SPI Master/Slave Communication Sample Code (Using CMSIS Driver Package) for RE01 1500KB Group, 256KB Group

SPI Sample Code Using CMSIS Driver Package

Summary

This application note describes the SPI Master / slave sample code that conforms to the RE01 CMSIS driver package. This sample code can be found in the project delivered with this application note.

The overview of this sample code is shown in the table below.

Table Overview of Sample Code

Overview of Sample Code Operation	Peripheral Module Mainly Used	Driver Module Mainly Used
Transfers ROM data to RAM using the SPI driver.	SPI	R_SPI

Target Device

RE01 1500KB Group

RE01 256KB Group

Note

When applying the sample code covered in this application note to another microcomputer, please modify the code according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Related Document

RE01 1500KB, 256KB Group Startup Guide to Development Using CMSIS Package (R01AN4660)

Contents

1. Specifications

1.1 Description of Project

The following sample code projects are provided with this application note.

Sample code project for RE01 1500KB group : r01an4698_cmsis_spi_re_1500kb

Sample code project for RE01 256KB group : r01an4698_cmsis_spi_re_256kb

The r01an4698_cmsis_spi_re_1500kb project has been tested using the Evaluation Kit RE01 1500KB. This project is configured to match the settings of R7F0E015D2CFB mounted on the Evaluation Kit RE01 1500KB.

The r01an4698_cmsis_spi_re_256kb project has been tested using the Evaluation Kit RE01 256KB. This project is configured to match the settings of R7F0E01182CFP mounted on the Evaluation Kit RE01 256KB.

When using another device, change the device settings in the project to those of the target device

1.2 Pins Used

The pins used by the sample code are shown below.

Table 1-1 The pins used by the sample code for RE01 1500KB group

Pin Used	Purpose of Use
P501	MOSIA_C
P502	RSPCKA_C
P012	SSLA0_B
P609	MOSIB_B
P607	RSPCKB_B
P610	SSLB0_B

Table 1-2 The pins used by the sample code for RE01 256KB group

Pin Used	Purpose of Use
P010	MOSIA_B
P011	RSPCKA_B
P014	SSLA0_B
P110	MOSIB_A
P108	RSPCKB_A
P106	SSLB0_A

1.3 Folder Structure

The folder structure of the sample code is shown below.

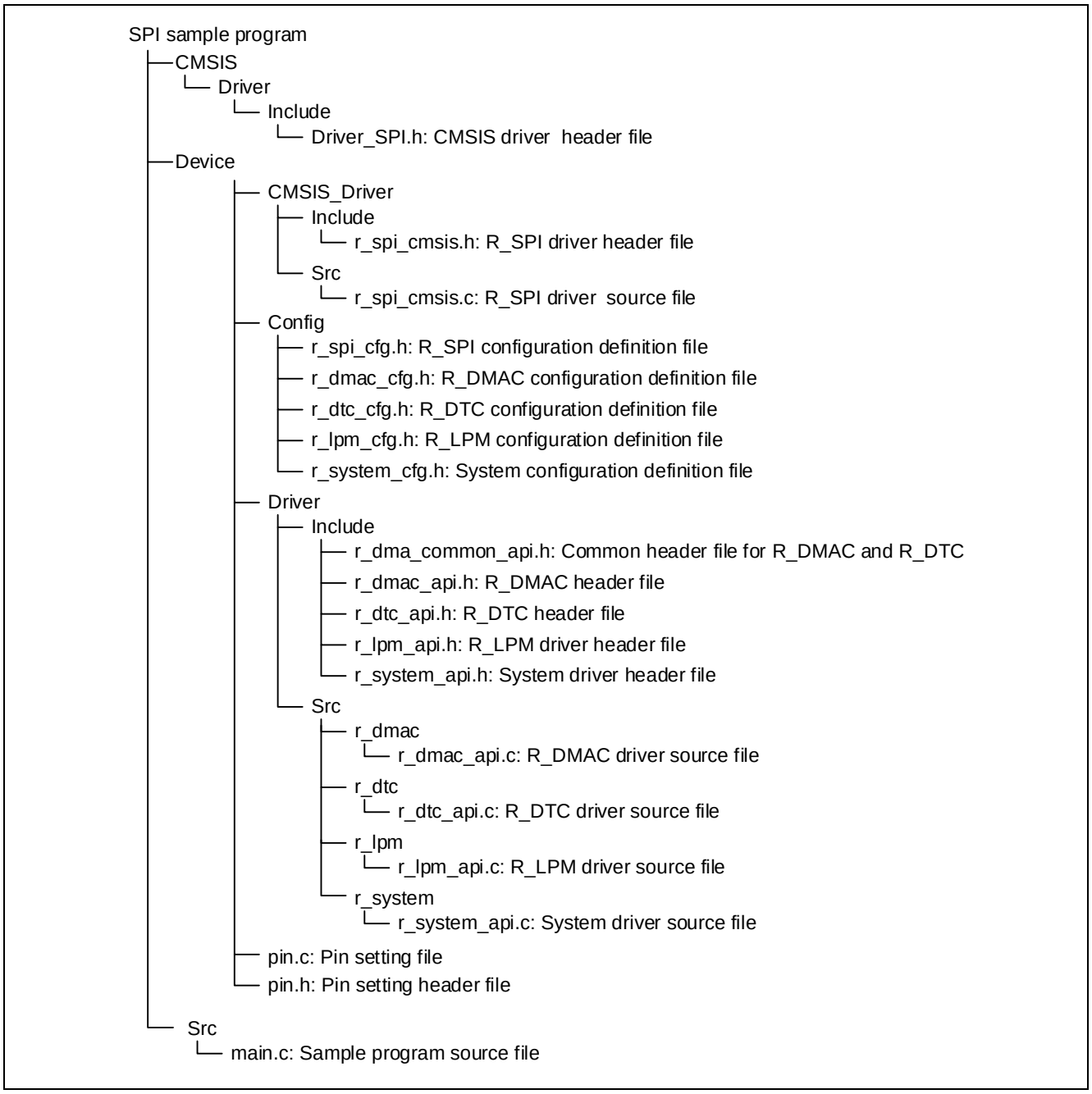


Figure 1.1 Folder Structure

1.4 File Configuration

Table 1-3 shows the files that are added or modified for this sample code.

Table 1-3 Files Added or Modified for this Sample Code

File Name	Overview of Processing or Configuration	Remarks
main.c	Main processing	
r_spi_cfg.h	SPI configuration	Specifying sources of DMAC and DTC transfer activation when SPI interrupts occur
pin.c	PIN setting	SPI1 pin setting
r_system_cfg.h	System configuration	Registering SPI transfer interrupt to NVIC

1.5 Option-Setting Memory

Table 1-4 shows the option-setting memory setting for the sample code. Set suitable values for a user system if required.

Table 1-4 Option-Setting Memory Setting for Sample Code

Symbol	Address	Setting	Description
AWS	0100A164h to 0100A167h	FFFF FFFFh	No access window settings
OSIS	0100A150h to 0100A15Fh	FFFF FFFFh	No ID code protection (All FFh)
SECMPUxxx	00000408h to 0000043Bh	FFFF FFFFh	MPU is disabled.
OFS1	00000404h to 00000407h	FFFF FFFFh	After a reset, the voltage monitor 0 reset is disabled. After a reset, HOCO oscillation is disabled.
OFS0	00000400h to 00000403h	FFFF FFFFh	Automatic activation of IWDT is disabled. Automatic activation of WDT is disabled.

2. Operating Conditions

The operation of the sample code provided with this application note has been tested under the following conditions (Table 2-1, Table 2-2).

Table 2-1 Operating Conditions for RE01 1500KB group

Item		Description
Microcontroller used		R7F0E015D2CFB 144pin
Operating frequency	PLL is selected as the system clock	• Main clock: 32 MHz • PLL: 64 MHz (main clock frequency is divided by 4 and then multiplied by 8) • System clock (ICLK): 64 MHz (PLL) • Peripheral module clock A (PCLKA): 64 MHz (PLL frequency is not divided) • Peripheral module clock B(PCLKB): 32 MHz (PLL frequency is divided by 2)
Operating voltage		• 3.3V
Target board		Evaluation Kit RE01 1500KB (RTK70E015DSXXXXXBE)
Integrated Development Environment	GCC	Renesas e² studio Version 2021-01
	IAR	IAR Embedded Workbench for ARM Version 8.50.6
C compiler	GCC	GCC ARM Embedded Version 6.3.1.20170620 GNU 6-2017-q2-update
	IAR	IAR C/C++ Compiler for ARM Version 8.50.6
Debugger		Segger J-Link OB
I/O header Version		Rev1.10
Sample code Version		Rev1.05

Table 2-2 Operating Conditions for RE01 256KB group

Item		Description
Microcontroller used		R7F0E01182CFP 100pin
Operating frequency	PLL is selected as the system clock	• HOCO: 64MHz • System clock (ICLK): 64 MHz (HOCO) • Peripheral module clock A (PCLKA): 64 MHz (HOCO is not divided) • Peripheral module clock B(PCLKB): 32 MHz (HOCO is divided by 2)
Operating voltage		• 3.3V
Target board		Evaluation Kit RE01 256KB (RTK70E0118CXXXXXB)
Integrated Development Environment	GCC	Renesas e² studio 2021-01
	IAR	IAR Embedded Workbench for ARM Version 8.50.6
C compiler	GCC	GCC ARM Embedded Version 6.3.1.20170620 GNU 6-2017-q2-update
	IAR	IAR C/C++ Compiler for ARM Version 8.50.6
Debugger		Segger J-Link OB
I/O header Version		Rev1.00
Sample code Version		Rev1.05

3. Description of Software

This sample code uses the R_SPI driver to transmit data from SPI0 in master transmission mode and receive the transmitted data using SPI1 operating in slave receive mode. Transmission from SPI0 uses the DMAC and reception in SPI1 uses the DTC.

The sample code performs the following operations.

- Make initial settings of SPI0 and SPI1 after release from the reset state.
- Starts slave reception in SPI1 and master transmission from SPI0.
- Compares the transmit data with the receive data after a transmission end interrupt occurs in SPI0 and a reception end interrupt occurs in SPI1.
- Restarts slave reception in SPI1 and master transmission in SPI0 when the transmit and receive data match.

Table 3-1 Information of Sample Program Operation (SPI0)

Item	Setting
Master or slave mode	Master mode
Transfer rate	100 kbps
RSPCK phase	Data sampling on rising edges and data change on falling edges are selected.
RSPCK polarity	RSPCK is driven low in the idle state.
SPI LSB first	MSB first
Data transfer size	5 bytes (40 bits)

Table 3-2 Information of Sample Program Operation (SPI1)

Item	Setting
Master or slave mode	Slave mode
RSPCK phase	Data sampling on rising edges and data change on falling edges are selected.
RSPCK polarity	RSPCK is driven low in the idle state.
SPI LSB first	MSB first
Data transfer size	5 bytes (40 bits)

3.1 System Configuration

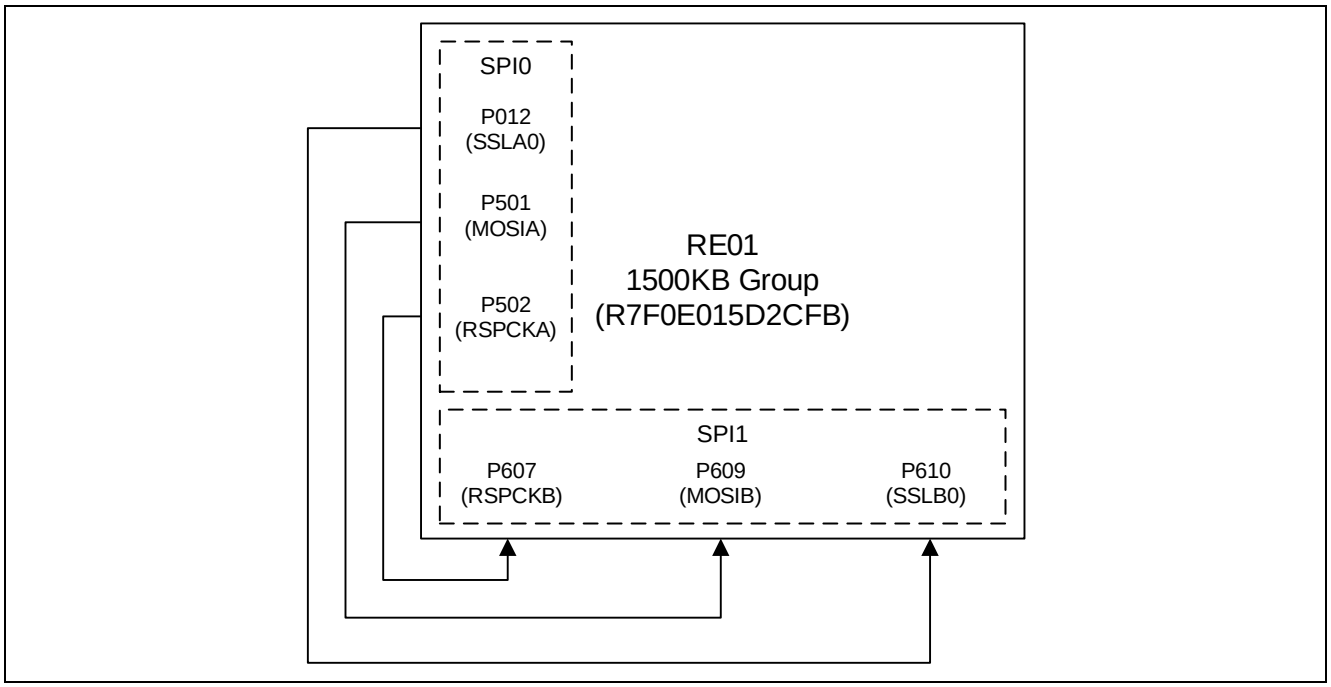


Figure 3.1 System Configuration for RE01 1500KB group

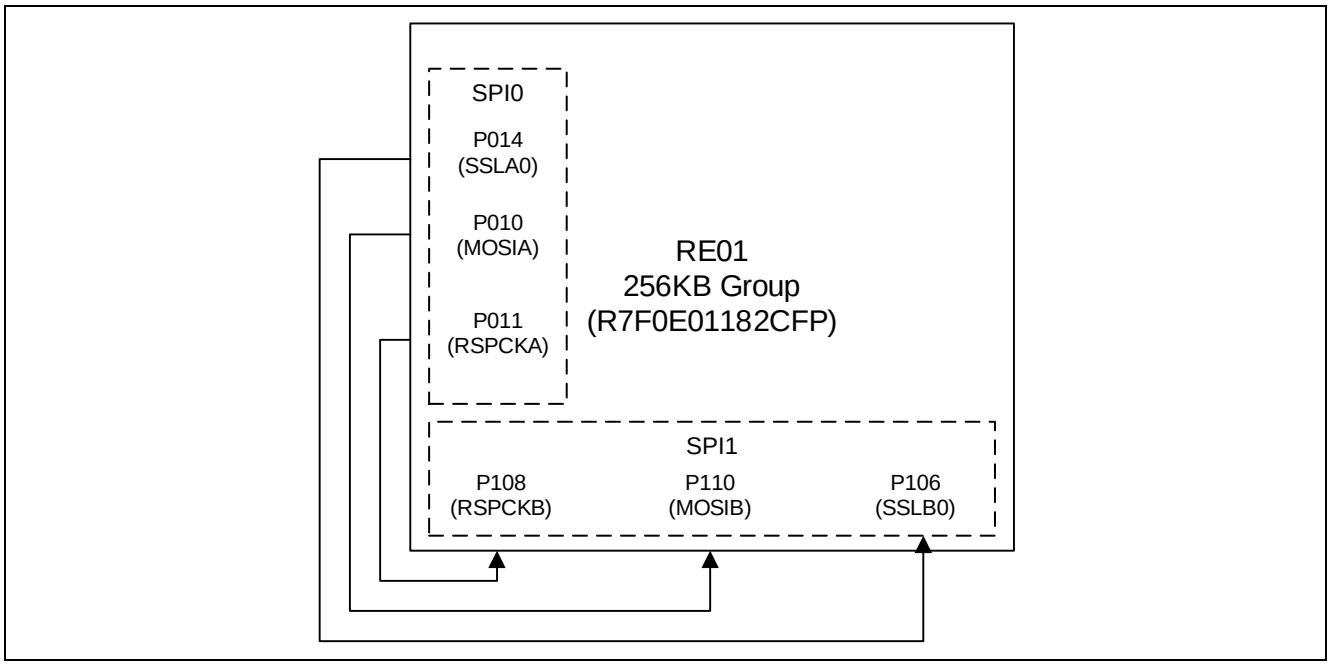


Figure 3.2 System Configuration for RE01 256KB group

3.2 Driver Configuration

3.2.1 Driver Configuration by the sample code for RE01 1500KB group

Table 3-3 Driver Configuration for 1500KB Group (1/2)

Item	Location of Change	Details of Change
Transferring the transmit data through the DMAC when SPI transmission ends	[r_spi_cfg.h] SPI0_TRANSMIT_CONTROL	• Setting change SPI_USED_DMAC0
Transferring the receive data through the DTC when SPI reception ends	[r_spi_cfg.h] SPI1_RECEIVE_CONTROL	• Setting change SPI_USED_DTC
Registering SPI0 idle interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI0_SPII	• Setting change (SYSTEM_IRQ_EVENT_NUMBER18)
Registering SPI1 transmit buffer empty interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI1_SPTI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER5)
Registering SPI1 receive buffer full interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI1_SPRI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER28)
Registering SPI1 error interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI1_SPEI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER7)
Registering DMAC0 transfer end interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_DMAC0_INT	• Setting change (SYSTEM_IRQ_EVENT_NUMBER0)

Table 3-4 Driver Configuration for 1500KB Group (2/2)

Item	Location of Change	Details of Change
Set P501 as MOSIA pin.	[pin.c] R_RSPI_Pinset_CH0() function	• Validate followings PFS->P501PFS_b.PMR = 0U; PFS->P501PFS_b.ASEL = 0U; PFS->P501PFS_b.ISEL = 0U; PFS->P501PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P501PFS_b.PMR = 1U;
Set P502 as RSPCKA pin.	[pin.c] R_RSPI_Pinset_CH0() function	• Validate followings PFS->P502PFS_b.PMR = 0U; PFS->P502PFS_b.ASEL = 0U; PFS->P502PFS_b.ISEL = 0U; PFS->P502PFS_b.PSEL = R_PIN_PRV_ RSPI _PSEL; PFS->P502PFS_b.PMR = 1U;
Set P012 as SSLA0 pin.	[pin.c] R_RSPI_Pinset_CH0() function	• Validate followings PFS->P012PFS_b.PMR = 0U; PFS->P012PFS_b.ASEL = 0U; PFS->P012PFS_b.ISEL = 0U; PFS->P012PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P012PFS_b.PMR = 1U;
Set P609 as MOSIB pin.	[pin.c] R_RSPI_Pinset_CH1() function	• Validate followings PFS->P609PFS_b.PMR = 0U; PFS->P609PFS_b.ASEL = 0U; PFS->P609PFS_b.ISEL = 0U; PFS->P609PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; • PFS->P609PFS_b.PMR = 1U;
Set P607 as RSPCKB pin.	[pin.c] R_RSPI_Pinset_CH1() function	• Validate followings PFS->P607PFS_b.PMR = 0U; PFS->P607PFS_b.ASEL = 0U; PFS->P607PFS_b.ISEL = 0U; PFS->P607PFS_b.PSEL = R_PIN_PRV_ RSPI _PSEL; PFS->P607PFS_b.PMR = 1U;
Set P610 as SSLB0 pin.	[pin.c] R_RSPI_Pinset_CH1() function	• Validate followings PFS->P610PFS_b.PMR = 0U; PFS->P610PFS_b.ASEL = 0U; PFS->P610PFS_b.ISEL = 0U; PFS->P610PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P610PFS_b.PMR = 1U;

3.2.2 Driver Configuration by the sample code for RE01 256KB group

Table 3-5 Driver Configuration for 256KB Group (1/2)

Item	Location of Change	Details of Change
Transferring the transmit data through the DMAC when SPI transmission ends	[r_spi_cfg.h] SPI0_TRANSMIT_CONTROL	• Setting change SPI_USED_DMAC0
Transferring the receive data through the DTC when SPI reception ends	[r_spi_cfg.h] SPI1_RECEIVE_CONTROL	• Setting change SPI_USED_DTC
Registering SPI0 idle interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI0_SPII	• Setting change (SYSTEM_IRQ_EVENT_NUMBER18)
Registering SPI1 transmit buffer empty interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI1_SPTI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER5)
Registering SPI1 receive buffer full interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI1_SPRI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER28)
Registering SPI1 error interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_SPI1_SPEI	• Setting change (SYSTEM_IRQ_EVENT_NUMBER7)
Registering DMAC0 transfer end interrupt to NVIC	[r_system_cfg.h] SYSTEM_CFG_EVENT_ NUMBER_DMAC0_INT	• Setting change (SYSTEM_IRQ_EVENT_NUMBER0)

Table 3-6 Driver Configuration for 256KB Group (2/2)

Item	Location of Change	Details of Change
Set P010 as MOSIA pin.	[pin.c] R_RSPI_Pinset_CH0() function	• Validate followings PFS->P010PFS_b.PMR = 0U; PFS->P010PFS_b.ASEL = 0U; PFS->P010PFS_b.ISEL = 0U; PFS->P010PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P010PFS_b.PMR = 1U;
Set P011 as RSPCKA pin.	[pin.c] R_RSPI_Pinset_CH0() function	• Validate followings PFS->P011PFS_b.PMR = 0U; PFS->P011PFS_b.ASEL = 0U; PFS->P011PFS_b.ISEL = 0U; PFS->P011PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P011PFS_b.PMR = 1U;
Set P014 as SSLA0 pin.	[pin.c] R_RSPI_Pinset_CH0() function	• Validate followings PFS->P014PFS_b.PMR = 0U; PFS->P014PFS_b.ASEL = 0U; PFS->P014PFS_b.ISEL = 0U; PFS->P014PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P014PFS_b.PMR = 1U;
Set P110 as MOSIB pin.	[pin.c] R_RSPI_Pinset_CH1() function	• Validate followings PFS->P110PFS_b.PMR = 0U; PFS->P110PFS_b.ASEL = 0U; PFS->P110PFS_b.ISEL = 0U; PFS->P110PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P110PFS_b.PMR = 1U;
Set P108 as RSPCKB pin.	[pin.c] R_RSPI_Pinset_CH1() function	• Validate followings PFS->P108PFS_b.PMR = 0U; PFS->P108PFS_b.ASEL = 0U; PFS->P108PFS_b.ISEL = 0U; PFS->P108PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P108PFS_b.PMR = 1U;
Set P106 as SSLB0 pin.	[pin.c] R_RSPI_Pinset_CH1() function	• Validate followings PFS->P106PFS_b.PMR = 0U; PFS->P106PFS_b.ASEL = 0U; PFS->P106PFS_b.ISEL = 0U; PFS->P106PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL; PFS->P106PFS_b.PMR = 1U;

3.3 List of Functions

The functions added to the sample code are described here.

main	
Overview	Main processing
Header	None
Declaration	void main(void)
Description	This function calls the system initialization function. Then, it sets up SPI communication and performs SPI transmission and reception. After data transmission and reception have finished, this function compares the transmitted and received data.
Argument	None
Return Value	None

system_init	
Overview	System initialization processing
Header	None
Declaration	static void system_init(void)
Description	This function initializes sections, the system, the R_LPM driver, and the R_LPM driver, and calls the IO power supply setting function.
Argument	None
Return Value	None

spi0_callback		
Overview	SPI0 transfer end callback processing	
Header	None	
Declaration	static void spi0_callback(uint32_t event)	
Description	After the SPI0 data transmission has completed, this function sets the transmission end flag.	
Argument	uint32_t event	Cause of callback ARM_SPI_EVENT_TRANSFER_COMPLETE (Transfer end)
Return Value	None	

spi1_callback		
Overview	SPI1 transfer end callback processing	
Header	None	
Declaration	static void spi1_callback(uint32_t event)	
Description	After the SPI1 has received the data, this function sets the reception end flag.	
Argument	uint32_t event	Cause of callback ARM_SPI_EVENT_TRANSFER_COMPLETE (Transfer end)
Return Value	None	

3.4 List of Constants

Table 3-7 shows a list of constants.

Table 3-7 Constants (User Changeable) Used in Sample Code

Constant Name	Setting	Description
SPI_TRANSFER_SPEED	100000U	SPI transfer rate
SPI_DATA_SIZE	5	SPI transfer data size

3.5 Flowcharts

Figure 3.3 shows a flowchart of the main processing.

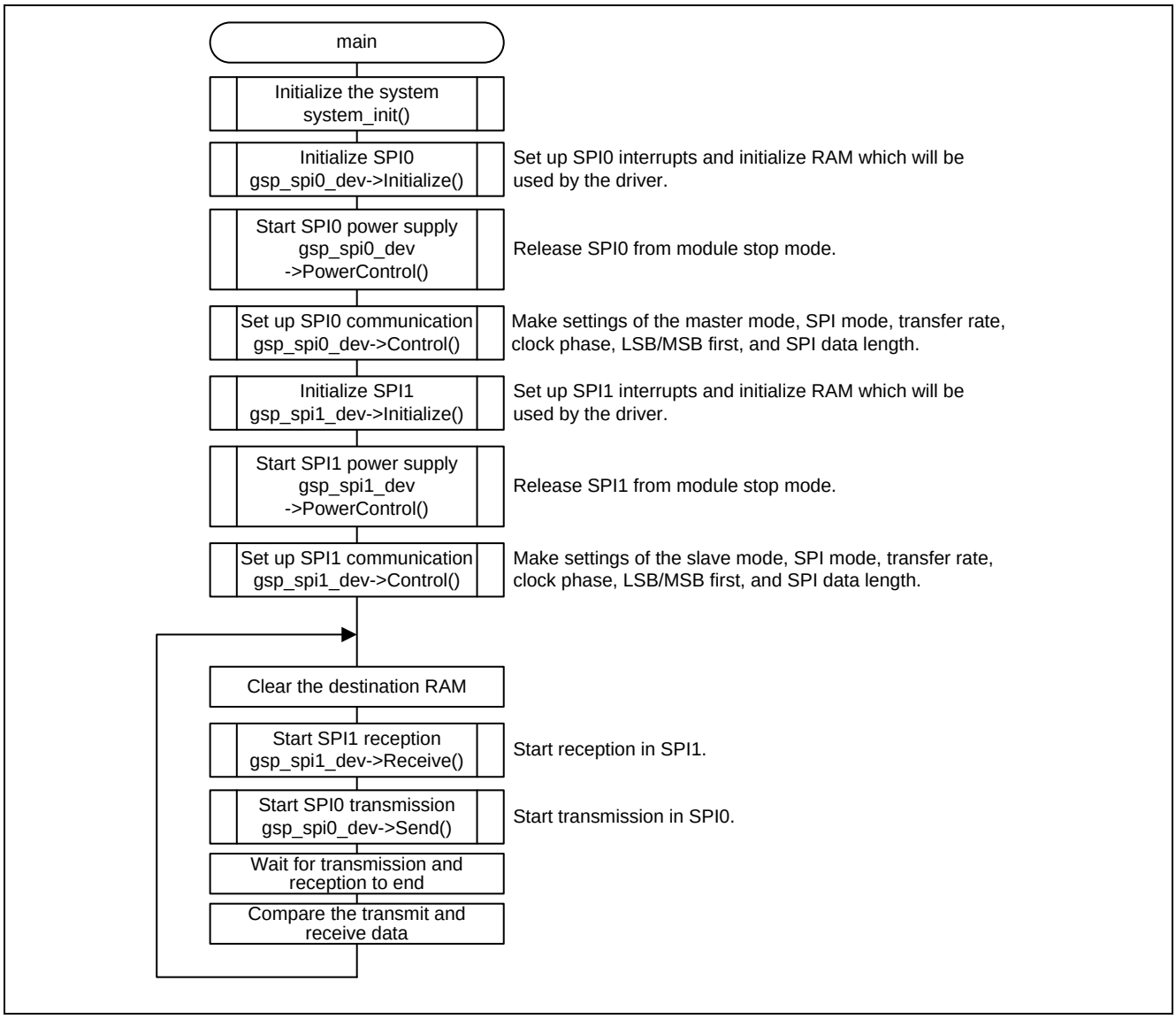


Figure 3.3 Main Processing

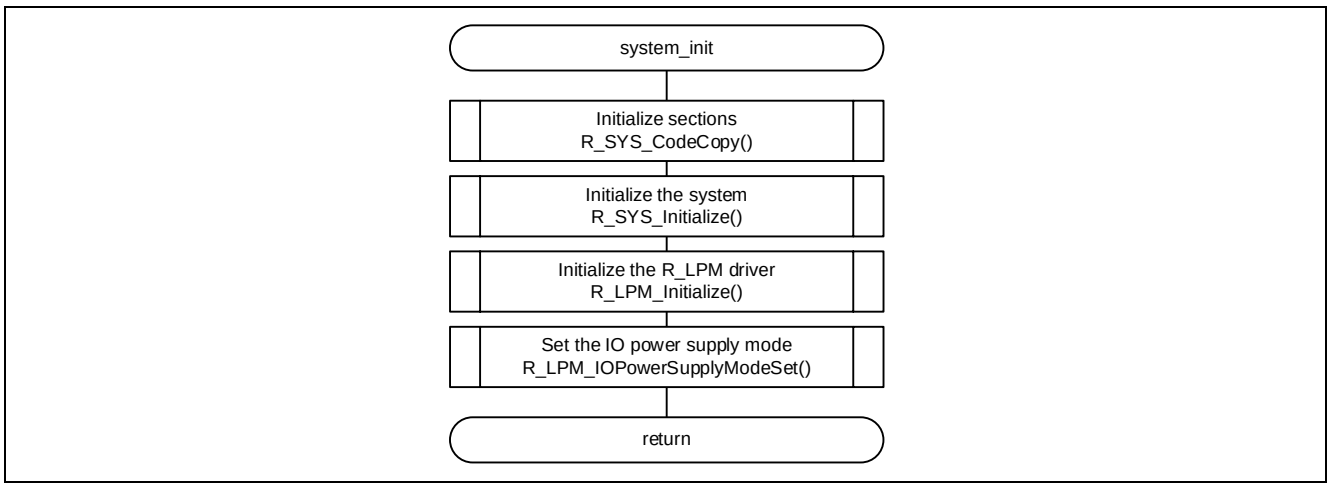


Figure 3.4 System Initialization Processing

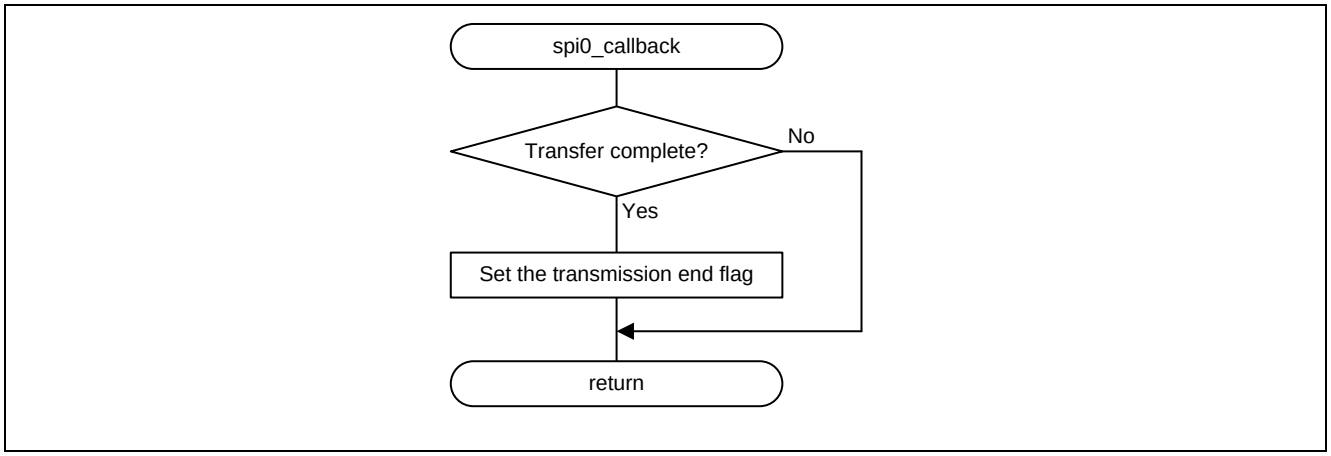


Figure 3.5 SPI0 Transfer End Callback Processing

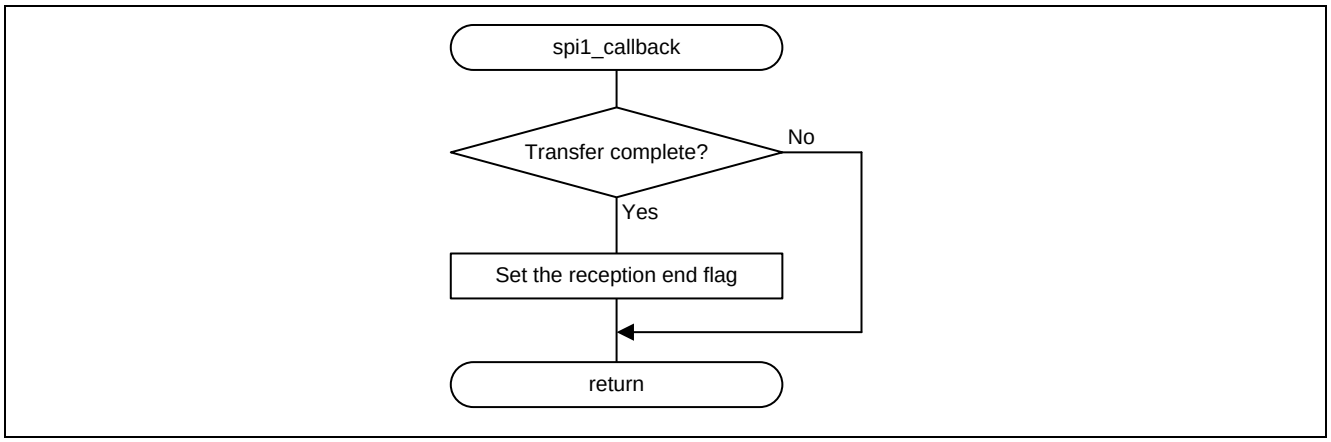


Figure 3.6 SPI1 Transfer End Callback Processing

4. SPI Communication Using the DTC and DMAC

This sample code uses DMAC0 for SPI0 transmission and uses the DTC for SPI1 reception. The following shows the settings for using the DMAC and DTC in r_spi_cfg.h and r_system_cfg.h.

4.1 DMAC Settings for SPI0

Table 4-1 and Table 4-2 show the settings for SPI0 (master transmission).

Table 4-1 Change of Setting in r_spi_cfg.h

Item	Location of Change	
Transferring the transmit data through DMAC0	SPI0_TRANSMIT_CONTROL	SPI_USED_INTERUPT → SPI_USED_DMAC0

Table 4-2 Change of Setting in r_system_cfg.h

Item	Location of Change	
Registering DMAC0 transfer end interrupt to NVIC	SYSTEM_CFG_EVENT_NUMBER_DMAC0_INT	SYSTEM_IRQ_EVENT_NUMBER0

4.2 DTC Settings for SPI1

Table 4-3 shows the setting for SPI1 (slave reception).

Table 4-3 Change of Setting in r_spi_cfg.h

Item	Location of Change	
Transferring the receive data through DTC	SPI1_RECEIVE_CONTROL	SPI_USED_INTERRUPT → SPI_USED_DTC

5. Specifications of Driver APIs

5.1 External Specification

This driver contains documents that describes the external API specification. These files are contained in the Driver Specification folder within the Documents.

6. Usage Notes of R_SPI Driver

This chapter introduces the main points regarding the usage of the R_SPI driver. Note: not all points regarding the usage of the driver are given here.

For other notes, please see the external specification document described in "5 Specifications of Driver APIs".

6.1 SPI Communication Using DMAC or DTC

When transmitting or receiving data using the DMAC or DTC, change the settings for controlling transmission and reception in r_spi_cfg.h.

Table 6-1 shows the definitions of configuration parameters for transmission and reception control. Table 6-2 shows the definitions of values indicating methods of transmission and reception control.

Table 6-1 Definitions of Configuration Parameters for Transmission and Reception Control (n = 0 or 1)

Definition	Initial Value	Description
SPIn_TRANSMIT_CONTROL	SPI_USED_INTERRUPT	Transmission control for SPIn (Initial value: Interrupt)
SPIn_RECEIVE_CONTROL	SPI_USED_INTERRUPT	Reception control for SPIn (Initial value: Interrupt)

Table 6-2 Definitions of Values Indicating Methods of Transmission and Reception Control

Definition	Initial Value	Description
SPI_USED_INTERRUPT	(0)	An interrupt is used for transmission/reception control.
SPI_USED_DMAC0	(1<<0)	DMAC0 is used for transmission/reception control.
SPI_USED_DMAC1	(1<<1)	DMAC1 is used for transmission/reception control.
SPI_USED_DMAC2	(1<<2)	DMAC2 is used for transmission/reception control.
SPI_USED_DMAC3	(1<<3)	DMAC3 is used for transmission/reception control.
SPI_USED_DTC	(1<<15)	DTC is used for transmission/reception control.

6.2 Registering Interrupts to NVIC

When using the SPI, register the interrupts to the NVIC in r_system_cfg.h.

Table 6-3 shows the definition of NVIC registration for each intended use. Figure 6.1 shows an example of registering interrupts to the NVIC.

Table 6-3 Definition of NVIC Registration for Each Intended Use

Intended Use	Definition of NVIC Registration	Remarks
Used for master transmission	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPTI	(*1)
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPII	
Used for master transmission and reception (*4)	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPTI	(*1)
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPRI	(*3)
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPII	
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPEI	
Used for slave transmission	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPTI	(*1)
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPEI	
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPTEND	
Used for slave transmission and reception (*4)	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPTI	(*1)
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPRI	(*2)
	SYSTEM_CFG_EVENT_NUMBER_SPIn_SPEI	

Note 1. When DMACm (m = 0 to 3) is used for transmission control, register SYSTEM_CFG_EVENT_NUMBER_DMACm_INT to the NVIC.

Note 2. When DMACm (m = 0 to 3) is used for reception control, register SYSTEM_CFG_EVENT_NUMBER_DMACm_INT to the NVIC.

Note 3. When DMACm (m = 0 to 3) is used for reception control, this interrupt does not need to be registered to the NVIC.

Note 4. Even when only reception is to be performed, dummy data should be transmitted. Accordingly, the interrupt for transmission should also be registered.

```
...
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_AM
            (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)   /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPRI
            (SYSTEM_IRQ_EVENT_NUMBER0)             /*!< Numbers 0/4/8/12/16/20/24/28 only */
#define SYSTEM_CFG_EVENT_NUMBER_SOL_DH
            (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)   /*!< Numbers 0/8/16/24 only */
...
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_TXI
           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPTI
           (SYSTEM_IRQ_EVENT_NUMBER1)              /*!< Numbers 1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_SOL_DL
           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 1/9/17/25 only */
...
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_TEI
           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPII
           (SYSTEM_IRQ_EVENT_NUMBER2)              /*!< Numbers 2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPTEND
           (SYSTEM_IRQ_EVENT_NUMBER6)              /*!< Numbers 2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_USBFS_USBI
           (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 2/10/18/26 only */
...
```

Figure 6.1 Example of Registering Interrupts to NVIC (When Using SPI0)

6.3 Outputting the SS Signal under Software Control

To control the SS signal output by software (the Control function using the ARM_SPI_CONTROL_SS command), specify the pin to be used to output the SS signal in r_spi_cfg.h.

Figure 6.2 shows an example of pin setting when using the SS pin under software control.

```
...
/* When using the ARM_SPI_CONTROL_SS command, cancel the following comment and set the terminal
to use */
#define SPI0_SS_PORT PORT1->PODR              /* Uncommented. Port 1 is selected. */
#define SPI0_SS_PIN 7                           /* P107 is selected as the SS pin */
...
```

Figure 6.2 Example of Pin Setting When Using the SS Pin under Software Control

6.4 Pin Settings

The pins to be used by this driver will be set when the operating mode is selected in the Control function. To change the pins to be used, modify code in the R_SPI_Pinset_CHn and R_SPI_Pinclr_CHn functions (n = 0 or 1) of pin.c. Pin names of SPI function have added _A, _B, _C, and _D suffixes. When assigning SPI functions, select the functional pins with the same suffix. (Note)
Also, the pin setting example when setting the SPI0 MISOA pin to P500, the MOSIA pin to P010, the RSPCKA pin to P011, the SSLA0 pin to P012, and the SSLA1 to SSLA3 pins to unused is shown in Figure 6-3, Figure 6-4 and Figure 6-5.

Note. In the case of SPI function, the same signal names are present with the suffixes “_A”, “_B” “_C” and “_D” attached. These indicate groups in terms of timing adjustment, and signals from different groups cannot be used at the same time. The exceptions are the RSPCKA_C and MOSIA_C signals for the SPI and the SSLB0_D signal, which can be used at the same time as signals from group B.

```
/*****************************************************************************//**
* @brief This function sets Pin of RSPI0.
* @note  Several pin names have added _A, _B, and _C suffixes.@n
*       When assigning the SPI functions, select the functional pins with the same suffix.@n
*       Comment out the terminal of unused suffix.
*****************************************************************************/
/* Function Name : R_RSPI_Pinset_CH0 */
void R_RSPI_Pinset_CH0(void)  // @suppress("API function naming") @suppress("Function length")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

//    /* MISOA_A : P105 */
//    PFS->P105PFS_b.PMR  = 0U;
//    PFS->P105PFS_b.ASEL = 0U;
//    PFS->P105PFS_b.ISEL = 0U;
//    PFS->P105PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P105PFS_b.PMR  = 1U;

/* Set P500 as MISOA pin.*/
    /* MISOA_B : P500 */
    PFS->P500PFS_b.PMR  = 0U;
    PFS->P500PFS_b.ASEL = 0U;
    PFS->P500PFS_b.ISEL = 0U;
    PFS->P500PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
    PFS->P500PFS_b.PMR  = 1U;

//    /* MOSIA_A : P104 */
//    PFS->P104PFS_b.PMR  = 0U;
//    PFS->P104PFS_b.ASEL = 0U;
//    PFS->P104PFS_b.ISEL = 0U;
//    PFS->P104PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P104PFS_b.PMR  = 1U;

/* Set P010 as MOSIA pin.*/
    /* MOSIA_B : P010 */
    PFS->P010PFS_b.PMR  = 0U;
    PFS->P010PFS_b.ASEL = 0U;
    PFS->P010PFS_b.ISEL = 0U;
    PFS->P010PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
    PFS->P010PFS_b.PMR  = 1U;

    /* MOSIA_C : P501 */
//    PFS->P501PFS_b.PMR  = 0U;
//    PFS->P501PFS_b.ASEL = 0U;
//    PFS->P501PFS_b.ISEL = 0U;
//    PFS->P501PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P501PFS_b.PMR  = 1U;
```

Figure 6-3 Coding Example for Changing Pin Configuration (1/3)

```
//    /* RSPCKA_A : P107 */
//    PFS->P107PFS_b.PMR  = 0U;
//    PFS->P107PFS_b.ASEL = 0U;
//    PFS->P107PFS_b.ISEL = 0U;
//    PFS->P107PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P107PFS_b.PMR  = 1U;

/* Set P011 as RSPCKA pin.*/
    /* RSPCKA_B : P011 */
    PFS->P011PFS_b.PMR  = 0U;
    PFS->P011PFS_b.ASEL = 0U;
    PFS->P011PFS_b.ISEL = 0U;
    PFS->P011PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
    PFS->P011PFS_b.PMR  = 1U;

    /* RSPCKA_C : P502 */
//    PFS->P502PFS_b.PMR  = 0U;
//    PFS->P502PFS_b.ASEL = 0U;
//    PFS->P502PFS_b.ISEL = 0U;
//    PFS->P502PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P502PFS_b.PMR  = 1U;

//    /* SSLA0_A : P103 */
//    PFS->P103PFS_b.PMR  = 0U;
//    PFS->P103PFS_b.ASEL = 0U;
//    PFS->P103PFS_b.ISEL = 0U;
//    PFS->P103PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P103PFS_b.PMR  = 1U;

/* Set P012 as SSLA0 pin.*/
    /* SSLA0_B : P012 */
    PFS->P012PFS_b.PMR  = 0U;
    PFS->P012PFS_b.ASEL = 0U;
    PFS->P012PFS_b.ISEL = 0U;
    PFS->P012PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
    PFS->P012PFS_b.PMR  = 1U;

//    /* SSLA1_A : P102 */
//    PFS->P102PFS_b.PMR  = 0U;
//    PFS->P102PFS_b.ASEL = 0U;
//    PFS->P102PFS_b.ISEL = 0U;
//    PFS->P102PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P102PFS_b.PMR  = 1U;

...

    /* Enable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);
}/* End of function R_RSPI_Pinset_CH0() */
```

Figure 6-4 Coding Example for Changing Pin Configuration (2/3)

```
/*****************************************************************************//**
* @brief This function clears the pin setting of RSPI0.
*****************************************************************************/
/* Function Name : R_RSPI_Pinclr_CH0 */
void R_RSPI_Pinclr_CH0(void)  // @suppress("API function naming")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

//    /* MISOA_A : P105 */
//    PFS->P105PFS &= R_PIN_PRV_CLR_MASK;

/* Set P500 as MISOA pin.*/
    /* MISOA_B : P500 */
    PFS->P500PFS &= R_PIN_PRV_CLR_MASK;

//    /* MOSIA_A : P104 */
//    PFS->P104PFS &= R_PIN_PRV_CLR_MASK;

/* Set P010 as MOSIA pin.*/
    /* MOSIA_B : P010 */
    PFS->P010PFS &= R_PIN_PRV_CLR_MASK;

//    /* MOSIA_C : P501 */
//    PFS->P501PFS &= R_PIN_PRV_CLR_MASK;

//    /* RSPCKA_A : P107 */
//    PFS->P107PFS &= R_PIN_PRV_CLR_MASK;

/* Set P011 as RSPCKA pin.*/
    /* RSPCKA_B : P011 */
    PFS->P011PFS &= R_PIN_PRV_CLR_MASK;

//    /* RSPCKA_C : P502 */
//    PFS->P502PFS &= R_PIN_PRV_CLR_MASK;

//    /* SSLA0_A : P103 */
//    PFS->P103PFS &= R_PIN_PRV_CLR_MASK;

/* Set P012 as SSLA0 pin.*/
    /* SSLA0_B : P012 */
    PFS->P012PFS &= R_PIN_PRV_CLR_MASK;

//    /* SSLA1_A : P102 */
//    PFS->P102PFS &= R_PIN_PRV_CLR_MASK;

//    /* SSLA1_B : P013 */
//    PFS->P013PFS &= R_PIN_PRV_CLR_MASK;

//    /* SSLA2_A : P101 */
//    PFS->P101PFS &= R_PIN_PRV_CLR_MASK;

//    /* SSLA2_B : P014 */
//    PFS->P014PFS &= R_PIN_PRV_CLR_MASK;

//    /* SSLA3_A : P100 */
//    PFS->P100PFS &= R_PIN_PRV_CLR_MASK;

    /* SSLA3_B : P015 */
//    PFS->P015PFS &= R_PIN_PRV_CLR_MASK;

    /* Enable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);
}/* End of function R_RSPI_Pinclr_CH0() */
```

Figure 6-5 Coding Example for Changing Pin Configuration (3/3)

6.5 Resuming communication in slave mode and CPHA0

When CPHA (clock phase) is set to 0 (data sampling at the rising edge, data change at the falling edge) in slave mode, resuming communication during the last half of RSPCK cycle may cause incorrect operations such as underrun error or bit shifts.

After calling the callback function or after ensuring that SPI status is ready by using GetStatus function, wait for half RSPCK cycle before restarting the communication.

Figure 6-6 shows the example of restarting communication in slave mode when CPHA0 is set.

```
static uint8_t tx_data[3] = {0x01, 0x02, 0x03};

/**********************************************************************************************
* callback function
**********************************************************************************************/
static void spi_callback(uint32_t event)
{
    switch( event )
    {
        case ARM_SPI_EVENT_TRANSFER_COMPLETE:
        {
            /* RSPCK half cycle wait (5us for communication speed 100kbps) */
            R_SYS_SoftwareDelay(5, SYSTEM_DELAY_UNITS_MICROSECONDS);
            /* restart */
            spi0Drv->Send(&tx_data[0], 3);
        }
        break;

        case ARM_SPI_EVENT_DATA_LOST:
        default:
        {
            /* Describes processing when a communication error occurs */
        }
        break;
    }

}   /* End of function spi_callback() */
```

Figure 6-6 Example of restarting communication in slave mode and CPHA0

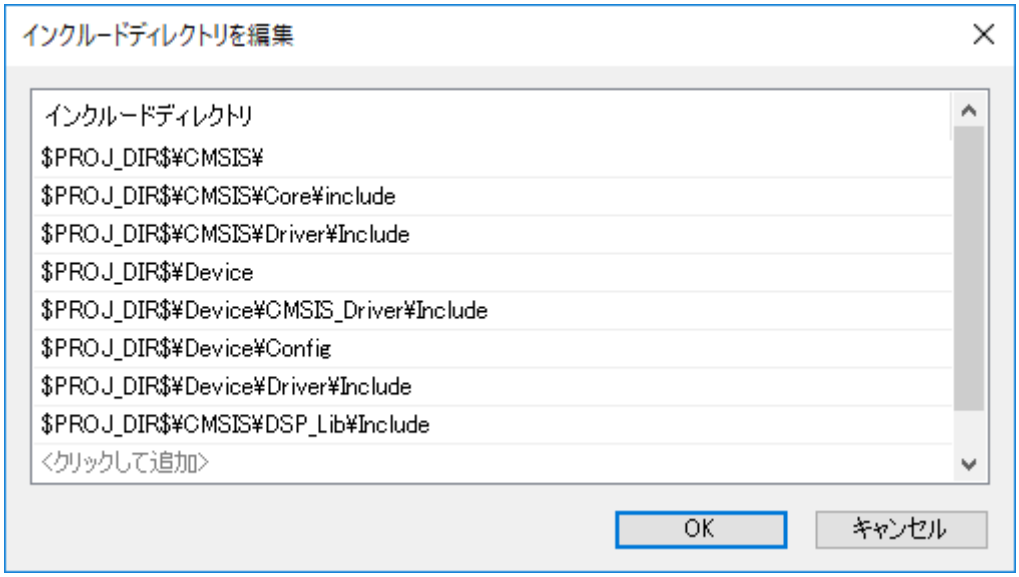
7. Troubleshooting

7.1 Occurrence of Build Error with IAR Compiler

A-1) Have the include directories been specified correctly?

When using EWARM, we recommend that the include directories are specified as shown in the example below.

The include directories can be specified from IDE Options [C/C++ Compiler] → [Preprocessor].



7.2 Occurrence of HardFault Error when API of CMSIS Driver Is Called

A) The API has possibly not been copied to RAM.

Before calling an API function that is mapped to RAM, make sure that it has been copied to RAM by the R_SYS_CodeCopy function. For details, refer to the related document No. R01AN4660.

7.3 Peripheral Function Fails to Operate when API Is Called

A) Has the API been set up correctly?

Check the API's return value to see if an error has occurred.

Errors are often caused by problems related to interrupts not being set in r_system_cfg.h. For details, refer to the related document No. R01AN4660.

7.4 Normal API Return Value But No Pin Output from Peripheral Function

A) Are the pin settings correct?

Check to make sure the pins have been set up correctly by the functions in pin.c.

For details, refer to the related document No. R01AN4660.

7.5 Peripheral Function's Input or Output Does Not Operate as Expected

A) Check to make sure the VOCR register has been set up correctly before making the initial settings for peripheral functions.

For details, refer to the related document No. R01AN4660.

8. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

9. Reference Documents

User's Manual: Hardware

RE01 1500KB Group User's Manual: Hardware R01UH0796

RE01 256KB Group User's Manual: Hardware R01UH0894

(The latest version can be downloaded from the Renesas Electronics website.)

RE01 1500KB, 256KB CMSIS Package Startup Guide

RE01 1500KB, 256KB Group Startup Guide to Development Using CMSIS Package R01AN4660

(The latest version can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News

(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Development Tools

(The latest version can be downloaded from the Renesas Electronics website.)

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	Sep.19.2019	-	First edition issued
1.01	Jan.20.2020	-	Add RE01 256KB group.
1.02	Mar.19.2020	3,6 10,21,22 - Program	RE01 256KB group target board changed to Evaluation Kit RE01 256KB Changed the pin setting description according to the specification of RE01 1500KB group pin.c (ver.1.10) Clerical error correction Replaced CMSIS Driver Package - RE01 1500KB: CMSIS Driver Package Rev.1.10 - RE01 256KB: CMSIS Driver Package Rev.0.80
1.03	Jun.01.2020	3 6 Program (256KB)	Change the name of the sample code project Updated "Operating Conditions" Replaced CMSIS Driver Package - RE01 256KB: CMSIS Driver Package Rev.1.00
1.04	Nov.05.2020	- 6 Program	Clerical error correction Updated "Integrated Development Environment" and "C compiler". Remove unnecessary interrupt settings (SPTI for SPI0 and SPTEND for SPI1)
1.05	Jan.11.2021	3 8 10, 12 Program	Updated "Pins Used" Change the name of the sample code project Updated system configuration The SSLA0 pin setting is described in Driver Configuration. Enable SSL terminal of SPI0 (master) Change the name of the sample code project

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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/.