Renesas Synergy™ Platform

RSPI 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 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 a valuable resource for creating more complex designs.

The SPI HAL module is a generic API for communication using the SPI protocol. The module supports both the SPI and SCI peripherals available on the Synergy microcontrollers and is implemented on `r_rspi` and `r_sci_spi`. This guide refers to the `r_rspi` HAL module, also called the SPI module (formerly known as RSPI). The SPI HAL module supports standard SPI master and slave mode communications functions. Callbacks are provided for transfer events.

The SPI HAL modules are enabled with data-transfer support by incorporating the Data Transfer Controller (DTC) module of the MCU. This performs SPI transfers through the DTC without intervention from the CPU.

Target Device

Synergy SK-S7G2 Kit and the S7G2 MCU Group.

Contents

1. RSPI HAL Module Features....3
2. RSPI HAL Module APIs Overview....4
3. RSPI HAL Module Operational Overview....5
3.1 RSPI HAL Module important operational notes and limitations....5
3.1.1 RSPI HAL Module operational notes....5
3.1.2 RSPI HAL Module limitations....6
4. Including the RSPI HAL Module in an Application....6
5. Configuring the RSPI HAL Module....7
5.1 RSPI HAL Module Clock Configuration....11
5.2 RSPI HAL Module Pin Configuration....11
6. Using the RSPI Module in an Application....11
7. The RSPI HAL Module Application Project....12
8. Customizing the RSPI HAL Module for a Target Application....15
9. Running the RSPI HAL Module Application Project....15
10. RSPI HAL Module Conclusion....16
11. RSPI HAL Module Next Steps....16

12. RSPI HAL Module Reference Information....16

Revision History....18

1. RSPI HAL Module Features

The SPI HAL module supports the following key features:

- Initialization of the driver
- SPI transfer functions
  - Serial communication through SPI operation using the four-wire method
  - Serial communication in master and slave modes
  - Switching the polarity of the serial transfer clock
  - Switching the phase of the serial transfer clock
- Data format
  - MSB-first/LSB-first selectable
  - Transfer bit length is selectable as 8, 16, and 32 bits
- Error detection
  - Mode fault detection
  - Overrun error detection
  - Parity error detection
- SSL control functions
  - Internally select up to four SSL signals (SSLn0 to SSLn3) for each channel in master mode
  - External hardware slave select can be used in master mode
- Interrupts
  - RSPI receive interrupt (receive buffer full)
  - RSPI transmit interrupt (transmit buffer empty)
  - RSPI error interrupt (mode fault, overrun and parity error)
- Delays
  - Add SPI clock delay
  - Add slave select negation delay
  - Add next-access delay

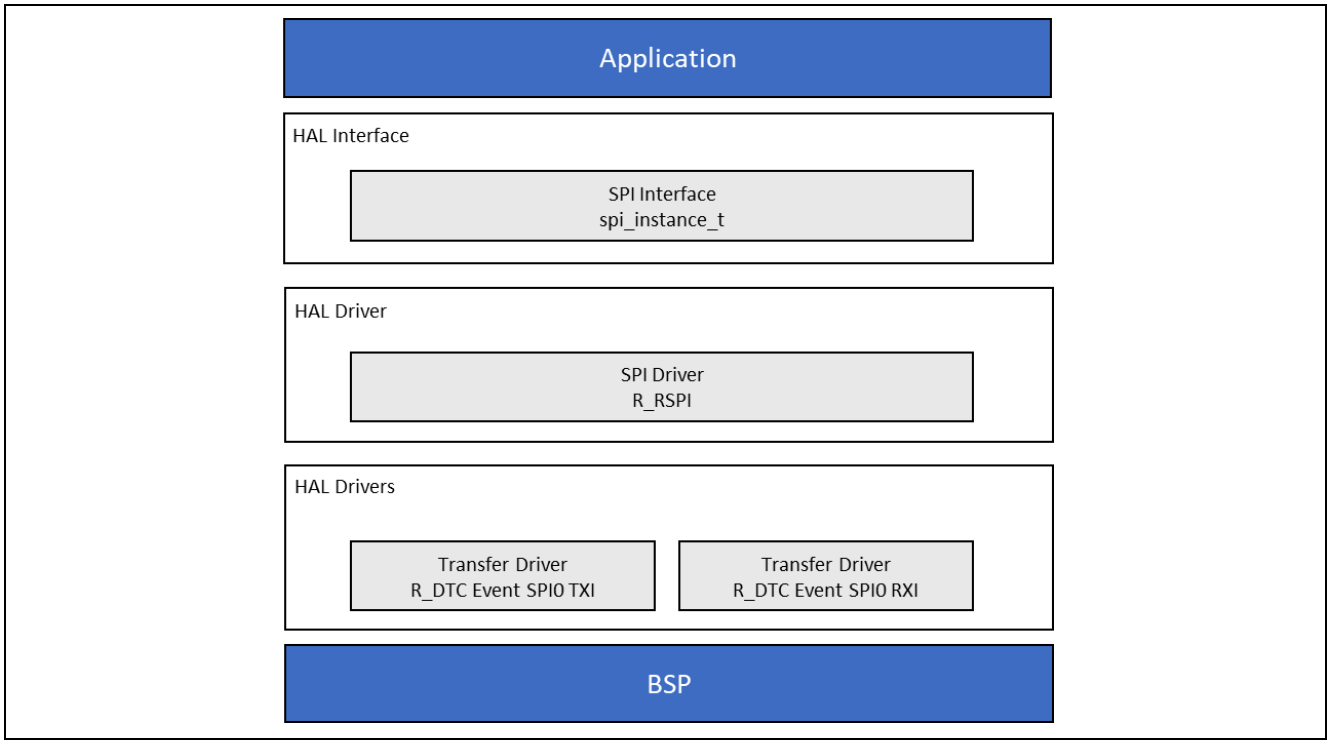

Figure 1. RSPI HAL Module Block Diagram

2. RSPI HAL Module APIs Overview

The RSPI HAL module defines APIs for opening, closing, reading, writing, and other useful functions. 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. RSPI 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);` Open a designated SPI device.
.read	`g_spi.p_api->read(g_spi.p_ctrl, dst16, length, SPI_BIT_WIDTH_16_BITS);` Receive data from SPI device.
.write	`g_spi.p_api->write (g_spi.p_ctrl, source, length, SPI_BIT_WIDTH_8_BITS);` Transmit data to SPI device
.writeRead	`g_spi.p_api ->writeRead (g_spi.p_ctrl, &source, &destination, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER);` Simultaneously transmit data to an SPI device, while receiving data from an SPI device (full duplex). The writeRead API fetches the mutex object, handles SPI data transmission at SPI HAL layer, and receives data from the SPI HAL layer. The API uses the event flag wait to synchronize to complete the data transfer.
.close	`g_spi.p_api->close(g_spi,p_ctrl)` Disable the SPI device designated by the control handle and close the RTOS services used by the bus if no devices are connected to the bus. This function removes power to the SPI channel designated by the handle and disables the associated interrupts.
.versionGet	`g_spi.p_api ->versionGet (&version);` Get the version information of the underlying driver.

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	Function completed successfully
SSP_ERR_INVALID_MODE	Invalid mode
SSP_ERR_INVALID_CHANNEL	Invalid channel
SSP_ERR_IN_USE	In-use error
SSP_ERR_INVALID_ARGUMENT	Invalid argument
SSP_ERR_QUEUE_UNAVAILABLE	Queue unavailable
SSP_ERR_INVALID_POINTER	Invalid pointer
SSP_ERR_INTERNAL	Internal error
SSP_ERR_TRANSFER_ABORTED	Transfer aborted
SSP_ERR_MODE_FAULT	Mode fault
SSP_ERR_READ_OVF	Read overflow
SSP_ERR_PARITY	Parity error
SSP_ERR_OVERRUN	Overrun error
SSP_ERR_UNDEF	Unknown error
SSP_ERR_TIMEOUT	Timeout error
SSP_ERR_CH_NOT_OPEN	Device not opened
SSP_ERR_HW_LOCKED	Locked during attempt to writeRead

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. RSPI HAL Module Operational Overview

The SPI HAL module enables communication with a peripheral device using the SPI communications protocol. After opening the SPI HAL module instance, the SPI module handle is used to perform various transfer operations. The device control handle will be used within the API calls to indicate the specific SPI device to communicate with.

The Driver allows the user to:

- Initialize the driver.
- Serial Communication through SPI operation.

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

- Transfer aborted
- Transfer complete
- Mode fault
- Error events

SPI module supports 8, 16, and 32-bit data transfers. SPI Module supports GPIO pins configured as chip selects. In addition, the SPI peripheral supports dedicated chip select signals SSLn0 to SSLn3. In the SPI peripheral, all chip select handling is performed by the MCU.

Clock Settings

The SPI uses PCLKA as its clock source. You can set the PCLKA frequency using the clock configurator in Renesas Synergy™ e² studio or the CGC_API at run-time.

Note: For Synergy S1 MCU Series, the SPI clock source is PCLKB.

IO Port Settings

To use with the SPI, you must configure the I/O port pin(s) used as output pins as SPI peripheral pins in the pin configurator. If you are using an external chip select, configure Chip select pin as GPIO output.

SPI Interrupts

Use e² studio ISDE to configure the SPI interrupts using the **Threads** tab: See the Configuring Interrupts section in the *SSP User's Manual*.

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

SPI Interrupts

Precondition: To enable interrupts of SPI, on the **ICU** tab of the Project Configurator, highlight the driver module and set the priority of the RSPIn SPRI, SPTI, SPII and SPEI interrupts (where n is the SPI channel number): See the Configuring Interrupts section in the *SSP User's Manual*.

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.

Extended Configuration

Many extended MCU specific configurations are present for SPI Driver.

Note: All parameters are set in the SPI extended driver configuration structure `spi_on_rspi_cfg_t.`

3.1 RSPI HAL Module important operational notes and limitations

3.1.1 RSPI HAL Module operational notes

Warning: Setting the interrupts to different priority levels while configuring SPI HAL drivers could result in improper operation.

Data transfer support for the SPI HAL driver module is enabled by incorporating the Data Transfer Controller module of the MCU. This performs SPI transfer through DTC without intervention of the CPU.

In the application, data transfer over DTC is used in the same way as we use for normal SPI transfer. To enable DTC transfers, add the DTC module under the SPI HAL module.

SPI module supports only 32-bit data transfer in DTC transfer mode, and it supports all 8, 16, and 32-bit data transfers in the SPI transfer mode.

SPI Data Transfer using the DTC

Data transfer support for the SPI HAL driver module is enabled by incorporating the Data Transfer Controller module of the MCU. This performs SPI transfer through the DTC without intervention of the CPU.

The SPI HAL driver module takes care of the configuration of the DTC driver. No user configuration is needed for this. Upon completion of data transfer, a callback function will be notified with the SPI_EVENT_TRANSFER_COMPLETE event.

A non-DTC transfer (IRQ mode) can be done by just removing the Rx and Tx DTC modules under the SPI module configuration. Use the **X** mark near the **Threads** menu for this.

3.1.2 RSPI HAL Module limitations

Refer to the most recent *SSP Release Notes* and r_rspi Module Guide Resource for any additional operational limitations for this module.

4. Including the RSPI HAL Module in an Application

This section describes how to include the RSPI 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 this link *SSP User's Manual* to learn how to manage each of these important steps in creating SSP-based applications.

To add the RSPI HAL 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 SPI Framework is `g_spi0`. This name can be changed in the associated **Properties** window.)

Table 3. RSPI HAL driver selection sequence

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

The following figure shows when the RSPI HAL module on `r_rspi` is added to the thread stack, the configurator automatically adds the needed lower-level drivers. Modules with a Gray band are individual modules that stand alone.

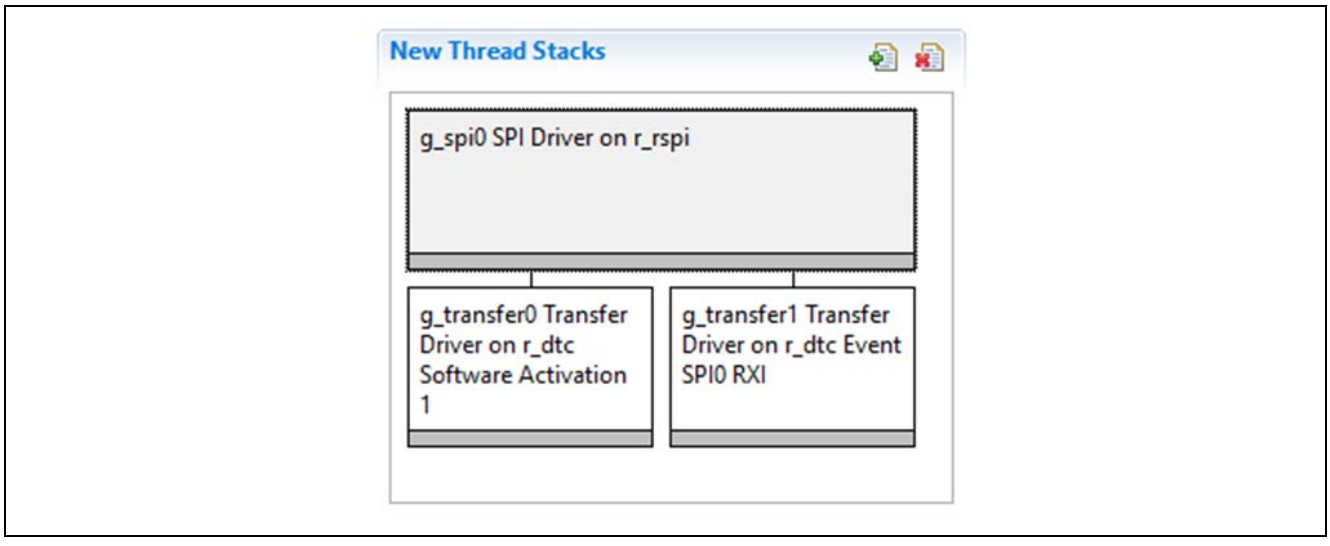

Figure 2. RSPI HAL Module Stack

5. Configuring the RSPI HAL Module

Configure the RSPI HAL module on `r_rspi` for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in red) any configurations, such as interrupts or operating modes, for lower-level modules required for successful operation. Only properties that can be changed without causing conflicts are available for modification. Properties that cannot be modified are **locked** with a lock icon displayed for the **locked** property in the ISDE **Properties** window. 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 will include an indication as to the validity of the setting based on the targeted MCU (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 RSPI HAL Module on r_rspi

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_spi0	Module name
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master, Slave Default: Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
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	Data sampling on odd or even clock edge.
Clock Polarity	Low when idle, High when idle Default: Low when idle	Clock level when idle.
Mode Fault Error	Enable, Disable Default: Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First, LSB First Default: MSB First	Select transmit order MSB/LSB first
Bitrate	500000	Transmission or reception rate. Bits per second.
Callback	NULL	Optional callback function pointer.
SPI Mode	SPI Operation, Clock synchronous operation Default: SPI Operation	Select SPI or clock sync mode operation.
SPI Communication Mode	Full Duplex, Transmit Only Default: Full Duplex	Select full-duplex or transmit-only communication.

ISDE Property	Value	Description
Slave Select Polarity(SSL0)	Active Low, Active High Default: Active Low	Select SSL0 signal polarity
Slave Select Polarity(SSL1)	Active Low, Active High Default: Active Low	Select SSL1 signal polarity
Slave Select Polarity(SSL2)	Active Low, Active High Default: Active Low	Select SSL2 signal polarity
Slave Select Polarity(SSL3)	Active Low, Active High Default: Active Low	Select SSL3 signal polarity
Select Loopback1	Normal, Inverted Default: Normal	Select loopback1
Select Loopback2	Normal, Inverted Default: Normal	Select loopback2
Enable MOSI Idle State	Enable, Disable Default: Disable	Select MOSI idle fixed value and selection
MOSI Idle State	MOSI Low, MOSI High Default: MOSI Low	Select MOSI idle fixed value and selection
Enable Parity	Enable, Disable Default: Disable	Enable/disable parity
Parity Mode	Parity Odd, Parity Even Default: Parity Odd	Select parity
Select SSL(Slave Select)	SSL0, SSL1, SSL2, SSL3 Default: SSL0	Select which slave to use; 0-SSL0; 1-SSL1; 2-SSL2; 3-SSL3
Select SSL Level After Transfer	SSL Level Keep, SSL Level Do Not Keep Default: SSL Level Do Not Keep	Select SSL level after transfer completion; 0-negate; 1-keep
Clock Delay Enable	Clock Delay Enable, Clock Delay Disable Default: Clock Delay Disable	Clock delay enable selection
Clock Delay Count	Clock Delay 1 thru 8 RSPCK Default: Clock Delay 1 RSPCK	Clock delay count selection
SSL Negation Delay Enable	Negation Delay Enable, Negation Delay Disable Default: Negation Delay Disable	SSL negation delay enable selection
Negation Delay Count	Negation Delay 1 thru 8 RSPCK Default: Negation Delay 1 RSPCK	Negation delay count selection
Next Access Delay Enable	Next Access Delay Enable, Next Access Delay Disable Default: Next Access Delay Disable	Next access delay enable selection
Next Access Delay Count	Next Access Delay 1 thru 8 RSPCK Default: Next Access Delay 1 RSPCK	Next access delay count selection

ISDE Property	Value	Description
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
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 Synergy S7G2 MCU Group. Other MCUs may have different default values and available configuration settings.

Table 5. Configuration Settings for the DTC HAL Module on r_dtc Software Activation 1

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 selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	4 bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	
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

ISDE Property	Value	Description
Activation Source (Must enable IRQ)	Software Activation 1	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 Synergy S7G2 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 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
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	4 bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 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 Synergy S7G2 MCU Group. Other MCUs may have different default values and available configuration settings.

5.1 RSPI HAL Module Clock Configuration

The SPI peripheral is clocked via 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 bit rate can be achieved with the stated value of PCLKA. The ISDE will not indicate if the specified bit rate is not achievable. At run time, the SPI driver will attempt to configure the SPI peripheral to the correct bit rate and will return an error if the desired bit rate cannot be set. The bit rate 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 7. Baud Rate Calculation Equations

SPI HAL	Bitrate Calculation	Description
SPI on SPI	$n = \frac{PCLKA\ (MHz)}{2 * 2^N * B} - 1$	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 (BRDV[1:0] bits value set according to the SPBR register for desired bit rate) B = Desired Bit Rate

5.2 RSPI HAL Module Pin Configuration

The SPI 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 listing a selection for SPI pins.

Table 8. Pin Selection Sequence for the RSPI HAL Module

Resource	ISDE	Pin selection Sequence
RSPI	Pins	Select Peripherals > Connectivity:SPI > SPI0

Note: The top selection sequence assumes SPIO is the desired hardware target for the driver and the bottom selection sequence assumes SPI_0 is the desired target.

Table 9. Pin Configuration Settings for the RSPI HAL Module

Property	Value	Description
Operation Mode	Disabled, Custom, Enabled (Default: Disabled)	Select Enabled for SPI Operation
MISO	None, P100, P410 (Default: None)	MISO Pin selection
MOSI	None, P101, P411 (Default: None)	MOSI Pin selection
RSPCLK	None, P102, P412 (Default: None)	RSPCLK Pin selection
SSL0:3	None, P103:106, P413:415 (Default: None)	SSL0:3 Pin selections

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

6. Using the RSPI Module in an Application

To write an SPI application using the SPI, follow these steps. 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 an SPI instance with the SPI implemented by SPI. The 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 auto-generated after the SPI 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, dst16, length, SPI_BIT_WIDTH_16_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, dst16, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER);` 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 diagram illustrates the typical steps in an operational flow.

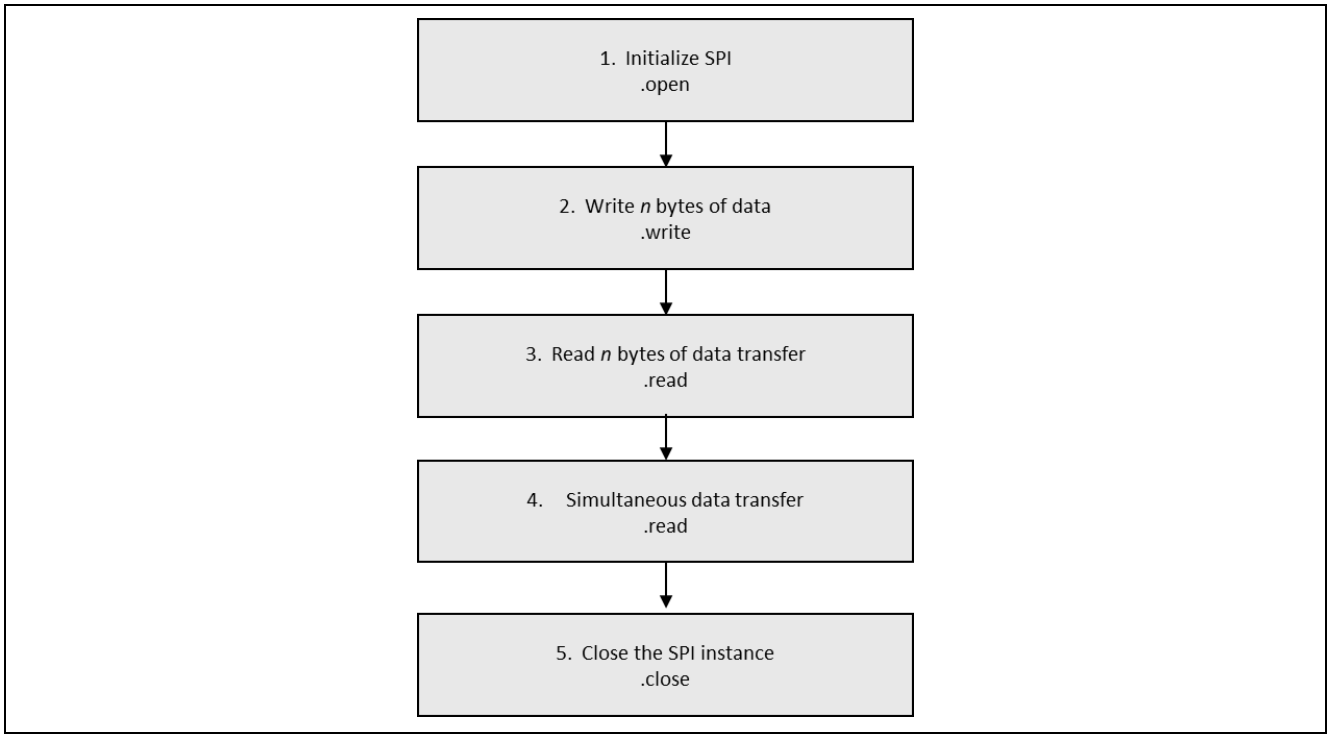

Figure 3. Flow Diagram of a Typical RSPI HAL Module Application

7. The RSPI HAL Module Application Project

The application project demonstrates the typical use of the SPI APIs, including SPI communication between the Synergy S7G2 MCU (SPI master) and a MAX31723 temperature sensor (SPI slave) plugged into the PMOD-A interface on the MCU board.

Table 10. Software and MCU Resources Used by the Application Project

Resource	Revision	Description
e² studio	v7.3.0 or later	Integrated Solution Development Environment
SSP	v1.6.0 or later	Synergy Software Platform
IAR EW for Renesas Synergy	v8.23.3 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	v7.3.0 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following figure shows a simple flow diagram of the application project.

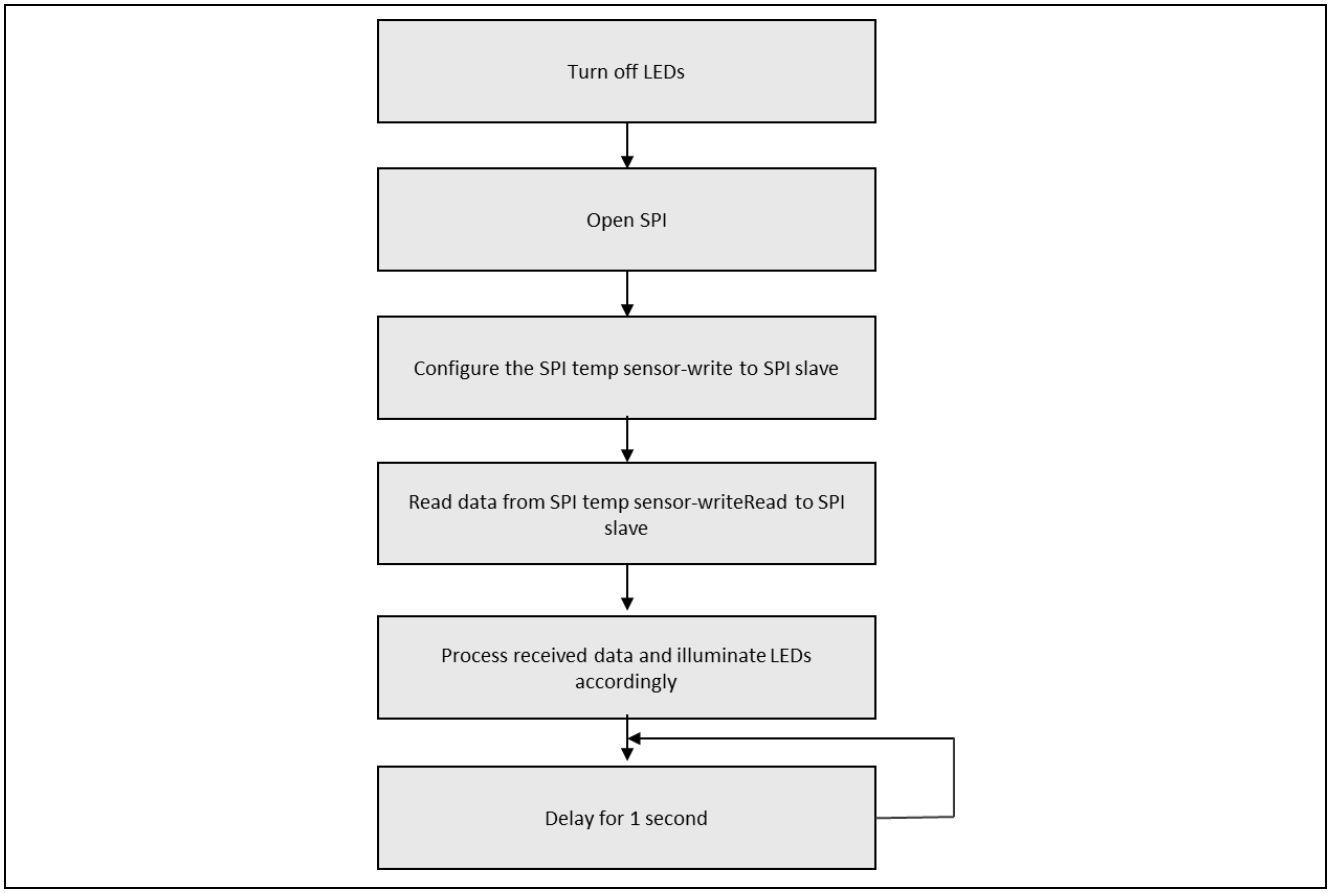

Figure 4. RSPI HAL Module Application Project Flow Diagram

The `spi_hal.c` file is available in the project once it has been imported into the ISDE. You can open this file in the ISDE while reading the following description to help you identify key uses of the APIs.

The first section of `spi_hal.c` has the header files. The header files reference the SPI instance structure and the math functions used to perform floating point calculation of the temperature. If enabled, via a #define in the file `spi_hal.h`, a code section included allows semi-hosting to display results using `printf()`. The next section has a #define for the I/O pin used for the SSL line. Following the I/O pin section are 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. The application project illuminates LEDs based on the calculated temperature. The starting state for all LEDs is OFF.

The next stage is to configure the temperature sensor. The SSL line is set high, the configuration data then 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 successfully completed, the SSL line is set low, terminating the configuration process. A successful completion of the write function results initiates the SPI callback function. The callback function sets a software flag that indicates the application can proceed.

The application now enters a `while(1)` loop. The temperature is read using the `writeRead` API. The SSL line is set high before and then low after the API call. The data written is the address where the temperature data is read. The temperature data is 12 bits in size. A minimum of 2 bytes of data should be read. The storage location for temperature data is 3 bytes in size. During the write, the address receives dummy data.

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

The first temperature calculated by the application is stored as the reference temperature. Subsequent temperature calculations occur every second and are compared to the reference temperature. If the new temperature differs from the reference temperature, the LEDs are illuminated accordingly.

2°C delta Green LED
3°C delta Green and Orange LEDs
4°C delta Green, Orange, and Red LEDs

If enabled, the Debug console shows the temperatures measured.

Note: It is assumed you are familiar with the SSP Debug console and using `printf()`. If you need assistance, refer to the **How do I use Printf() with the Debug Console** in the Synergy Software Package (SSP) given in the References section at the end of this document. Alternatively, you can see results via the watch variables in the debug mode.

Configure the project's key properties to support any required operations, target board physical properties, as well as the MCU. The following table lists properties with values set for this specific project. You can also open the application project and view these settings in the **Properties** window as a hands-on exercise.

Table 11. Configuration Settings for the RSPI HAL Module Application Framework

ISDE Property	Value Set
g_spi0 SPI Driver on r_spi	
Name	g_spi
Channel	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	500000
Callback	spi_callback
SPI Mode	SPI Operation
Slave Select Polarity (SSL0)	Active Low
Slave Select Polarity (SSL1)	Active Low
Slave Select Polarity (SSL2)	Active Low
Slave Select Polarity (SSL3)	Active Low
Select Loopback1	Normal
Select Loopback2	Normal
Enable MOSI Idle	Disable
Parity Mode	Parity Odd
Select SSL (Slave Select)	SSL0
Select SSL Level After Transfer	SSL Level Do Not Keep
Clock Delay Enable	Clock Delay Disable
Clock Delay Count	Clock Delay 1 RSPCK
SSL Negation Delay Enable	Negation Delay Disable
Negation Delay Count	Negation Delay 1 RSPCK
Next Access Delay Enable	Next Access Delay Disable
Next Access Delay Count	Next Access Delay 1 RSPCK
Receive Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
Transmit Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
Error Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
DTC Driver for Transmission	Removed
DTC Driver for Reception	Removed
Pin Selection	
SPI0 MISO	P100

ISDE Property	Value Set
SPI0 MOSI	P101
SPI0 RSPCK	P102
SSL PIN - IOPORT P103	Output Mode (Initial Low)

8. Customizing the RSPI HAL Module for a Target Application

Within a user-target application, you can change settings from those configured in the application project. For example, the you can easily change the SPI channel configuration settings for bit rate, or the phase relationship between clock and data. You can also change the port pins to match the chosen SPI channel.

9. Running the RSPI HAL Module Application Project

To run the SPI application project and to see it executed on a target kit, you can import it into your ISDE, compile and run debug. Refer to the *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf), included in this package, for instructions on importing the project into e² studio or IAR Embedded Workbench® for Renesas Synergy™, as well as building and running the application.

To implement the SPI application in a new project, use the following steps. These steps help you to define, configure, auto-generate files, add code, compile and debug the target kit. After these steps, there is a hands-on approach that can help make the development process with SSP more practical, while just reading over this guide will be more theoretical.

Note: The following steps are sufficient for someone experienced with the basic flow through the Synergy development process. If these steps are unfamiliar, refer to the first few chapters of the *SSP User's Manual* for assistance.

To create and run the RSPI HAL module application project, simply follow these steps:

Note: Refer to the table listing configuration settings for the RSPI HAL Module Application Framework.

1. Create a new Renesas Synergy project for the SK-S7G2 MCU (S7G2-BSP) called, RSPI_HAL_MG_AP SPI_HAL.
2. For the S7G2-SK MCU, select the BSP option. Create the project.
3. Open `Configuration.xml` from the generated project and select **Threads** tab.
4. Add the SPI driver to use the SPI driver on `r_spi` in HAL/Common from **New Stack > Driver > Connectivity**.
5. Set SPI Driver parameters from the **Stack Properties** (see configuration settings table).
6. Enable SCI peripheral pins for selected channel from **Pins** tab (see configuration settings table).
7. Click **Generate Project Content**.
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. Connect the micro USB cable to J19 on the SK-S7G2 MCU.
10. Connect the temperature sensor to PMODA.
11. Start to debug the application.
12. Touch the temperature sensor (change the temperature) and watch the LEDs turn on.

Note**:** After flashing the program on to the board —and while not using the debugger — execute a hard reset of the board once by shorting the reset pins at J2. This reset needs to be done due to the reset issue when not using debugger. Note, you can ignore this step while using debugger in e² studio.

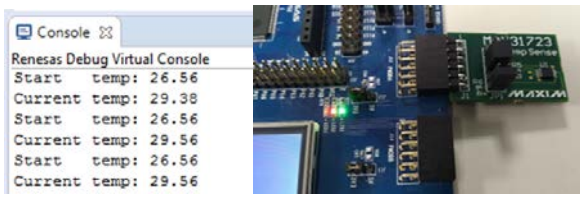

Figure 5. Example Output from RSPI HAL Module Application Project

10. RSPI 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) illustrates 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. RSPI HAL Module Next Steps

After you have mastered a simple RSPI HAL Module project, you may want to review a more complex example. Other application projects and application notes that demonstrate RSPI HAL use can be found as described in the References section below.

12. RSPI 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_rspi module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977502.

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.23.17	-	Initial version
1.01	Sep.07.17	12	Update to Hardware and Software Resources Table
1.02	Nov.09.17	-	Added the .module_descriptions folder to the source project
1.03	Feb.22.19	-	Added support for SSP v1.5.0
1.04	Apr.30.19	-	Added support for SSP v1.6.0

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.