Renesas Synergy™ Platform

SPI on SCI 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 included in this document and should be valuable resources for creating more complex designs.

The SCI SPI HAL module is a high-level API for master-based SPI serial communications and is implemented on the r_sci_spi (simple SPI) module. The SCI SPI HAL module configures and controls the SPI functionality of a Synergy MCU using the SCI (Serial Communications Interface) peripheral. The module is configured in the ISDE. A user-defined callback can be created to signal when the SPI has transmitted data, aborted a data transfer, or detected an error condition.

The SCI SPI HAL modules are enabled with a data transfer support by incorporating the DTC HAL module of the MCU. This performs SCI SPI transfer through DTC without intervention of the CPU.

Contents

1. SCI SPI HAL Module Features 2
2. SCI SPI Module APIs Overview 2
3. SCI SPI HAL Module Operational Overview 3
3.1 SCI SPI HAL Module Important Operational Notes and Limitations 4
3.2 SCI SPI HAL Module Operational Notes 4
3.3 SCI SPI HAL Module Limitations 4
4. Including the SCI SPI HAL Module in an Application 4
5. Configuring the SCI SPI HAL Module 5
5.1 SCI SPI HAL Module Clock Configuration 6
5.2 SCI SPI HAL Module Pin Configuration 6
6. Using the SCI SPI HAL Module in an Application 7
7. The SCI SPI HAL Module Application Project 8
8. Customizing the SCI SPI HAL Module for a Target Application 10
9. Running the SCI SPI HAL Module Application Project 10
10. SCI SPI HAL Module Conclusion 11
11. SCI SPI HAL Module Next Steps 11
12. SCI SPI HAL Module Reference Information 11
Revision History 13

1. SCI SPI HAL Module Features

The SCI SPI HAL module supports the configuration and control of the various SPI functionality on the Synergy MCU. Key features include the following:

- Driver initialization
- Serial communication through SPI operation using 8-bit data transfers
- Configurable among four clock phase and clock polarity settings
- Support for callbacks. The callback functions are called with the following events:
  - Transfer aborted
  - Transfer complete
  - Over run error
- SPI communication in master mode.

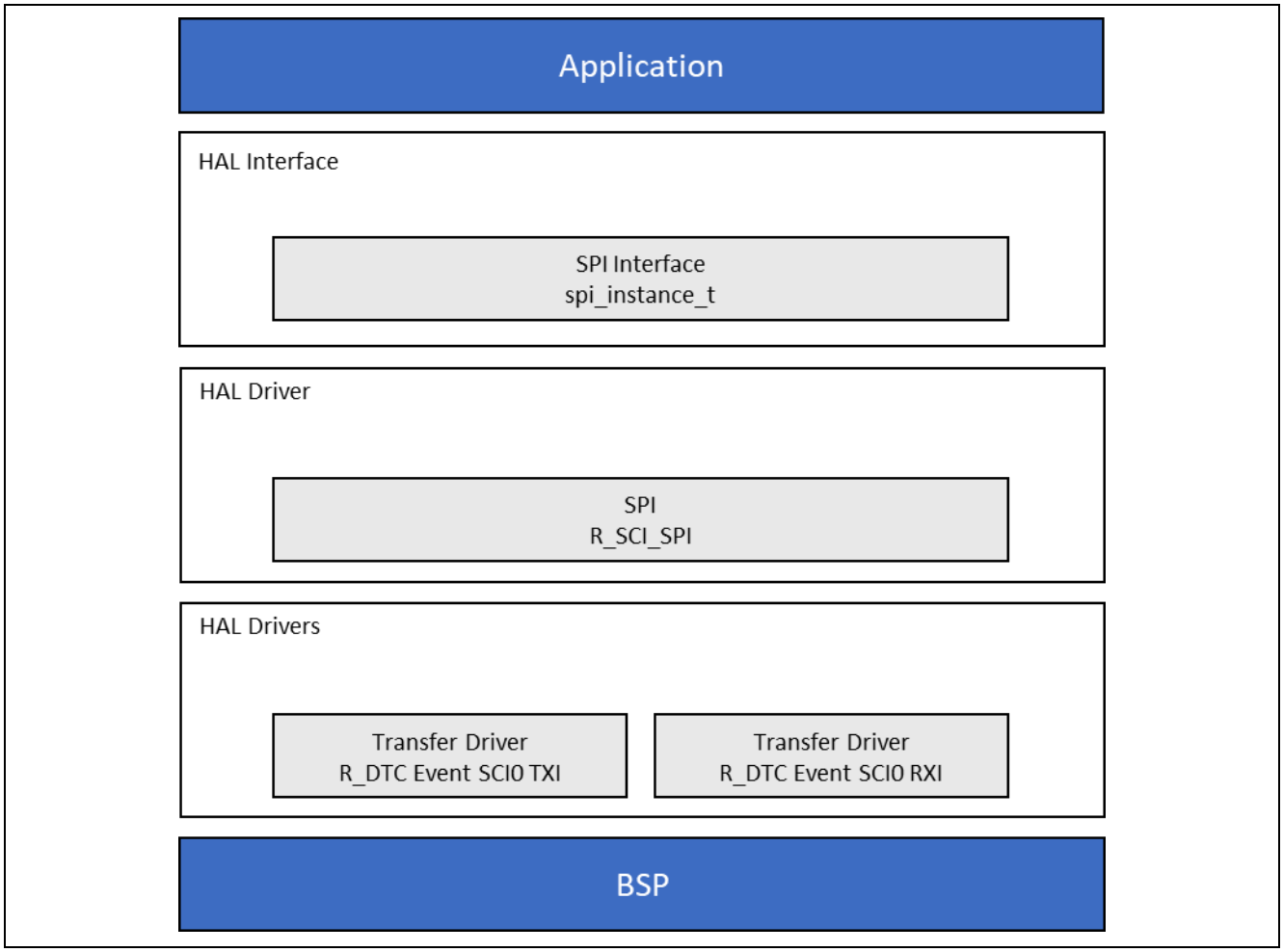

Figure 1. SCI SPI HAL Module Block Diagram

2. SCI SPI Module APIs Overview

The SPI defines APIs for opening and closing the SCI peripheral and transmitting and receiving data. 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. SCI SPI HAL Module API Summary

Function Name	Example API Call and Description
.open	`g_spi.p_api->open(g_spi.p_ctrl, g_spi.p_cfg);` Opens the SPI driver and initializes the hardware.

Function Name	Example API Call and Description
.close	`g_spi.p_api->close(g_spi.p_ctrl);` Closes the driver and releases the SPI device.
.read	`g_spi.p_api->read(g_spi.p_ctrl, &read_buffer, number_of_bytes_to_read, bit_width);` Performs a read operation on an SPI device.
.write	`g_spi.p_api->write(g_spi.p_ctrl, &write_buffer, number_of_bytes_to_write, bit_width);` Performs a write operation on an SPI device.
.writeRead	`g_spi.p_api->writeRead(g_spi.p_ctrl, &read_buffer, &write_buffer, number_of_bytes_to_read_and_write, bit_width);` Performs a read and write (full duplex) operations on an SPI device.
.versionGet	`g_spi.p_api->versionGet(&version);` Retrieve the API version with the version pointer.

Note: For details on 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_IN_USE	Attempted to open an already open device instance OR Another transfer was in progress.
SSP_ERR_INVALID_POINTER	p_version is NULL.
SSP_INVALID_ARGUMENT	Channel number invalid.
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_CH_NOT_OPEN	The channel has not been opened. Open channel first.

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. SCI SPI HAL Module Operational Overview

The SCI SPI interface is able to configure and use the SPI functionality of the Synergy MCU. The HAL module enables communication with a peripheral device using the SPI communications protocol. After opening the SCI SPI HAL module instance, the SCI SPI module uses its handle to perform various transfer operations. The device control handle is used within API calls to indicate a specific SCI SPI device with which to communicate.

The driver allows the user to:

- Initialize the driver.
- Serial Communication through SPI operation.
  - Read from, write to, and simultaneous read/write (full duplex) a SPI device — by calling the read, write, and writeRead APIs.

The driver also provides support for callbacks. The callback functions are called with the following events:

- Transfer aborted
- Transfer complete
- Overrun error

The SCI SPI Modules support only 8-bit data transfer operations. SCI SPI Modules uses GPIO pins configured as chip selects.

Clock settings:

The SCI SPI uses PCLKA as its clock source. You can set the PCLKA frequency using the clock configurator in e^2 studio or the @ref CGC_API at run-time.

IO Port settings:

To use with the SPI, the I/O port pin(s) used as output pins must be configured as SCI SPI peripheral pins in the pin configurator. For external chip select, configure Chip select pin as GPIO output.

SCI SPI Interrupts:

To enable interrupts of SCI SPI, highlight the driver module and set the priority of the SCI RXI, TXI, TEI and ERI interrupts on the Threads tab of the Project Configurator in e^2 studio: @ref configuring-interrupts.

This sets the corresponding interrupts in ssp_cfg/bsp/bsp_irq_cfg.h to the priority level selected.

Warning: Setting the interrupts to different priority levels could result in improper operation.

3.1 SCI SPI HAL Module Important Operational Notes and Limitations

3.2 SCI SPI HAL Module Operational Notes

- Chip select outputs are supported using GPIOs
- The SPI on SCI module uses only 8-bit data transfers.
- Setting the interrupts to different priority levels could result in improper operation.
- The SCI SPI HAL module is enabled with a data transfer support by incorporating the Data Transfer Controller module of the MCU. This performs SPI transfers through the DTC without intervention of the CPU. The DTC transfer is enabled by default, the user has to remove it from the configurator for an IRQ mode transfer.

3.3 SCI SPI HAL Module Limitations

- When implementing SPI on SCI, only the master mode is supported.
- Refer to the most recent SSP Release Notes for any additional operational limitations for this module.

4. Including the SCI SPI HAL Module in an Application

This section describes how to include the SPI 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.

To add the SPI Driver to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for a SPI is g_spi0. This name can be changed in the associated Properties window.)

Table 3. SPI Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_spi0 SPI Driver on g_sci_spi	Threads	New Stack> Driver> Connectivity> SPI Driver on g_sci_spi

When the SPI HAL module is added to the thread stack as shown in the following figure, the configurator automatically adds the needed lower-level drivers. Any drivers that need additional configuration information will be highlighted in Red. Modules with a Gray band are individual modules that stand alone.

The following figure shows SPI on SCI drivers added to the HAL/Common Thread. The resource name is `g_spi0`. The SPI HAL module can support a transfer driver, whereas the data-transfer controller can be used to transmit and receive data between the hardware peripherals and data buffers in RAM (rather than the CPU performing the transfers). By default, a transfer driver is added for both transmit and receive.

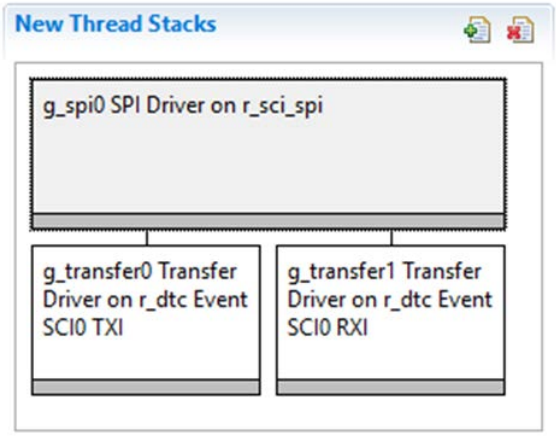

Figure 2. SCI SPI HAL Module Stack

5. Configuring the SCI SPI HAL Module

The SPI HAL module must be configured by the user for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in Red) any required configuration selections, such as interrupts or operating modes, which must be configured for lower-level modules for successful operation. Only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and unavailable for changes in the Properties window in the ISDE. These are identified with a lock icon for the 'locked' property. This approach simplifies the configuration process and makes it much less error-prone than previous 'manual' approaches to configuration. The available configuration settings and defaults for all the user-accessible properties are given in the Properties tab within the SSP Configurator, and are shown in the following tables for easy reference.

Note: You may want to open your ISDE, create the 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 SCI SPI HAL Module on r_sci_spi

ISDE Property	Value	Description
Parameter Checking	Default (BSP), Enabled, Disabled	Enable or disable the parameter error checking.
Name	g_spi0	Instance name of driver. Can be renamed by user.
Channel	0	SCI channel number to be used for SPI communication.
Operating Mode	Master	How the SPI driver operates – master or slave (not supported).
Clock Phase	Data sampling on odd edge, data variation on even edge Data sampling on even edge, data variation on odd edge (Default: Data sampling on odd edge, data variation on even edge)	How the clock signal is working in terms of clock signal edges and data operations.
Clock Polarity	Low when idle, High when idle (Default: Low when idle)	High output when no operations are performed. Alternative: low output when no operations are performed.
Mode Fault Error	Disable, Enable (Default: Disable)	How errors are handled.
Bit Order	MSB First, LSB First (Default: MSB First)	Bit order within transmitted/received bytes.
Bitrate	100000	Transmission speed in bits per second.

ISDE Property	Value	Description
Bit Rate Modulation Enable	Enable, Disable (Default: Disable)	The bit rate can be evenly corrected using the MDDR register. This will help to reduce %error on baud rates in data communications. Beneficial for systems with none standard crystal frequencies and for lower baud rate communications.
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 spi_event_t.
Receive Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	The SPI driver is interrupt driven. MCU interrupt priority level.
Transmit Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	The SPI driver is interrupt driven. MCU interrupt priority level.
Transmit End Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	The SPI driver is interrupt driven. MCU interrupt priority level.
Error Interrupt Priority	Priority 0 > 15 (Default: Priority 2)	The SPI driver is interrupt driven. MCU interrupt priority level.

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

In many cases, settings other than the defaults for the SPI Driver will be required to communicate via the SPI. For example, changes to the bit rate, clock polarity, and bit order may differ between slave devices.

5.1 SCI SPI HAL Module Clock Configuration

The SCI peripheral is clocked via the Peripheral Clock A (PCLKA.) The clock frequencies are configurable in the ISDE by using the Clocks tab in the configurator. Invalid selections are indicated in red when selected. Ensure that desired SPI bitrate can be achieved with the stated value of PCLKA. The ISDE will not be indicated if the specified bitrate is not achievable. At run time, the SPI HAL module will attempt to configure the SCI peripherals to the correct bitrate and will return an error if the desired bitrate cannot be set. The bitrate is calculated via the equations in the following table. If the result of the equation (N) is in the range of 0 to 255, then the bit rate can be achieved.

Table 5. Baud Rate Calculation Equations

SPI HAL	Bitrate calculation	Description
SPI on SCI	$N = \frac{PCLKA\ (MHz)}{8 * 2^{(2n-1)} * \left(256/M\right) * B} - 1$	N = Peripheral register value. This must be in the range of 0 to 255 PLCKA = value of PLCKA in MHz n = 0, 1, 2 or 3(SMR_SMCI.CKS[1:0] bit which decides the divisor value for the PCLKA) M = Bit Rate Modulation Index 128 < M < 256 *If the Bit Rate Modulation is disabled, then M=256* B = Desired Bit Rate

5.2 SCI SPI HAL Module Pin Configuration

The SCI peripheral use 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 SPI pins.

Table 6. Pin Selection Sequence for SCI SPI HAL Driver

Resource	ISDE Tab	Pin selection Sequence
SPI on SCI	Pins	Select Peripherals > Connectivity:SCI > SCIx Where x is the required SCI peripheral channel.

Table 7. Pin Configuration Settings for SPI on SCI –Example Case of using SCI0

Pin Configuration Property	Value	Description
Pin Group Selection	Mixed, _A only, _B only	Synergy devices support peripheral functionality via multiple pins location, identified by _A, _B. Selecting Mixed allows the user to select any combination of locations (_A and _B). Selecting _A allows the user to select only _A locations. Selecting _B allows the user to select only _B locations.
Operation Mode	Disabled Custom Asynchronous UART Simple SPI Simple I2C Synchronous UART Smart Card	Set the operating mode to: Simple SPI.
TXD_MOSI	None, P411, P101	Specify the port pin to be used as MOSI.
RXD_MISO	None, P410, P100	Specify the port pin to be used as MISO.
SCK	None, P412, P102	Specify the port pin to be used as CLK.
CTS_RTS_SS	None, P413, P103	Specify the port pin to be used as SS.

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

6. Using the SCI SPI HAL Module in an Application

A sample procedure for using the SCI SPI in an application is as follows.

The `g_spi.p_api->open()` function must be called first. The rest of the calls may be used in any order depending on the application requirements:

1. Open a SPI instance with the SPI implemented by SCI SPI. The SCI SPI driver is called through the SPI Interface `g_spi.p_api->open (g_spi.p_ctrl, g_spi.p_cfg)` where `p_ctrl` and `p_cfg` are the instances of control and configuration structures autogenerated after the configuration step.
2. Initiate a write to a slave device by calling `g_spi.p_api->write (g_spi.p_ctrl, source, length, SPI_BIT_WIDTH_8_BITS);` where `g_spi.p_ctrl` is the same control instance that was used in the open call.
3. Initiate a read from a slave device by calling `g_spi.p_api->read(g_spi.p_ctrl, dst, length, SPI_BIT_WIDTH_8_BITS);` where `g_spi.p_ctrl` is the same control instance that was used in the open call.
4. Initiate a simultaneous transfer from and to a slave device by calling `g_spi.p_api->writeRead(g_spi.p_ctrl, source, dst, length, SPI_BIT_WIDTH_8_BITS);` where `g_spi.p_ctrl` is the same control instance that was used in the open call.
5. To close the SPI channel, do so by calling `g_spi.p_api->close(g_spi,p_ctrl)` where `g_spi.p_ctrl` is the same control structure that was used in the open call.

The following flow diagram illustrates the previously described common steps:

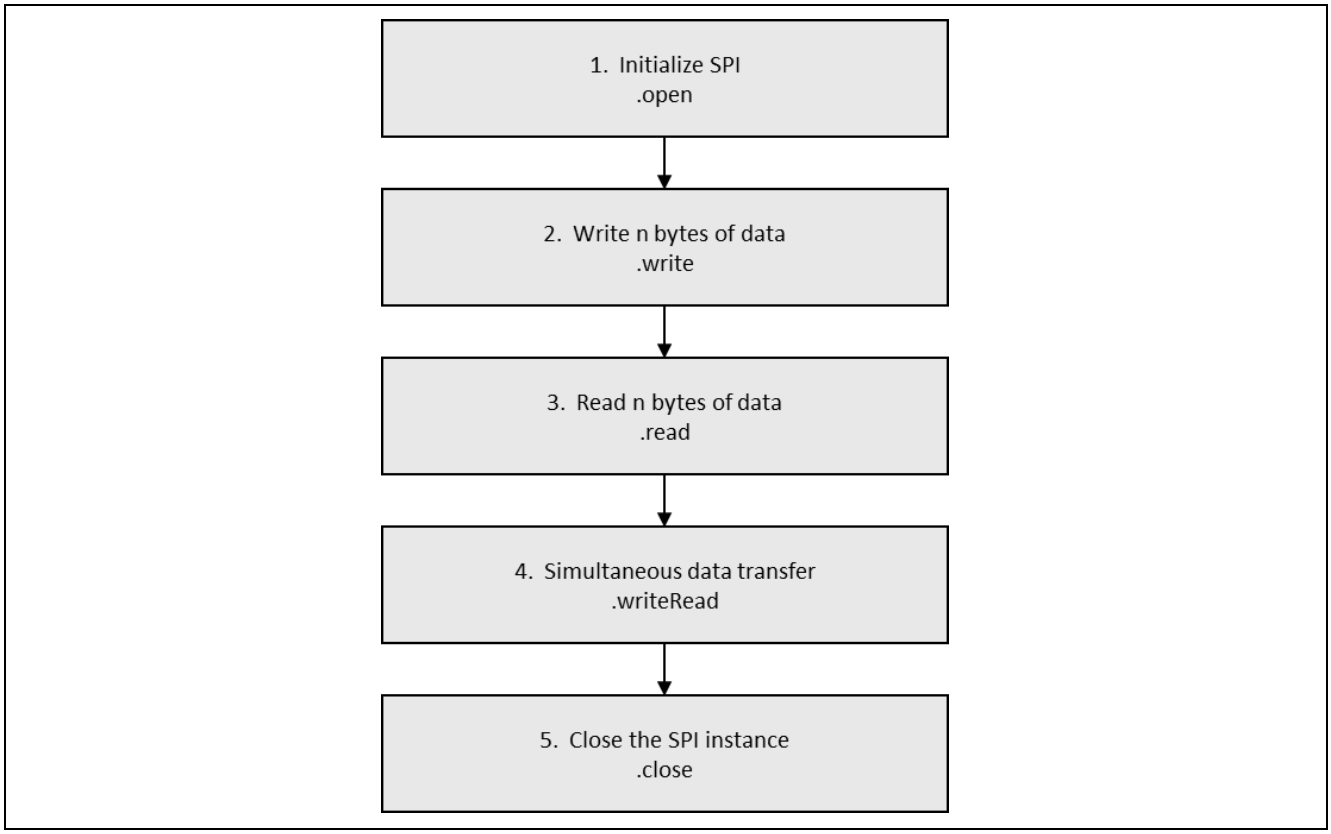

Figure 3. Flow Diagram of a Typical SPI Application

7. The SCI SPI HAL Module Application Project

The application project demonstrates the typical use of the SPI APIs. The application project demonstrates SPI communication between the Synergy S7 (SPI master) and a MAX31723 temperature sensor (SPI slave) – MAX31723 – plugged into the board's PMOD-B interface.

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

Resource	Revision	Description
e^2 studio	6.2.1 or later	Integrated Solution Development Environment
SSP	1.5.0 or later	Synergy Software Platform
IAR EW for Renesas Synergy	8.23.1 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	6.2.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following diagram shows the application project in a simple flow:

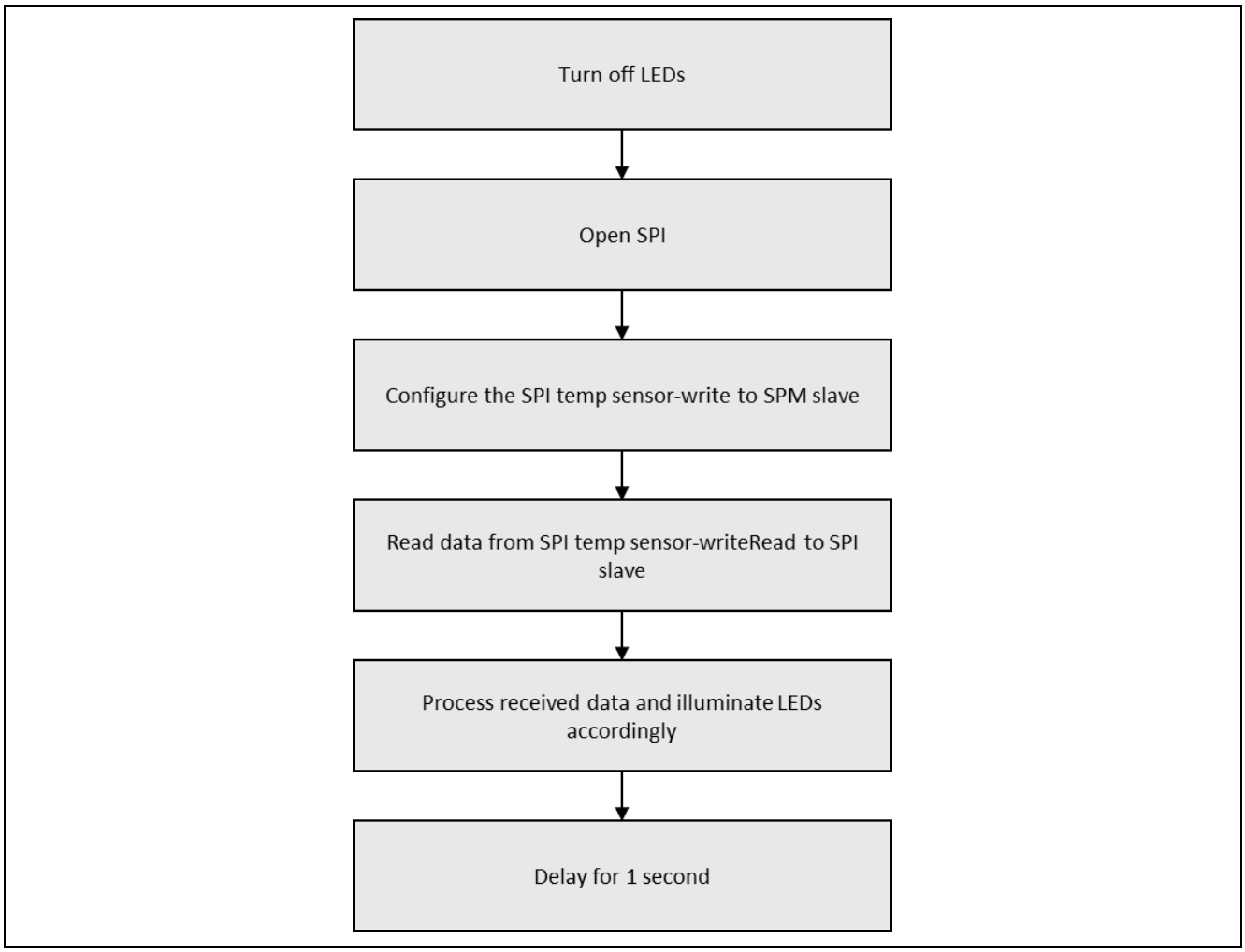

Figure 4. SCI SPI HAL Module Application Project Flow Diagram

The `spi_hal.c` file is in the project once it has been imported into the ISDE. You can open this file within the ISDE, and read along with the following description to help identify key uses of APIs.

The first section of `spi_hal.c` has the header files which reference the SPI instance structure and the math functions that are used to perform floating-point calculation of the temperature. The next section has a #define for the I/O pin that is used for the SSL line; this is followed by the global variable definitions used within the application and the function prototypes.

The entry function for the main program control section is `spi_hal_module_guide_project()`. Within the function, local variables for temperature calculation are defined, as well as data arrays containing configuration data for configuring the temperature sensor and a storage location for the received temperature data.

Because the application project illuminates the LEDs according to the calculated temperature, the LEDs are initially set to their starting state of all off.

The next stage is to configure the temperature sensor. The SSL line is set high and then the configuration data is written to the temperature sensor using the `write` API. (The data configures the temperature sensor for 12-bit resolution.) Once the write function has been successfully completed, the SSL line is then set low, terminating the configuration process. A successful completion of the write function results in the SPI callback function being called. The callback function simply sets a software flag indicating that the application can proceed.

The application now enters a `while(1)` loop. The temperature is read using the `writeRead` API; the SSL line is then set (high before and low after) the API call. The data written is the address from where the temperature data can be read from. The temperature data is 12-bits in size, so a minimum of 2 bytes of data must be read. The storage location for temperature data is 3 bytes in size, as the process of writing the address will receive dummy data.

The temperature is then calculated using the valid 2 bytes received.

The first temperature that is calculated by the application is stored as the reference temperature. All subsequent temperature calculations are compared to this reference temperature. If the new temperature differs from the reference temperature, then the LEDs are illuminated accordingly. 2^{O}C delta – Green LED, 3^{O}C delta – Green and Orange LEDs, 4^{O}C delta Green and Orange and Red LEDs.

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

Table 9. SPI on SCI Configuration Settings for the Application Project

ISDE Property	Value Set
g_spi0 SPI Driver on r_sci_spi	
Name	g_spi0
Unit	0
Operating Mode	Master
Clock Phase	Data sampling on even edge, data variation on odd edge
Clock Polarity	High when idle
Mode Fault Error	Disable
Bit Order	MSB First
Bitrate	100000
Bit Rate Modulation Enable	Enable
Callback	spi_callback
Receive Interrupt Priority	Priority 8
Transmit Interrupt Priority	Priority 8
Transmit End Interrupt Priority	Priority 8
Error Interrupt Priority	Priority 8
DTC Driver for Transmission	Removed
DTC Driver for Reception	Removed
Pin Selection	
SCI0: TXD_MOSI	P411
SCI0: RXD_MISO	P410
SCI0: SCK	P412
IOPORT P413	Output Mode (Initial Low)

8. Customizing the SCI SPI HAL Module for a Target Application

Within a user-target application, the configuration settings will be changed by the developer from those shown in the application project. For example, you can easily change the configuration settings for the SPI channel to use a different bitrate and phase relationship between the clock and data. You can also change the port pins to match the chosen SPI channel.

9. Running the SCI SPI HAL Module Application Project

To run the SPI Application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug. Refer to the Synergy Project Import Guide (r11an0023eu01121-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 SPI application in a new project, use the following steps for defining, configuring, auto-generating files, adding code, compiling, and debugging on the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical, while just reading over this guide will tend to be more theoretical.

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters of the SSP User's Manual for a description of how to accomplish these steps.

To create and run the SPI application project, simply follow these steps:

1. Create a new Renesas Synergy project for the SK-S7G2 board (S7G2-BSP) called "**SCI_SPI_HAL_MG_AP**".
2. Select the S7G2-SK board, select the BSP option and create the project.
3. Open the configuration.xml from the generated project and select the **Threads** tab.
4. Add the SPI driver to use: SPI driver on r_sci_spi in HAL/Common from New Stack > Driver > Connectivity.
5. Set SPI Driver parameters (Name, Channel, Clock Phase, Clock Polarity, Callback etc.) from the stack's properties.
6. Enable the SCI peripheral pins for the selected channel from the Pins tab.
7. Click on the **Generate Project Content** button.
8. Add the code from the supplied project file "`spi_hal.c/h`" or copy the file over the generated "`spi_hal.c/h`" file.
9. Connect to the host PC via a micro USB cable to J19 on SK-S7G2.
10. Connect the temperature sensor to PMODB.
11. Start to debug the application.
12. Touch the temperature sensor (change the temperature) and watch the LEDs illuminate.

10. SCI SPI HAL Module Conclusion

This module guide has provided all the background information needed to select, add, configure, and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy™ Platform makes these steps much less time consuming and removes the common errors, like conflicting configuration settings or 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. SCI SPI HAL Module Next Steps

After you have mastered a simple SPI HAL module project, you may want to review a more complex example. Other application projects and application notes that demonstrate SPI HAL module functionality can be found as described in the References section at the end of this document.

12. SCI SPI HAL Module Reference Information

SSP User Manual: Available in HTML format at www.renesas.com/us/en/products/synergy/software/ssp.html as a SSP distribution package, and also as a pdf from the Synergy Gallery.

Links to all the most up-to-date r_sci_spi module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977512.

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	May.12.17	—	Version 1.0
1.01	Aug.24.17	—	Update to Hardware and Software Resources Table
1.02	Feb.25.19	—	Updated to SSP v1.5.0

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/.