (g_sf_i2c_device.p_ctrl);</pre>

	Disables the I ² C device designated by control handle. Closes the RTOS services used by the bus if no devices are connected to the bus.
.read	<pre>g_sf_i2c_device.p_api->read (g_sf_i2c_device.p_ctrl, &reg, no_of_bytes_to_read, 0, TX_WAIT_FOREVER);</pre> Receives data from I ² C device.
.write	<pre>g_sf_i2c_device.p_api->write (g_sf_i2c_device.p_ctrl, command, no_of_bytes_to_write , false, TX_WAIT_FOREVER);</pre> Transmits data to I ² C device.
.lock	<pre>g_sf_i2c_device.p_api->lock (g_sf_i2c_device.p_ctrl);</pre> Locks the bus for a device. Locking reserves the bus until unlocking and allows several reads and writes without interrupt.
.unlock	<pre>g_sf_i2c_device.p_api->unlock (g_sf_i2c_device.p_ctrl);</pre> Unlocks the bus from a particular device and makes it available for other devices.
.reset	<pre>g_sf_i2c_device.p_api->reset (g_sf_i2c_device.p_ctrl, TX_NO_WAIT);</pre> Aborts any in-progress transfer and forces the I ² C peripheral into ready state.
.versionGet	<pre>g_sf_i2c_device.p_api->versionGet (version);</pre> Retrieves the version information using 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	I ² C function performed successfully
SSP_ERR_INVALID_MODE	Illegal I ² C mode is specified
SSP_ERR_INVALID_CHANNEL	Omitted I ² C channel is specified
SSP_ERR_IN_USE	I ² C channel has already been opened
SSP_ERR_INVALID_ARGUMENT	Argument is not one of the predefined values
SSP_ERR_INVALID_POINTER	Null pointer(s) is (are) given
SSP_ERR_INTERNAL	Internal error has occurred
SSP_ERR_ASSERTION	A critical assertion has failed
SSP_ERR_NOT_OPEN	Device instance not opened
SSP_ERR_TRANSFER_ABORTED	The data transfer was aborted
SSP_ERR_INVALID_RATE	The requested rate cannot be set
SSP_ERR_TIMEOUT	Timeout error occurs

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 Framework Module Operational Overview

The I²C Framework module complies with the layered driver architecture of the SSP. It uses the lower-level I²C HAL modules to communicate with the I²C peripherals and controls the I²C-capable peripherals on a Synergy microcontroller, as configured by a user. With the I²C Framework module, one or more I²C buses can be created and multiple I²C peripherals can be connected to each I²C bus. The I²C Framework module APIs use a ThreadX-mutex to acquire and release the shared bus for I²C slave devices. Acquire and release are implemented by `lock` and `unlock` APIs respectively in the I²C Framework module.

As the I²C framework module configures I²C communication in master mode, this allows the user to:

- Initialize the driver
- Read from a slave device
- Write to a slave device
- Reset the I²C peripheral
- Lock the I²C bus
- Unlock the I²C bus.

The I²C Framework module works with the Synergy MCU I²C hardware modules, the RIIC and SCI HAL modules. Both I²C modules support the I²C fast-mode with bit rates of up to 400 kHz. The IIC peripheral and the RIIC HAL module support fast-mode plus with 1-MHz bit-rates. The module supports only master mode for both implementations.

The I²C Framework module uses a bus and device on bus architecture. Every device is linked to the bus to which it will be connected. The user must configure the framework shared-bus and the lower-level I²C HAL layer for each framework device connecting to the bus. The user can add the existing framework shared-bus when configuring multiple devices on the same bus. A common start and stop procedure is used for all I²C data-transfer operations. Only one device is configured to the lower level and the remaining devices perform read or write operations by switching the device address within the framework.

All I²C Framework devices on the same bus must use the same lower-level configuration settings (for example, the I²C HAL module), except for the slave address and addressing mode. The framework will use the configuration of the first device that it opens in the application to configure the bus. This means that all I²C Framework devices on the same bus must have the same lower-level configuration settings (except for the slave address and addressing mode). If different configurations are used, proper operation cannot be guaranteed.

The I²C Framework supports bus-locking functionality, meaning that the bus can be locked for a given peripheral. The locking allows devices to reserve a bus to themselves for a given period of time (between lock and unlock). This allows devices to complete several reads and writes on the bus without interruption (which is required in some instances).

3.1 I²C Framework Module Important Operational Notes and Limitations

3.1.1 I²C Framework Module Operational Notes

- 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.
- The I²C can trigger the start of other peripherals available from the ELC. See the ELC Module Guide for further information.
- The I²C Framework can support multiple I²C devices on the same bus if the clock rate remains the same for all the devices. That means multiple devices can be opened in the same bus if they are of the same clock rate. If devices have different clock rates, only one device can be opened at a time.
- SDA and SCL output pin type should be n-channel open drain when using I²C on SCI.
- In the I²C Framework configuration, the channel number given to this bus overrides the channel number given in the HAL module.
- Shared bus can be used by multiple slave devices with the respective configuration. The framework also handles mutual exclusion in `lock` and `unlock` APIs when multiple devices are using the same I²C channel.
- To configure multiple I²C devices on the same bus, add and configure the following modules for each device connecting to the bus:

- I²C Framework device module
- Configure the I²C shared bus module for the first device being configured, then use the same bus for the remaining devices
- I²C HAL module
- DTC module (optional).
- Lock functionality will be effective for devices from different threads. If multiple devices connected to the bus are from the same thread, the I²C bus will be locked for all devices from that thread. In such cases, even if the bus is locked, all devices from same thread can access the bus.

Note: Each I²C Framework device must be configured with a unique name in the ISDE configurator.

Note: Provide the same configuration settings for all the devices connected on the same bus (except the slave address and addressing modes).

3.1.2 I²C Framework Module Limitations

The I²C framework module does not currently support the following features:

- The use of any timer other than the GPT
- The use of DMA
- Refer to the most recent SSP Release Notes for any additional operational limitations for this module.

4. Including the I²C Framework Module in an Application

This section describes how to include the I²C Framework module in an application using the SSP configurator.

Note: This section assumes that you are familiar with creating a project, adding threads, adding a stack to a thread and configuring a block within the stack. If you are unfamiliar with any of these items, refer to the first few chapters of the *SSP User's Manual* to learn how to manage each of these important steps in creating SSP-based applications.

To add the I²C framework module to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the I²C framework module is `g_sf_i2c_device0`. This name can be changed in the associated Properties window.)

Table 3 I²C Framework Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
<code>g_sf_i2c_device0</code> I ² C Framework on <code>sf_i2c</code>	Threads	New Stack> Framework> Connectivity> I ² C Framework on <code>sf_i2c</code>

When the I²C Framework module on `sf_i2c` is added to the thread stack as shown in the figure below, the configurator automatically adds any needed lower-level modules. Any drivers that need additional configuration information will have 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.

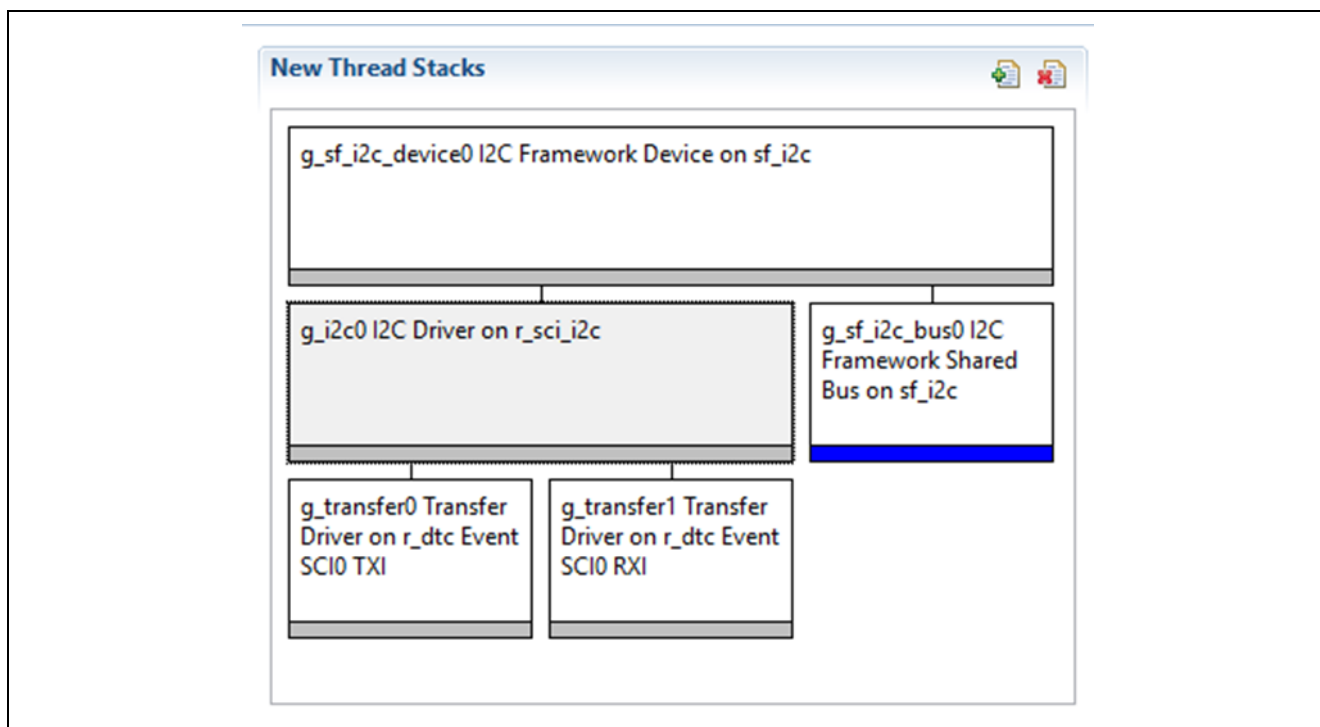


Figure 2 I²C Framework Module Stack

5. Configuring the I²C Framework Module

The I²C framework 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 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.

One of the properties most often identified as requiring a change is the interrupt priority. This configuration setting is available within the Properties window of the associated module. Simply select the indicated module and then view the Properties window. The interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Also, note that the interrupt priorities listed in the Properties window in the ISDE include an indication as to the validity of the setting based on the 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 4: Configuration Settings for the I²C Framework Module on sf_i2c

Parameter	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: Enabled)	Enable or disable parameter error checking.
Name	g_sf_i2c_device0	Give a name to identify the I ² C Framework device. API, Config and Control instances will be created based on this name.

Note: The example values and defaults are for a project using the Synergy S7G2 MCU. 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 byte ordering or pixel-color format. The configurable properties for the lower-level modules are given in the following sections for completeness and as a reference.

Note: Most of the property settings for lower-level modules are intuitive and can usually be determined by inspection of the associated Properties window with the SSP configurator.

5.1 Configuration Settings for the I²C Framework Lower-Level Drivers

Typically, only a small number of settings must be modified from the default for lower-level drivers as indicated by 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 Settings for the I²C Master HAL Module on r_sci_i2c

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable parameter error checking.
Name	g_i2c0	Module name.
Channel	0 to 9	Specify the SCI channel to be used with this configuration. SCI has the following channels: <ul style="list-style-type: none"> Series S7: 0 1 2 3 4 5 6 7 8 9 Series S3: 0 1 2 3 4 - - - - 9 Series S1: 0 1 - - - - - - 9.
Rate	Standard, Fast-mode (Default: Standard)	Standard and Fast.
Slave Address	0x00	Address of the slave device.
Address Mode	7-Bit, 10-Bit (Default: 7-Bit)	Only 7-bit addresses are currently supported.
SDA Output Delay (nanoseconds)	300	SDA output delay in nanoseconds.
Bit Rate Modulation Enable	Enable, Disable (Default: Enable)	Enables bitrate modulation function.
Callback	NULL	A user callback function can be registered in <code>i2c_api_master_t::open</code> . If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in <code>i2c_event_t</code> . 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:	Receive interrupt priority selection.

ISDE Property	Value	Description
	valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Priority 2)	
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) (Default: Priority 2)	Transmit end interrupt priority selection.

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

Table 6 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 to 9	Specify the SCI channel to be used with this configuration. SCI has channels as follows: <ul style="list-style-type: none"> Series S7: 0 1 2 3 4 5 6 7 8 9 Series S3: 0 1 2 3 4 - - - 9 Series S1: 0 1 - - - - - 9.
Rate	Standard, Fast-mode Default: Standard	Standard and Fast
Slave Address	0x00	Address of the slave device
Address Mode	7-Bit, 10-Bit (Default: 7-Bit)	Only 7-bit addresses are currently supported
SDA Output Delay (nanoseconds)	300	SDA output delay in nanoseconds
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Enables bitrate modulation function

ISDE Property	Value	Description
Callback	NULL	<p>A user callback function can be registered in <code>i2c_api_master_t::open</code>. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in <code>i2c_event_t</code>.</p> <p>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.</p>
Receive Interrupt Priority	<p>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)</p> <p>(Default: Priority 2)</p>	Receive interrupt priority selection
Transmit Interrupt Priority	<p>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)</p> <p>(Default: Priority 2)</p>	Transmit interrupt priority selection
Transmit End Interrupt Priority	<p>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)</p> <p>(Default: Priority 2)</p>	Transmit end interrupt priority selection

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

Table 7 Configuration Settings for the I²C Framework Shared Bus on `sf_i2c`

ISDE Property	Value	Description
Name	<code>g_sf_i2c_bus0</code>	Give a name to identify the bus. This bus name is used in the Framework configuration to link the I ² C peripheral to a bus.
I ² C Implementation	<p>SCI I²C, RIIC</p> <p>(Default: RIIC)</p>	Choose any low-level interface to use in the Framework.

ISDE Property	Value	Description
Channel	0-9	SCI or RIIC Channel number to which the device has been connected. In framework, the channel number given in this bus configuration will overrides the channel number given in the HAL driver.

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

Table 8 Configuration Settings 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: Enabled)	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	
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	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
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. Other MCUs may have different default values and available configuration settings.

Table 9 Configuration Settings 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)	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	
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

ISDE Property	Value	Description
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. Other MCUs may have different default values and available configuration settings.

Table 10 Configuration Settings for the Transfer Driver 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: Enabled)	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 IIC0 TXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
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

Table 11 Configuration Settings for the Transfer Driver 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

ISDE Property	Value	Description
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. Other MCUs may have different default values and available configuration settings.

5.2 I²C Framework Module Clock Configuration

The SCI peripheral module uses PCLKB as its clock source. The PCLKB frequency is set using the SSP configurator clock tab prior to a build or by using the CGC Interface at run-time. During configuration, the I²C transfer rate is calculated and set internally by the driver, based on the user-selected PCLKB rate and the user-selected transfer rate. If the PCLKB is configured in such a manner that the user-selected rate cannot be achieved, an error is returned when initializing the driver.

5.3 I²C Framework Module Pin Configuration

The SCI 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 what MCU pins are required.

Table 12 Pin Selection Sequence for SCI1

Resource	ISDE Tab	Pin Selection Sequence
SCI	Pins	Select Peripherals > SCI1_3_5_7_9 > SCI1

Note: The selection sequence assumes SCI1 is the desired hardware target for the driver.

Table 13 Pin Configuration Settings for the I²C Framework Module on SCI

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I ² C, Simple SPI, SmartCard (Default: Disabled)	Select Simple I ² C as the Operation Mode for I ² C on SCI
RXD1_SCL1_MISO1	None, P212, P708 (Default: None)	SCL Pin
TXD1_SDA1_MOSI1	None, P213, P709 (Default: None)	SDA Pin

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

5.4 I²C Framework Module Other Settings

In addition to the SCL and SDA pins, an I²C RESET signal may be required to reset the I²C slave device. If this is the case, the RESET signal can be added using a GPIO pin and must be controlled directly by the application program. The external device reset function is not supported within the `r_sci_i2c` module.

6. Using the I²C Framework Module in an Application

Once the module has been configured, the files generated, and I²C pins are configured, the I²C Framework module is ready to be used in an application. The typical steps in using the I²C module in an application are as follows:

1. Initialize the I²C module using the `open` API.
2. Reset the module using `reset` API (if needed).
3. Lock the bus using the `lock` API (if needed).
4. Configure the channel using the `write` API.
5. Unlock the bus using the `unlock` API (if needed).
6. If a read is desired, lock the bus with the `lock` API (if needed).
7. The program starts a read from the buffer using the `read` API.
8. Unlock the bus using the `unlock` API (if needed).
9. The module can be closed if desired using the `close` API.

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

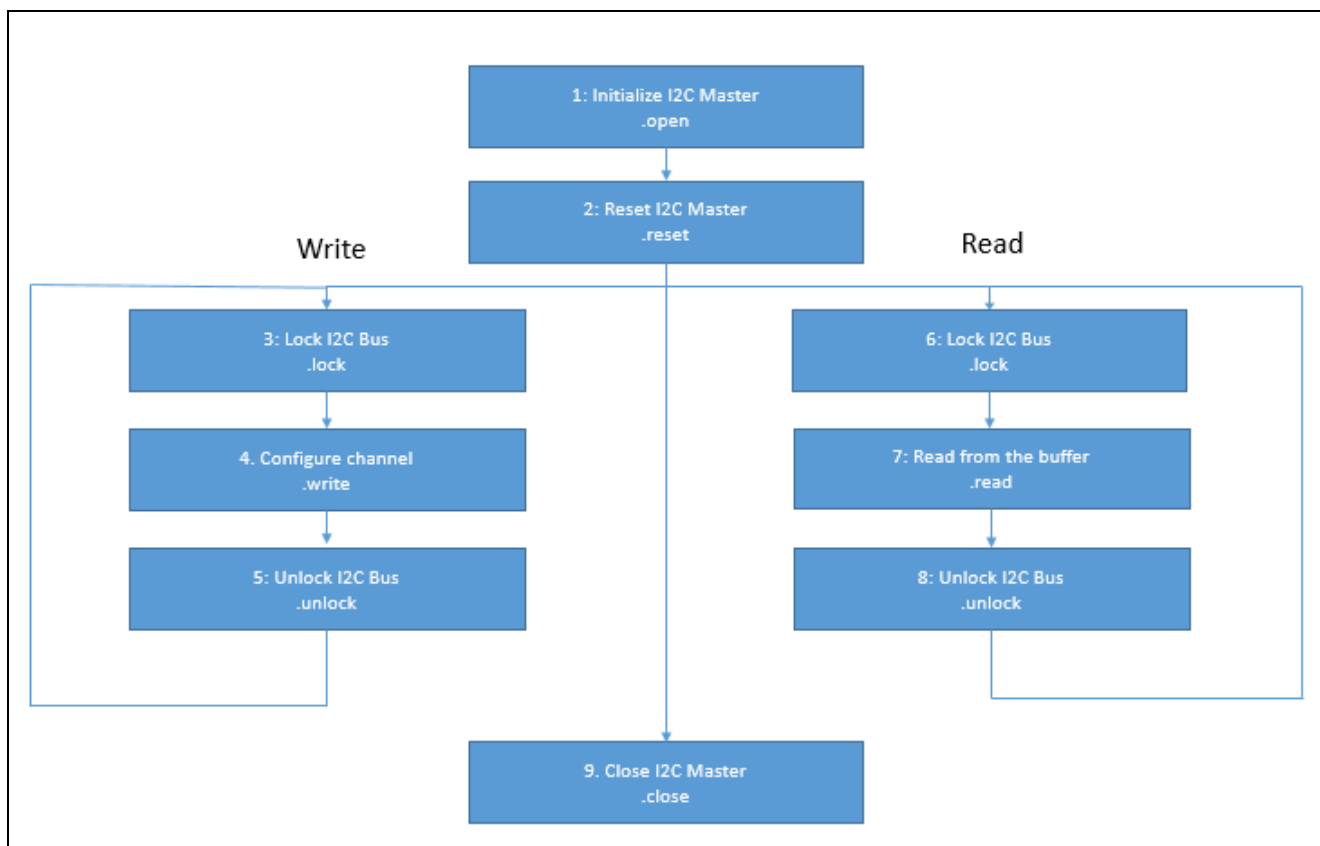


Figure 3 Flow Diagram of a Typical I²C Framework Module Application

7. The I²C Framework Module Application Project

The application project associated with this module guide demonstrates the aforementioned 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 Framework module. You can also read over the code (in `i2c_framework_entry.c`) which is used to illustrate the I²C Framework module APIs in a complete design.

The application project demonstrates the typical use of the I²C Framework module APIs. The application project main-thread entry initializes the ADC Periodic Framework module and periodically scans the temperature sensor. The scan result is placed in a user-specified buffer. A user-callback function is entered when the scan result is available. The user-specified callback function prints the result on the Debug Console using the common semi-hosting function. The following table identifies the target versions for the associated software and hardware used by the application project.

Table 14 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 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

Note: The application project assumes that you are familiar with using `printf()`, semi-hosting, and the Debug Console in your ISDE. If you are unfamiliar with these techniques, refer to the “*How do I Use Printf() with the Debug Console in the Synergy Software Package*” Knowledge Base article given in the References section at the end of this document.

The configuration settings in the application project need to be customized for the specifics of the target kit and MCU. The I²C Framework module in the application project uses the I²C HAL module and the I²C shared bus. The I²C HAL module is based on the `r_riic` module and uses channel 2 for I²C communication. The output pins for I²C communication are selected to conform with the signal connections from the touch controller. These are 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 as a check on the validity of the selected pins for the I²C signals. The location for the schematic is provided in the References section at the end of this document. The I²C shared bus is configured on channel 2. Finally, the external slave-reset signal is connected to GPIO pin P609 (also viewable on the SK-S7G2 schematic) and must be enabled and configured for proper operation.

Table 15 Application Project Configuration Settings (Changed from Defaults)

Pin Configuration Property	Setting	Description
g_sf_i2c_bus Channel	2	SCI or RIIC Channel number to which the device has been connected. In the framework, the channel number given in this bus configuration will override the channel number given in the HAL module.
g_sf_i2c_bus I ² C Implementation	RIIC	Choose any low-level interface to use in the framework
ICU (RXI, TXI, TEI, ERI)	Priority 3	Set interrupt priority
Channel	2	Set channel to conform to I ² C Slave signal connections to the MCU
SCL2	P512	SCL signal pin connection
SDA2	P511	SDA signal pin connection
GPIO Pin	P609	I ² C Slave reset signal pin connection

Pin Configuration Property	Setting	Description
GPIO Mode	Output	GPIO output mode to match reset signal requirements
Drive Capacity	Low	GPIO output drive to match reset signal requirements
Output Type	CMOS	GPIO output type to match reset signal requirements

Once the I²C Framework application project has been successfully added and configured, it can be used by the application program. The I²C Framework application project implements steps in a similar manner to those shown for the general case as described in the preceding chapter. 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.

A simple flow diagram of the application project is given in the figure below:

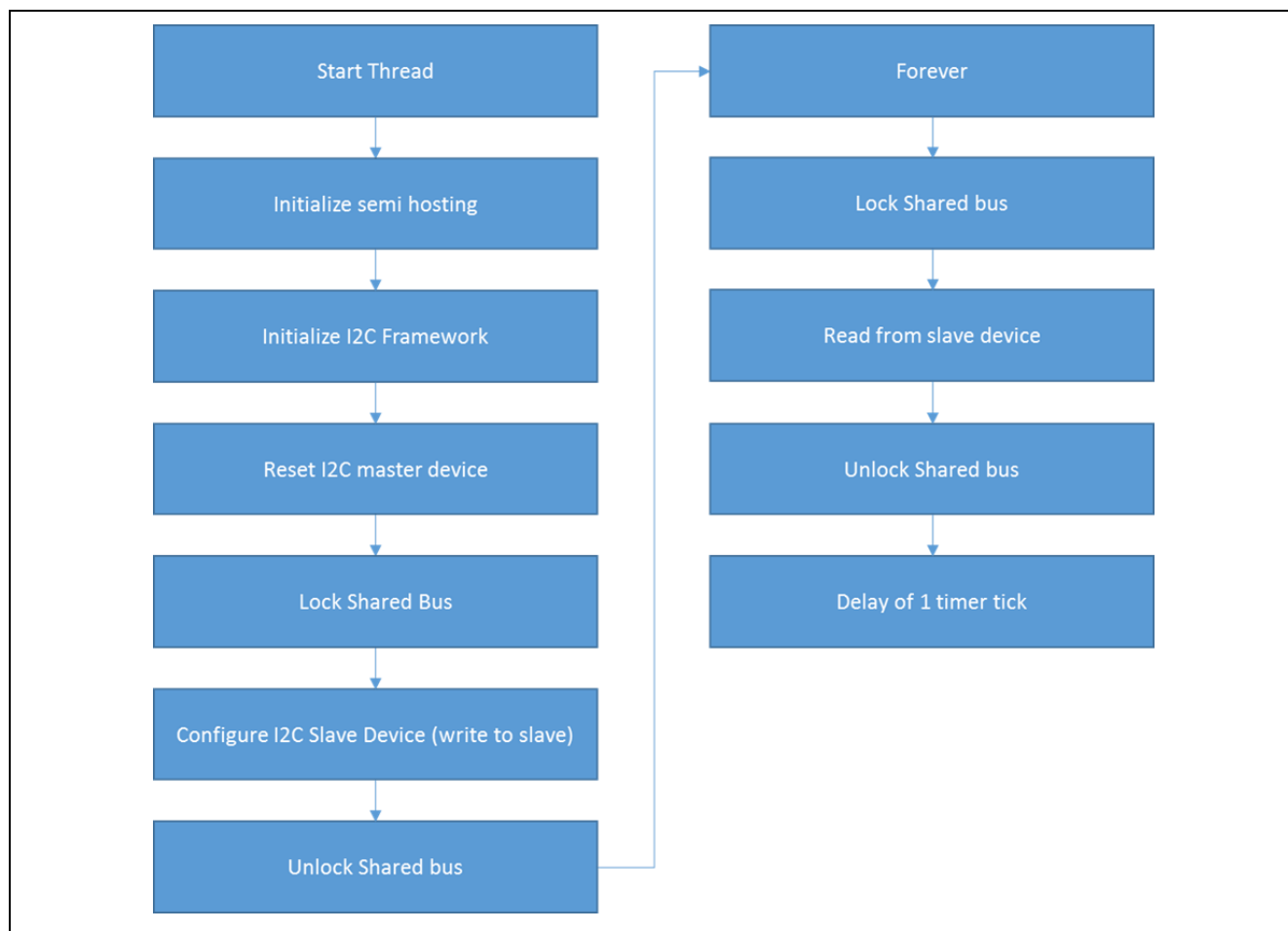


Figure 4 I²C Framework Module Application Project Flow Diagram

Referring to the code in `i2c_framework_entry.c`, you can follow along with the flow outlined in the preceding figure. The first section of the `i2c_framework_entry.c` has the header files which are referred to in the I²C instance structure to allow semi-hosting support to display results using `printf()`. The following section has the function definitions for I²C operation, for example, initializing, restarting, configuring, and reading from the slave device. The last section of the code defines the framework entry-point. The I²C functions declared in the previous sections are called here along with the read operation in the forever loop.

A few key properties of the application project are configured to support the required operations and the physical properties of the target board.

Note: It is assumed that you are familiar with using `printf()` and 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 using the watch variables in the debug mode.

A few key properties are configured in this application project to support the required operations and the physical properties of the target board and MCU. The properties with the values set for this specific project are listed in the following table. You can also open the application project and view these settings in the Properties window as a hands-on exercise.

8. Customizing the I²C Framework Module for a Target Application

The following resources are optional in case your application requires data transfer or triggering an event on an I²C interrupt. DMA/DTC transfers are not supported for the IIC module.

Table 16 Data Transfer or Triggering an Event on an I²C Interrupt

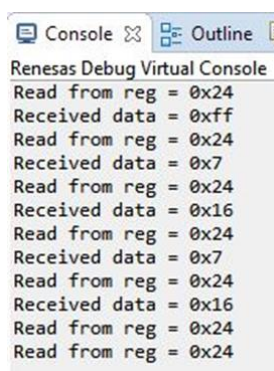
Resource	ISDE Tab	Selection
Data Transfer Control Driver	Threads	Highlight New Thread or HAL/Common and select New > Driver > Transfer > Transfer Driver on r_dtc
DMA Controller	Threads	Highlight New Thread or HAL/Common and select New > Driver > Transfer > Transfer Driver on r_dmac
Event Link Controller	Threads	The ISDE includes one instance of the ELC by default.

9. Running the I²C Framework Module Application Project

To run the I²C Framework application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug.

To implement the I²C Framework application in a new project, follow the steps below for defining, configuring, auto-generating files, adding code, compiling and debugging on the target kit.

1. Import attached I²C_Framework example project to e² studio or IAR EW for Synergy. For steps to import an example project, please refer to the *Importing a Renesas Synergy Project (r11an0023eu0116-synergy-ssp-import-guide.pdf)* document attached.
2. Compile the application without errors and warnings.
3. Connect to the host PC using a micro USB cable to J19 on SK-S7G2.
4. Start to debug the application.
5. LED1-3 will blink periodically when communication is ongoing. Also, if the `semi_hosting` is uncommented in `i2c_framework_slave_functions_mg.h`, the output can be viewed in the Renesas Debug Virtual Console as seen in the following figure. Upon touching the touch screen, the values of **Received data** in the console will change.



```

Console  Outline
Renesas Debug Virtual Console
Read from reg = 0x24
Received data = 0xff
Read from reg = 0x24
Received data = 0x7
Read from reg = 0x24
Received data = 0x16
Read from reg = 0x24
Received data = 0x7
Read from reg = 0x24
Received data = 0x16
Read from reg = 0x24
Read from reg = 0x24

```

Figure 5 Example Output from I²C Framework Application Project

- Note:
1. Output value may differ based on touch point on touch screen. When the screen is not being touched, output value is 0xff.
 2. To configure “Renesas Debug Virtual Console” in e² studio, refer to the document link given in the References section that provides a step by step process to add `printf()` with the debug console in the SSP.

10. I²C Framework 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 the incorrect selection of lower-level modules. 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, avoiding the time required in older development environments to use, or in some cases, create, lower-level drivers.

11. I²C Framework Module Next Steps

After you have mastered a simple I²C Framework application project, you may want to review a more complex example. The GUIX (r12an0021) for the SK-S7G2 application example demonstrates the use of the I²C Framework to implement the touch controller. This project is available through the link shown in the References section at the end of this document.

12. I²C Framework Module Reference Information

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

Links to all the most up-to-date `sf_i2c` module reference materials and resources are available on the Synergy Knowledge Base: [https://en-](https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/sf_i2c_Module_Guide_Resources)

[us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/sf_i2c_Module_Guide_Resources](https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/sf_i2c_Module_Guide_Resources).

Website and Support

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

Technical Contact Details:

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

All trademarks and registered trademarks are the property of their respective owners.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 15, 2017		Initial Release
1.01	Aug 7, 2017		Update to Hardware and Software Resources table

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
 2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
 4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
 6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
 7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
 10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
 11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

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

Renesas Electronics America Inc.

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

Renesas Electronics Canada Limited

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

Renesas Electronics Europe Limited

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

Renesas Electronics Europe GmbH

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

Renesas Electronics (China) Co., Ltd.

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

Renesas Electronics (Shanghai) Co., Ltd.

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

Renesas Electronics Hong Kong Limited

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

Renesas Electronics Taiwan Co., Ltd.

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

Renesas Electronics Singapore Pte. Ltd.

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

Renesas Electronics Malaysia Sdn.Bhd.

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

Renesas Electronics India Pvt. Ltd.

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

Renesas Electronics Korea Co., Ltd.

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