RX220 Group

Communication Example Using the RSPI

R01AN1807EJ0100
Rev. 1.00
Dec 16, 2013

Abstract

This document describes a method of full-duplex synchronous serial communications using the SPI operation (four-wire method) of the serial peripheral interface (RSPI) in the RX220 Group.

The sample code in this application note registers projects for the master device (master) and slave device (slave) in one workspace. The master or slave is selected by the active project in the High-performance Embedded Workshop.

Products

- RX220 Group 100-pin package with a ROM size between 64 KB and 256 KB
- RX220 Group 64-pin package with a ROM size between 32 KB and 256 KB
- RX220 Group 48-pin package with a ROM size between 32 KB and 256 KB

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

Full-duplex synchronous serial communications are performed between one master and two slaves (slave 0 and slave 1) using the SPI operation (four-wire method) of the RSPI.

The master transmits and receives 3-byte data to and from slave 0. When the 3-byte transmission and reception have been completed, LED0 is turned on. Then the master transmits and receives 3-byte data to and from slave 1. When the 3-byte transmission and reception have been completed, LED1 is turned on. If an error occurs during a transmission or reception, the operation is terminated and LED2 is turned on.

Slave 0 and slave 1 transmit and receive 3-byte data to and from the master. When the 3-byte transmission and reception have been completed, LED1 is turned on. If an error occurs during a transmission or reception, the operation is terminated and LED2 is turned on.

- RSPI mode: SPI operation (four-wire method)
- Communication mode: full-duplex synchronous serial communications
- Transfer rate: 6.25 kbps
- Transfer bit length: 8 bits
- Parity: None
- Sequence length: 1 sequence
- Number of frames: 1 frame

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows a Usage Example.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
RSPI	Full-duplex synchronous serial communications
I/O ports	Turn on LEDs

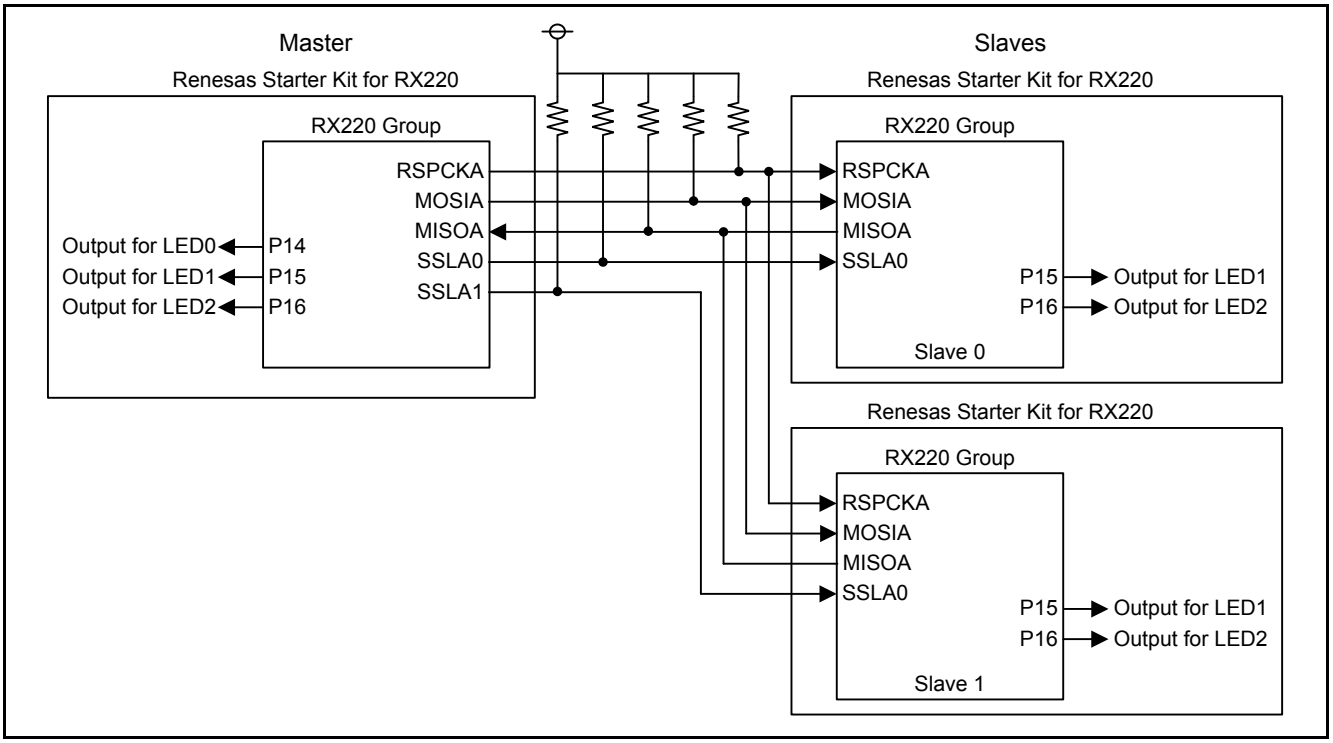


Figure 1.1 Usage Example

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F52206BDFP (RX220 Group)
Operating frequencies	- Main clock: 20 MHz - System clock (ICLK): 20 MHz (main clock divided by 1) - Peripheral module clock B (PCLKB): 20 MHz (main clock divided by 1)
Operating voltage	5.0 V
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01
	Compile options -cpu=rx200 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo (The default setting is used in the integrated development environment.)
iodefine.h version	Version 1.0A
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit for RX220 (product part no.: R0K5RX220C000BE)

3. Reference Application Note

For additional information associated with this document, refer to the following application note.

- RX220 Group Initial Setting Rev. 1.00 (R01AN1494EJ0100_RX220)

The initial setting functions in the reference application note are used in the sample code in this application note. The revision number of the reference application note is the one when this application note was made. However the latest version is always recommended. Visit the Renesas Electronics Corporation website to check and download the latest version.

4. Hardware (Master)

4.1 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

The number of pins in the sample code is set for the 100-pin package. When using products with less than 100 pins, select pins appropriate to the product used.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P14	Output	Outputs a signal for LED0 (completion of RSPI transmission to/reception from slave 0)
P15	Output	Outputs a signal for LED1 (completion of RSPI transmission to/reception from slave 1)
P16	Output	Outputs a signal for LED2 (RSPI transmit/receive error)
PA0/SSLA1	Output	Outputs a signal to select slave 1.
PA4/SSLA0	Output	Outputs a signal to select slave 0.
PC5/RSPCKA	Output	Clock output pin
PC6/MOSIA	Output	Data output pin
PC7/MISOA	Input	Data input pin

5. Software (Master)

After a reset, the user interface function (RSPI initialization) is called to initialize the RSPI.

After the initialization, slave 0 is specified, the user interface function (RSPI transmit/receive start) is called, and transmission and reception are enabled. When the 3-byte transmission and reception have been completed, RSPI transmission and reception are disabled, and the callback function (completion of RSPI transmission to/reception from slave 0) is called. LED0 is turned on with the callback function.

After the transmission and reception have been completed between the master and slave 0, transmission and reception between the master and slave 1 are performed in the same manner. After the transmission and reception have been completed, the callback function (completion of RSPI transmission to/reception from slave 1) is called. LED1 is turned on with the callback function.

If a transmit/receive error occurs, RSPI transmission and reception are disabled, the callback function (RSPI transmit/receive error) is called, and LED2 is turned on.

Settings for the peripheral function are as follows and Figure 5.1 shows the Software Configuration.

RSPI

- RSPI mode: SPI operation (four-wire method)
- Communication mode: Full-duplex synchronous serial communications
- Transfer rate: 6.25 kbps
- Clock source: PCLKB (20 MHz)
- Transfer bit length: 8 bits
- Parity: None
- Sequence length: 1 sequence
- Number of frames: 1 frame
- Error detection: Overrun error
- Interrupt source: RSPI error interrupt (SPEI) enabled
  RSPI receive interrupt (SPRI) enabled
  RSPI transmit interrupt (SPTI) enabled
  RSPI idle interrupt (SPII) enabled

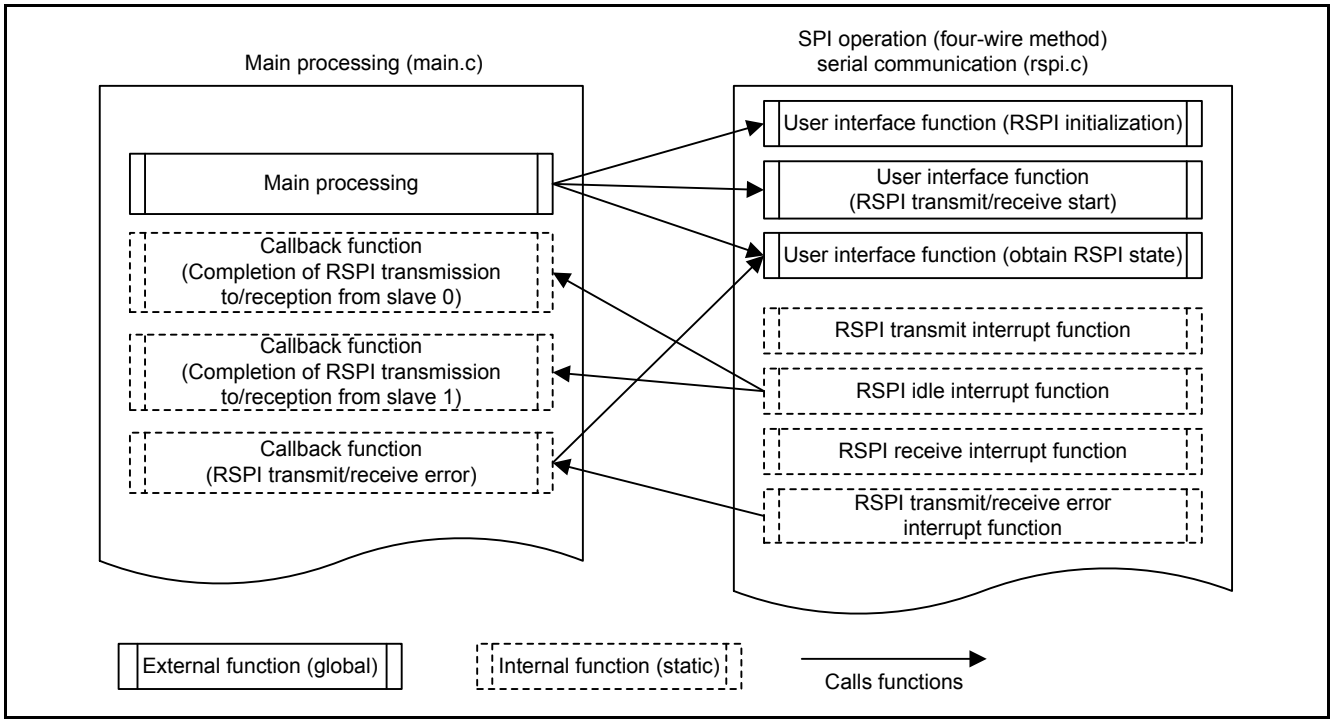


Figure 5.1 Software Configuration

5.1 Operation Overview

Figure 5.2 and Figure 5.3 show the timing of serial communication with the SPI operation (four-wire method), and (1) to (9) in the figures correspond to numbers in the operation descriptions below.

(1) Initialization
Initializes the RSPI using the user interface function (RSPI initialization).

(2) Starting transmission to/reception from slave 0
Calls the user interface function (RSPI transmit/receive start) with slave 0 selected as the argument.
With the user interface function, verifies the SPSR.IDLNF bit. When the bit is 1 (RSPI is in the transfer state), returns RSPI_BUSY (RSPI transmission/reception being processed). When the bit is 0 (RSPI is in the idle state), sets the transmit/receive busy flag to 1 and the SPCMD0.SSLA[2:0] bits to 000b (SSL0). Sets the SPCR.SPEIE bit to 1 (enables the generation of RSPI error interrupt requests), the SPCR.SPTIE bit to 1 (enables the generation of RSPI transmit interrupt requests), the SPCR.SPE bit to 1 (enables the RSPI function), and SPCR.SPRIE bit to 1 (enables the generation of RSPI receive interrupt requests). Sets the IEN bits for the RSPI error interrupt, RSPI receive interrupt, and RSPI transmit interrupt to 1 and starts a transmission and reception.

(3) Transmitting data to slave 0
In the RSPI transmit interrupt handling, writes the value in the transmit buffer for slave 0 to the SPDR register. When the last data has been written, sets the SPTIE bit to 0 (disables the generation of RSPI transmit interrupt requests) and the SPCR2.SPIIE bit to 1 (enables the generation of idle interrupt requests).

(4) Receiving data from slave 0
In the RSPI receive interrupt handling, writes the value in the SPDR register to the receive buffer for slave 0. When the last data has been received, sets the SPRIE bit to 0 (disables the generation of RSPI receive interrupt requests) and the SPEIE bit to 0 (disables the generation of RSPI error interrupt requests).

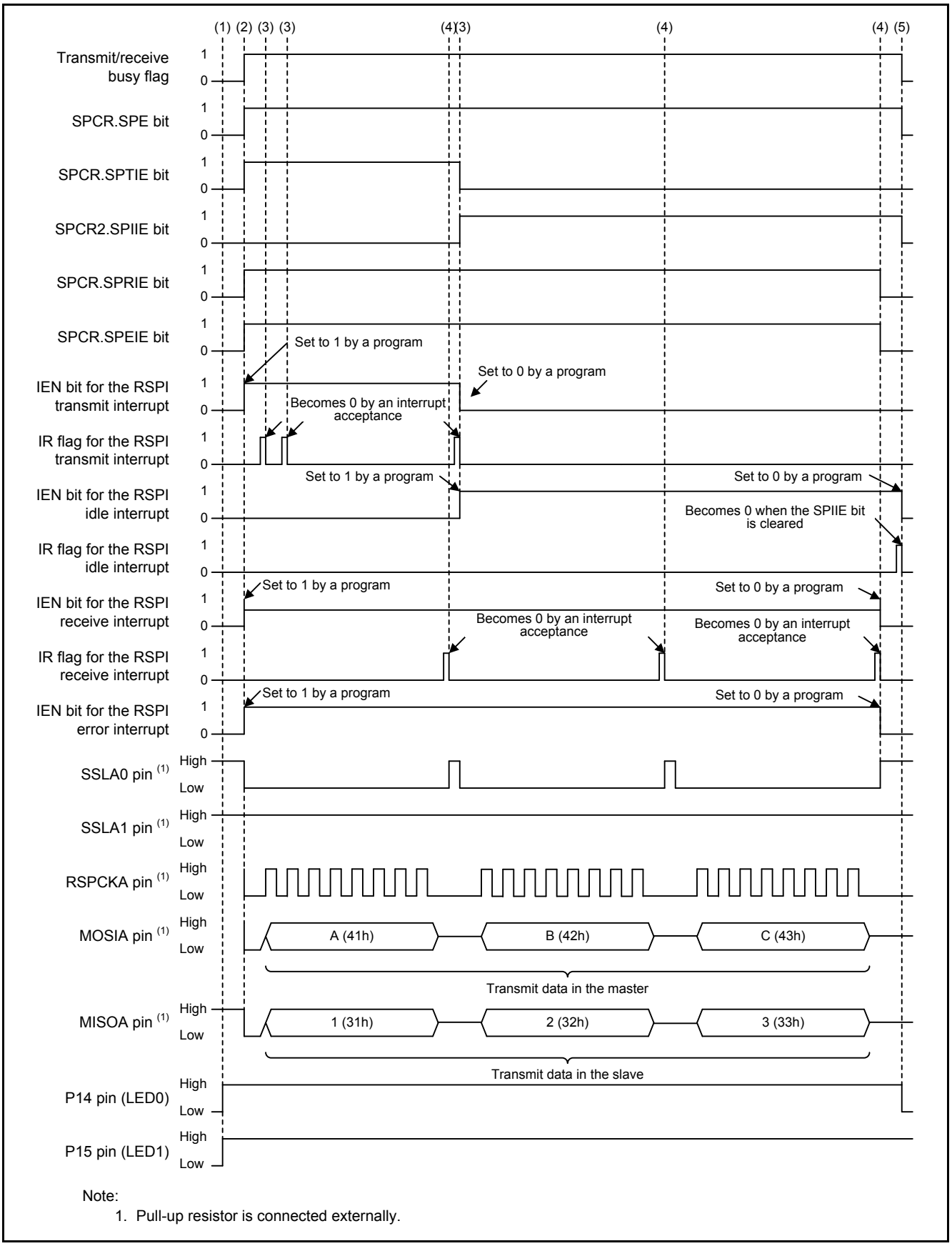
(5) Completing the transmission to/reception from slave 0
When the transmission and reception for the last data have been completed, the RSPI idle interrupt request is generated. In the RSPI idle interrupt handling, sets the SPE bit to 0 (disables the RSPI function), the SPIIE bit to 0 (disables the generation of idle interrupt requests), the transmit/receive busy flag to 0, and calls the callback function (completion of RSPI transmission to/reception from slave 0). Then LED0 is turned on.

(6) Starting a transmission to/reception from slave 1
Sets 001b (SSL1) to the SPCMD0.SSLA[2:0] bits and performs the same operations as (2) above to start a transmission and reception.

(7) Transmitting data to slave 1
In the RSPI transmit interrupt handling, writes the value in the transmit buffer for slave 1 to the SPDR register. Performs the same operations as (3) above to transmit data.

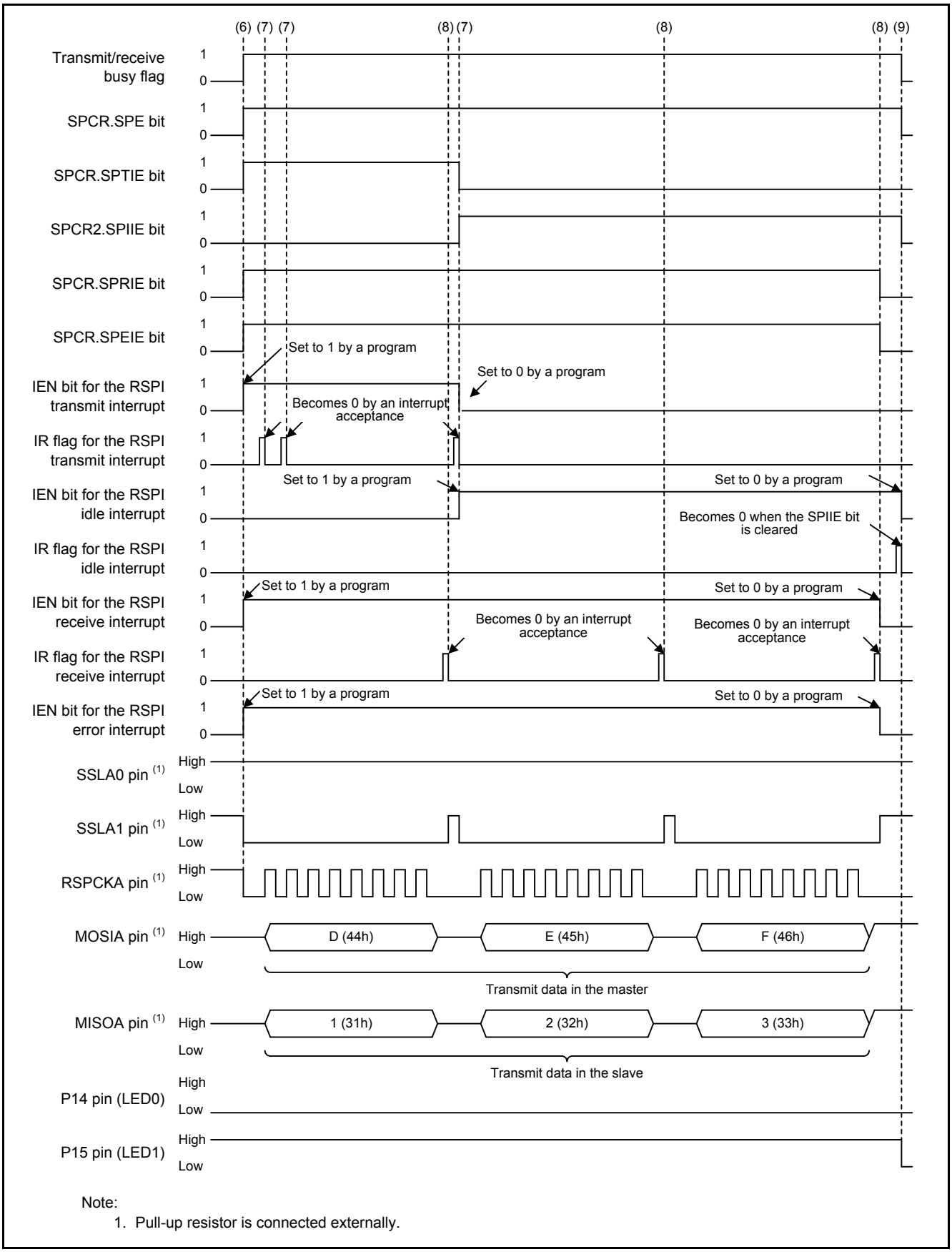
(8) Receiving data from slave 1
In the RSPI receive interrupt handling, writes the value in the SPDR register to the receive buffer for slave 1. Performs the same operations as (4) above to receive data.

(9) Completing the transmission to/reception from slave 1
Performs the same operations as (5) above to complete transmission and reception. Calls the callback function (completion of RSPI transmission to/reception from slave 1). Then LED1 is turned on.



Note:
1. Pull-up resistor is connected externally.

Figure 5.2 Timing of Serial Communication with SPI Operation (Four-Wire Method) when Transmitting to/Receiving from Slave 0



Note:
1. Pull-up resistor is connected externally.

Figure 5.3 Timing of Serial Communication with SPI Operation (Four-Wire Method) when Transmitting to/Receiving from Slave 1

5.2 File Composition

Table 5.1 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 5.1 Files Used in the Sample Code

File Name	Outline	Remarks
main.c	Main processing	
r_init_stop_module.c	Stop processing for active peripheral functions after a reset	
r_init_stop_module.h	Header file for r_init_stop_module.c	
r_init_non_existent_port.c	Nonexistent port initialization	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Clock initialization	
r_init_clock.h	Header file for r_init_clock.c	
rspi.c	Serial communication with SPI operation (four-wire method)	
rspi.h	Header file for rspi.c	

5.3 Option-Setting Memory

Table 5.2 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.2 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	The IWDT is stopped after a reset.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

5.4 Constants

Table 5.3 to Table 5.5 list the Constants Used in the Sample Code.

Table 5.3 Constants Used in the Sample Code (main.c)

Constant Name	Setting Value	Contents
LED0_REG_PODR	PORT1.PODR.BIT.B4	LED0 output data store bit
LED0_REG_PDR	PORT1.PDR.BIT.B4	LED0 I/O select bit
LED0_REG_PMR	PORT1.PMR.BIT.B4	LED0 pin mode control bit
LED1_REG_PODR	PORT1.PODR.BIT.B5	LED1 output data store bit
LED1_REG_PDR	PORT1.PDR.BIT.B5	LED1 I/O select bit
LED1_REG_PMR	PORT1.PMR.BIT.B5	LED1 pin mode control bit
LED2_REG_PODR	PORT1.PODR.BIT.B6	LED2 output data store bit
LED2_REG_PDR	PORT1.PDR.BIT.B6	LED2 I/O select bit
LED2_REG_PMR	PORT1.PMR.BIT.B6	LED2 pin mode control bit
LED_ON	0	LED output data: Turned on
LED_OFF	1	LED output data: Turned off
TR_SIZE	3	Transmission/reception size
BUF_SIZE	TR_SIZE	Buffer size

Table 5.4 Constants Used in the Sample Code (rspi.c)

Constant Name	Setting Value	Contents
SPSR_ERROR_FLAGS	0Dh	Bit pattern of an error flag in the RSPI.SPSR register
B_RSPI_BUSY	state.bit.b_rspi_busy	Transmit/receive busy flag 0: Transmission/reception ready 1: Transmission/reception busy
B_RX_ORER	state.bit.b_rx_orer	Overrun error flag 0: Overrun error not occurred 1: Overrun error occurred

Table 5.5 Constants Used in the Sample Code (rspi.h)

Constant Name	Setting Value	Contents
RSPI_OK	00h	Return value of the RSPI_PreTrans function: RSPI transmit/receive start
RSPI_NOT_IDLE	01h	Return value of the RSPI_PreTrans function: RSPI transmission/reception being processed
RSPI_NG	02h	Return value of the RSPI_PreTrans function: Argument error
RSPI_SSL0	0000h	Setting value of the SSL signal assertion setting bit in RSPI command registers 0 to 7 (SSL0 selected).
RSPI_SSL1	0010h	Setting value of the SSL signal assertion setting bit in RSPI command registers 0 to 7 (SSL1 selected).

5.5 Structure/Union List

Figure 5.4 shows the Structure/Union Used in the Sample Code.

```
#pragma bit_order     left          /* Bit field order: The bit field members are allocated from upper bits */
#pragma unpack                      /* The boundary alignment value for structure members: Alignment by member type */
typedef   union
{
    uint8_t    byte;
    struct
    {
        uint8_t    b_rspi_busy :1;  /* Transmit/receive busy flag   0: Transmission/reception ready   1: Transmission/reception busy */
        uint8_t    b_rx_orer   :1;  /* Overrun error flag           0: Overrun error not occurred     1: Overrun error occurred */
        uint8_t                :6;  /* Not used */
    } bit;
} rspi_state_t;
#pragma packoption                  /* End of specification for the boundary alignment value for structure members */
#pragma bit_order                   /* End of specification for the bit field order */
```

Figure 5.4 Structure/Union Used in the Sample Code

5.6 Variables

Table 5.6 lists the static Variables.

Table 5.6 static Variables

Type	Variable Name	Contents	Function Used
static uint8_t	tx_buf_0[]	Transmit buffer for slave 0	main
static uint8_t	rx_buf_0[BUF_SIZE]	Receive buffer for slave 0	main
static uint8_t	tx_buf_1[]	Transmit buffer for slave 1	main
static uint8_t	rx_buf_1[BUF_SIZE]	Receive buffer for slave 1	main
static const uint8_t *	pbuf_tx	Pointer to the transmit buffer	RSPI_PreTrans rspi_spti_isr
static uint8_t	tx_cnt	Transmit counter	
static uint8_t *	pbuf_rx	Pointer to the receive buffer	RSPI_PreTrans rspi_spri_isr
static uint8_t	rx_cnt	Receive counter	
static rspi_state_t	state	RSPI state	RSPI_PreTrans RSPI_GetState rspi_spii_isr rspi_spei_isr

5.7 Functions

Table 5.7 lists the Functions.

Table 5.7 Functions

Function Name	Outline
main	Main processing
port_init	Port initialization
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
peripheral_init	Peripheral function initialization
cb_rspi_slave0_end	Callback function (completion of RSPI transmission to/reception from slave 0)
cb_rspi_slave1_end	Callback function (completion of RSPI transmission to/reception from slave 1)
cb_rspi_rx_error	Callback function (RSPI transmit/receive error)
RSPI_Init	User interface function (RSPI initialization)
RSPI_PreTrans	User interface function (RSPI transmit/receive start)
RSPI_GetState	User interface function (obtain RSPI state)
rspi_spti_isr	RSPI transmit interrupt
rspi_spii_isr	RSPI idle interrupt
rspi_spri_isr	RSPI receive interrupt
rspi_spei_isr	RSPI error interrupt
Excep_RSPI0_SPEI0	RSPI0_SPEI0 interrupt handling
Excep_RSPI0_SPRI0	RSPI0_SPRI0 interrupt handling
Excep_RSPI0_SPTI0	RSPI0_SPTI0 interrupt handling
Excep_RSPI0_SPII0	RSPI0_SPII0 interrupt handling

5.8 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	After initialization, starts RSPI transmission to and reception from slave 0. When the transmission and reception have been completed, starts RSPI transmission to and reception from slave 1.
Arguments	None
Return Value	None

port_init	
Outline	Port initialization
Header	None
Declaration	static void port_init(void)
Description	Initializes the ports.
Arguments	None
Return Value	None

R_INIT_StopModule	
Outline	Stop processing for active peripheral functions after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	Configures the setting to enter the module stop state.
Arguments	None
Return Value	None
Remarks	Transition to the module stop state is not performed in the sample code. Refer to the RX220 Group Initial Setting Rev. 1.00 application note for details on this function.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Initializes port direction registers for ports that do not exist in products with less than 100 pins.
Arguments	None
Return Value	None
Remarks	The number of pins in the sample code is set for the 100-pin package (PIN_SIZE=100). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. Refer to the RX220 Group Initial Setting Rev. 1.00 application note for details on this function.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Initializes the clock.
Arguments	None
Return Value	None
Remarks	The sample code selects processing which uses PLL as the system clock without using the sub-clock. Refer to the RX220 Group Initial Setting Rev. 1.00 application note for details on this function.

peripheral_init	
Outline	Peripheral function initialization
Header	None
Declaration	static void peripheral_init(void)
Description	Initializes peripheral functions used.
Arguments	None
Return Value	None

cb_rspi_slave0_end	
Outline	Callback function (completion of RSPI transmission to/reception from slave 0)
Header	None
Declaration	static void cb_rspi_slave0_end(void)
Description	This function is called when RSPI transmission and reception have been completed between the master and slave 0.
Arguments	None
Return Value	None

cb_rspi_slave1_end	
Outline	Callback function (completion of RSPI transmission to/reception from slave 1)
Header	None
Declaration	static void cb_rspi_slave1_end(void)
Description	This function is called when RSPI transmission and reception have been completed between the master and slave 1.
Arguments	None
Return Value	None

cb_rspi_rx_error	
Outline	Callback function (RSPI transmit/receive error)
Header	None
Declaration	static void cb_rspi_rx_error(void)
Description	This function is called when an RSPI transmit/receive error occurs.
Arguments	None
Return Value	None
Remarks	Error processing is not performed in the sample code. Add a program as required.

RSPI_Init	
Outline	User interface function (RSPI initialization)
Header	rspi.h
Declaration	void RSPI_Init(void)
Description	Initializes the RSPI.
Arguments	None
Return Value	None

RSPI_PreTrans	
Outline	User interface function (RSPI transmit/receive start)
Header	rspi.h
Declaration	uint8_t RSPI_PreTrans(uint16_t ssl, const uint8_t * pbuf_t, uint8_t * pbuf_r, uint8_t num, CallBackFunc pcb_end, CallBackFunc pcb_rx_error)
Description	Verifies that the RSPI is in idle state. Sets the SSL pin specified by the argument. Enables the RSPI function, RSPI transmit interrupt, RSPI receive interrupt, and RSPI error interrupt, then starts RSPI transmission and reception.
Arguments	uint16_t ssl: SSL pin selection const uint8_t * pbuf_t: Pointer to the transmit data store buffer uint8_t * pbuf_r: Pointer to the receive data store buffer uint8_t num: Number of bytes to be transmitted/received CallBackFunc pcb_end: Pointer to the callback function (completion of RSPI transmission/reception) CallBackFunc pcb_rx_error: Pointer to the callback function (RSPI transmit/receive error)
Return Value	RSPI_NG: Argument error (number of bytes to be transmitted/received is 0) RSPI_NOT_IDLE: RSPI transmission/reception being processed RSPI_OK: RSPI transmission/reception started

RSPI_GetState	
Outline	User interface function (obtain RSPI state)
Header	rspi.h
Declaration	rspi_state_t RSPI_GetState(void)
Description	Returns the RSPI state.
Arguments	None
Return Value	rspi_state_t.bit.b_rspi_busy: Transmit/receive busy flag 0: Transmission/reception ready 1: Transmission/reception busy rspi_state_t.bit.b_rx_orer: Overrun error flag 0: Overrun error not occurred 1: Overrun error occurred

rspi_spti_isr	
Outline	RSPI transmit interrupt
Header	None
Declaration	static void rspi_spti_isr(void)
Description	This function is called in the RSPI0.SPTI0 interrupt handling. Writes the transmit data to the SPDR register. After transmitting the last data, disables generating the RSPI transmit interrupt request and enables generating the RSPI idle interrupt request.
Arguments	None
Return Value	None

rspi_spii_isr	
Outline	RSPI idle interrupt
Header	None
Declaration	static void rspi_spii_isr(void)
Description	This function is called in the RSPI0.SPII0 interrupt handling. Disables the RSPI function. Calls the callback function (completion of RSPI transmission to/reception from slave 0) or callback function (completion of RSPI transmission to/reception from slave 1).
Arguments	None
Return Value	None

rspi_spri_isr	
Outline	RSPI receive interrupt
Header	None
Declaration	static void rspi_spri_isr(void)
Description	This function is called in the RSPI0.SPRI0 interrupt handling. Reads the receive data from the SPDR register. After receiving the last data, disables generating the RSPI receive interrupt request.
Arguments	None
Return Value	None

rspi_spei_isr	
Outline	RSPI error interrupt
Header	None
Declaration	static void rspi_spei_isr(void)
Description	This function is called in the RSPI0.SPEI0 interrupt handling. Disables the RSPI function and calls the callback function (RSPI transmit/receive error).
Arguments	None
Return Value	None

Excep_RSPI0_SPEI0	
Outline	RSPI0.SPEI0 interrupt handling
Header	None
Declaration	static void Excep_RSPI0_SPEI0(void)
Description	Performs processing for the RSPI error interrupt.
Arguments	None
Return Value	None

Excep_RSPI0_SPRI0	
Outline	RSPI0.SPRI0 interrupt handling
Header	None
Declaration	static void Excep_RSPI0_SPRI0(void)
Description	Performs processing for the RSPI receive interrupt.
Arguments	None
Return Value	None

Excep_RSPI0_SPTI0	
Outline	RSPI0.SPTI0 interrupt handling
Header	None
Declaration	static void Excep_RSPI0_SPTI0(void)
Description	Performs processing for the RSPI transmit interrupt.
Arguments	None
Return Value	None

Excep_RSPI0_SPII0	
Outline	RSPI0.SPII0 interrupt handling
Header	None
Declaration	static void Excep_RSPI0_SPII0(void)
Description	Performs processing for the RSPI idle interrupt.
Arguments	None
Return Value	None

5.9 Flowcharts

5.9.1 Main Processing

Figure 5.5 shows the Main Processing.

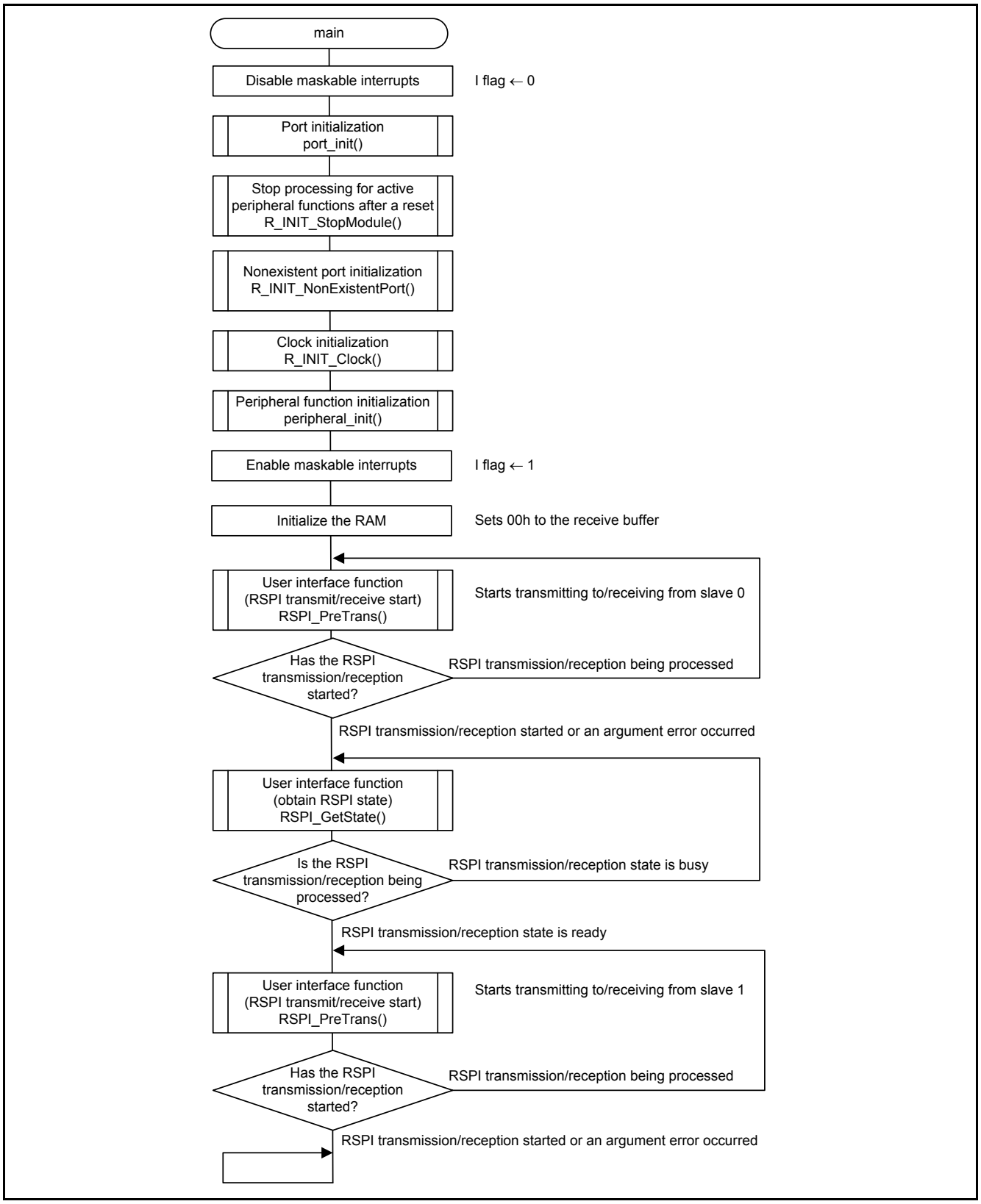


Figure 5.5 Main Processing

5.9.2 Port Initialization

Figure 5.6 shows the Port Initialization.

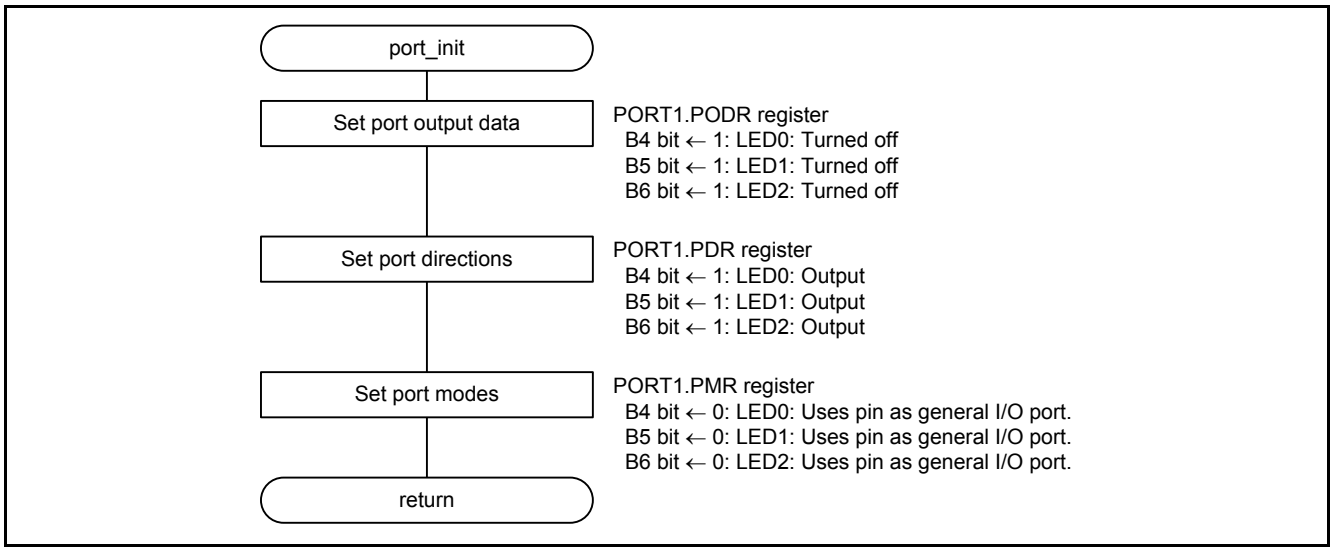


Figure 5.6 Port Initialization

5.9.3 Peripheral Function Initialization

Figure 5.7 shows the Peripheral Function Initialization.

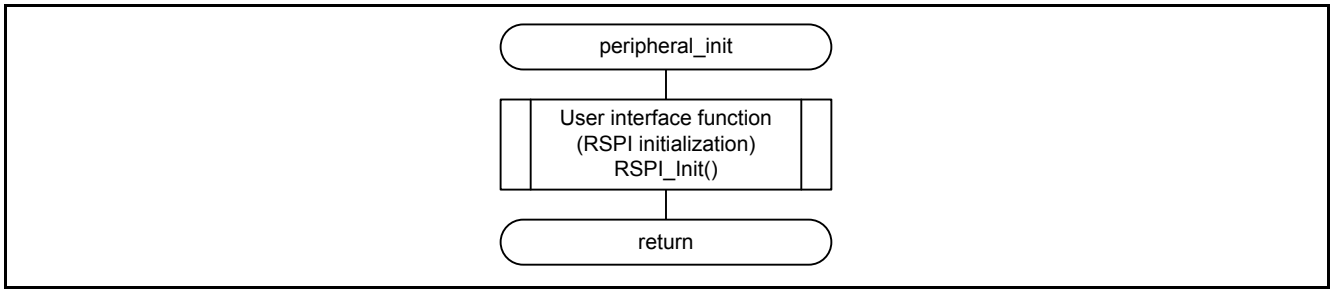


Figure 5.7 Peripheral Function Initialization

5.9.4 Callback Function (Completion of RSPI Transmission to/Reception from Slave 0)

Figure 5.8 shows the Callback Function (Completion of RSPI Transmission to/Reception from Slave 0).

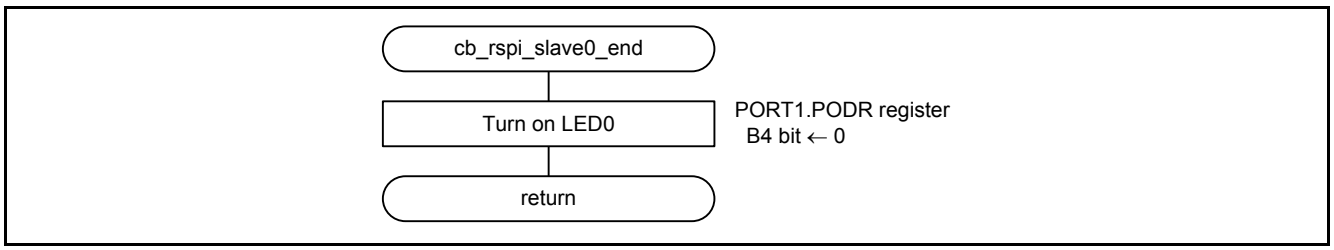


Figure 5.8 Callback Function (Completion of RSPI Transmission to/Reception from Slave 0)

5.9.5 Callback Function (Completion of RSPI Transmission to/Reception from Slave 1)

Figure 5.9 shows the Callback Function (Completion of RSPI Transmission to/Reception from Slave 1).

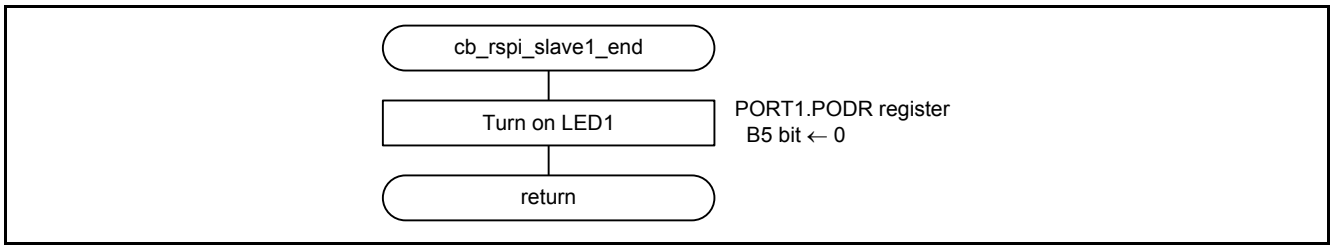


Figure 5.9 Callback Function (Completion of RSPI Transmission to/Reception from Slave 1)

5.9.6 Callback Function (RSPI Transmit/Receive Error)

Figure 5.10 shows the Callback Function (RSPI Transmit/Receive Error).

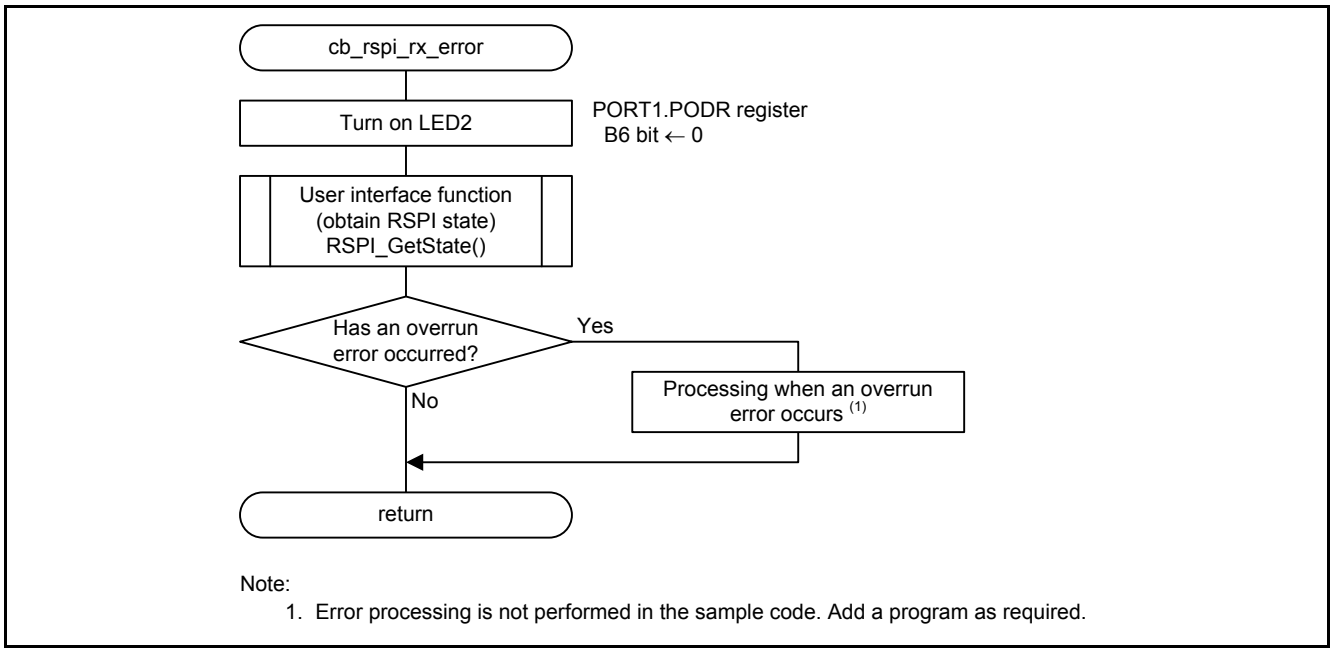


Figure 5.10 Callback Function (RSPI Transmit/Receive Error)

5.9.7 User Interface Function (RSPI Initialization)

Figure 5.11 and Figure 5.12 show the User Interface Function (RSPI Initialization).

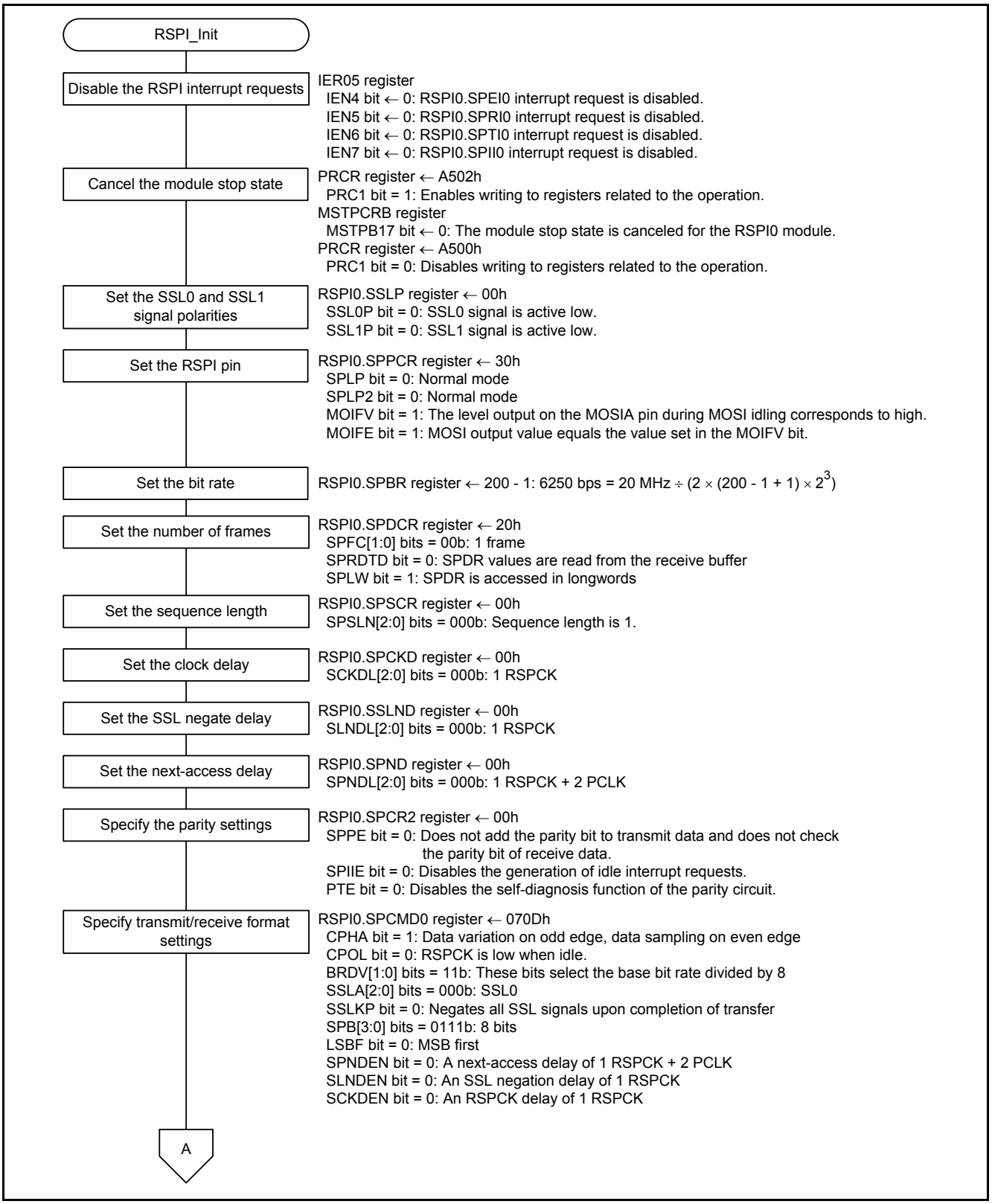


Figure 5.11 User Interface Function (RSPI Initialization) (1/2)

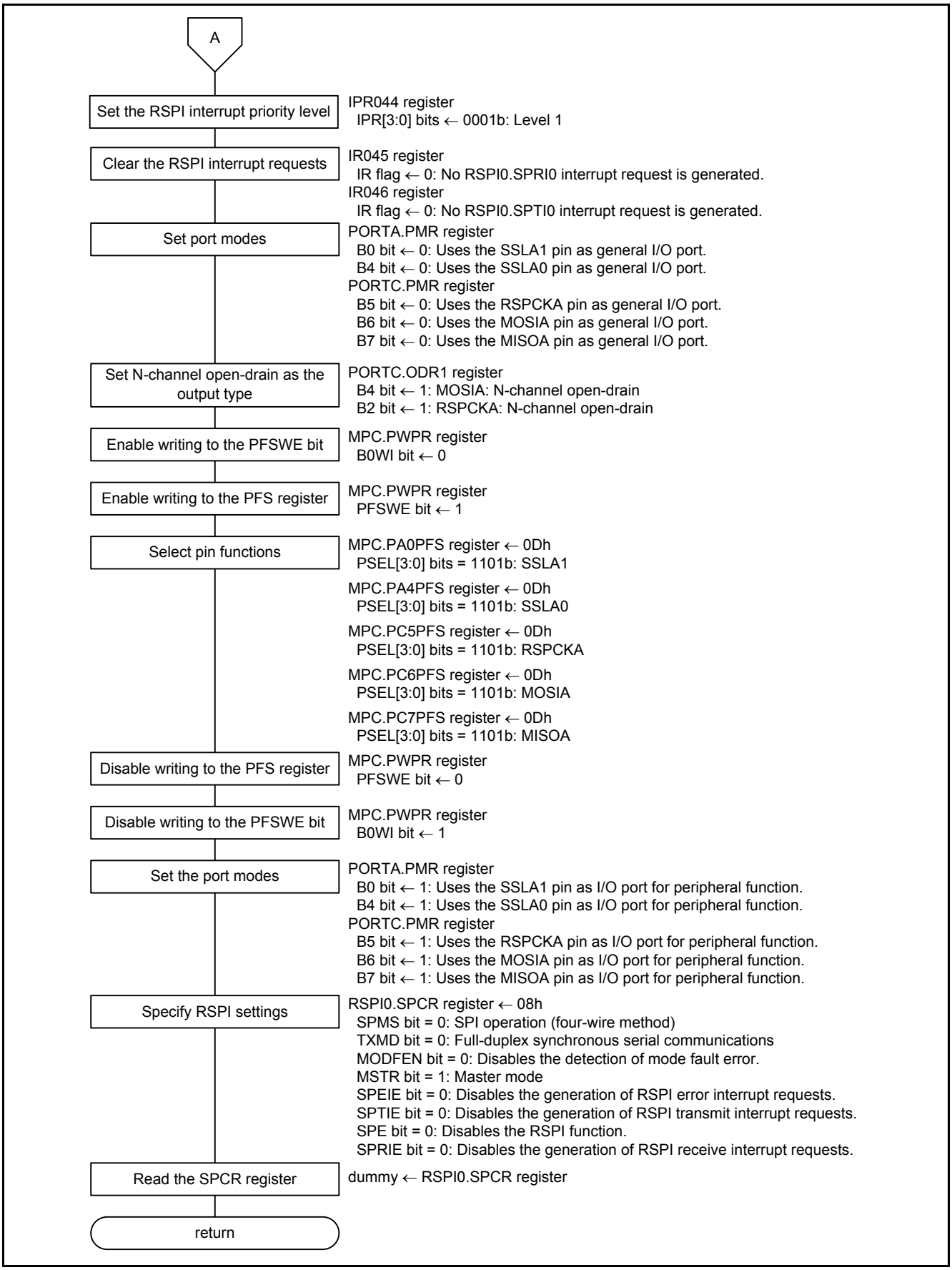


Figure 5.12 User Interface Function (RSPI Initialization) (2/2)

5.9.8 User Interface Function (RSPI Transmit/Receive Start)

Figure 5.13 and Figure 5.14 show the RSPI Transmit/Receive Start.

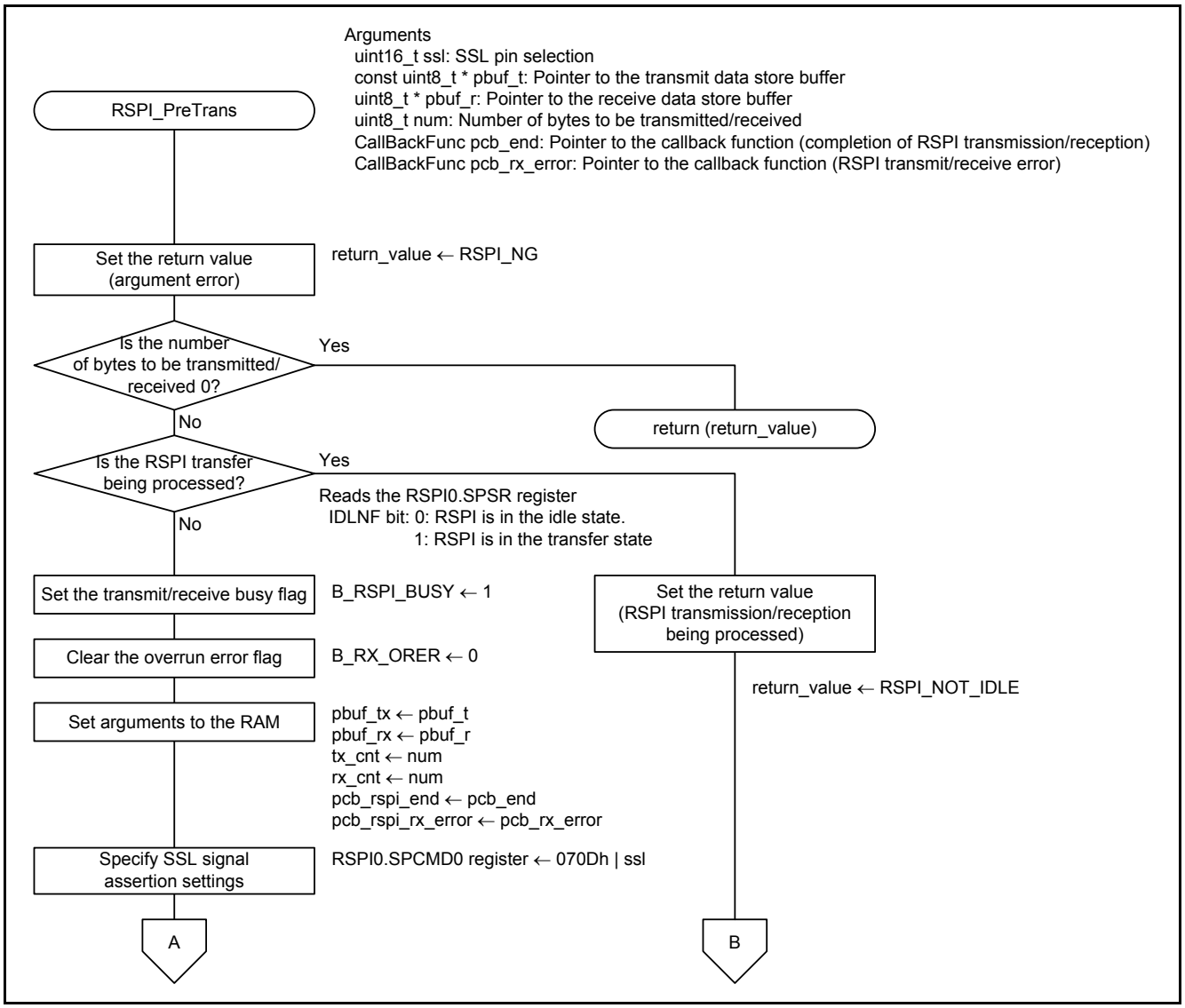
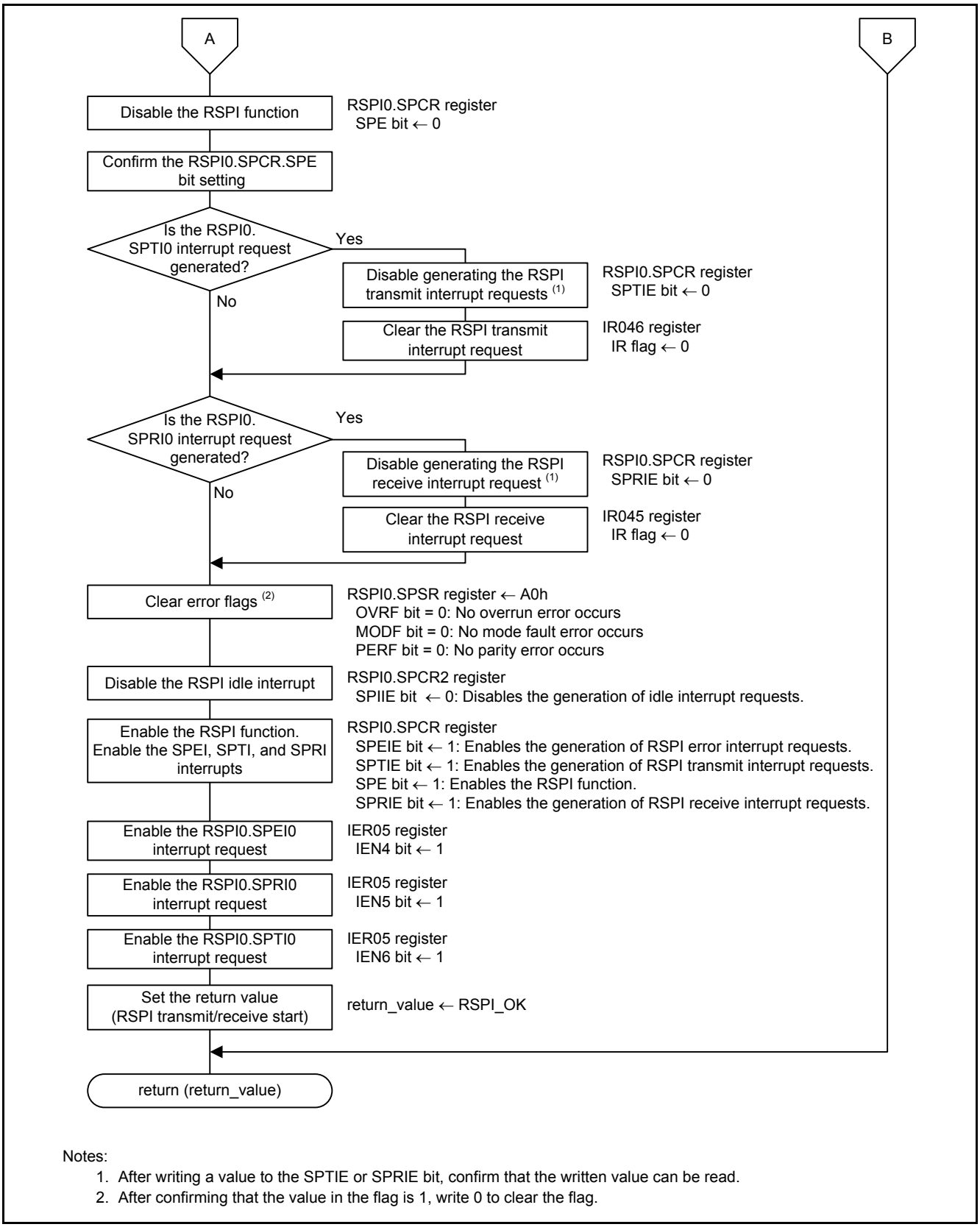


Figure 5.13 User Interface Function (RSPI Transmit/Receive Start) (1/2)



Notes:
1. After writing a value to the SPTIE or SPRIE bit, confirm that the written value can be read.
2. After confirming that the value in the flag is 1, write 0 to clear the flag.

Figure 5.14 User Interface Function (RSPI Transmit/Receive Start) (2/2)

5.9.9 User Interface Function (Obtain RSPI State)

Figure 5.15 shows the User Interface Function (Obtain RSPI State).

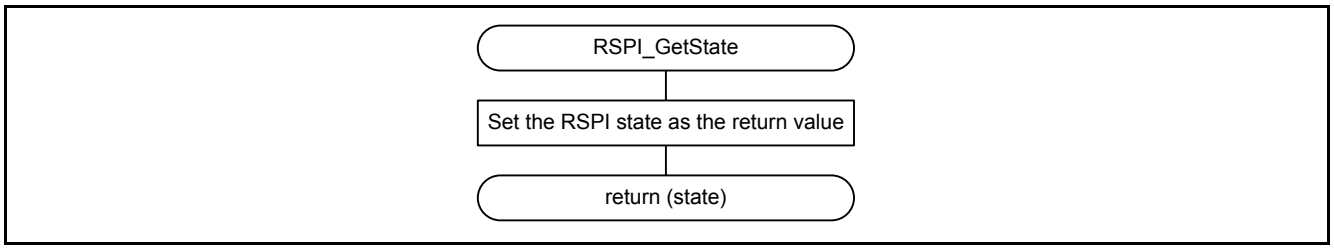
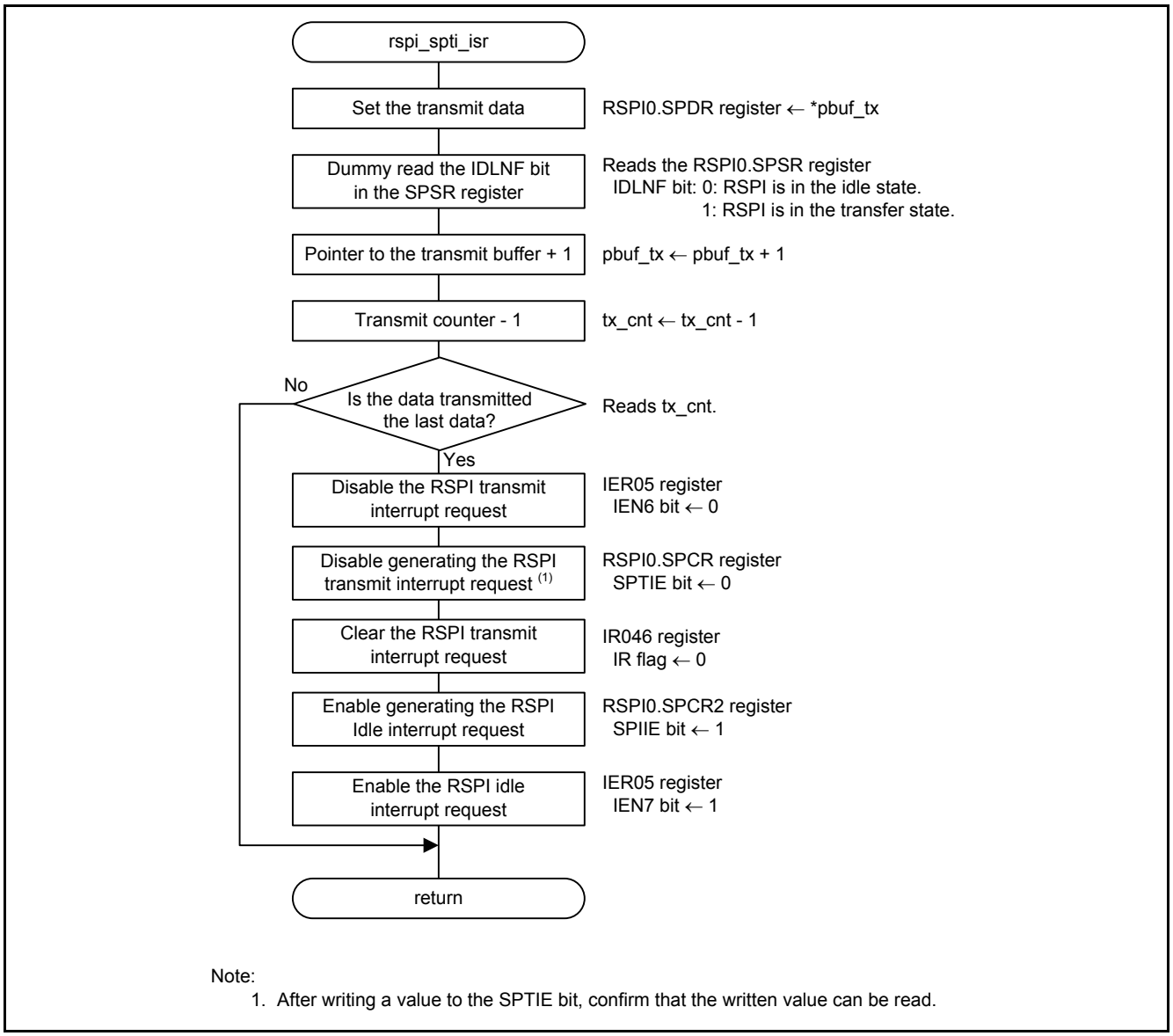


Figure 5.15 User Interface Function (Obtain RSPI State)

5.9.10 RSPI Transmit Interrupt

Figure 5.16 shows the RSPI Transmit Interrupt.



Note:
1. After writing a value to the SPTIE bit, confirm that the written value can be read.

Figure 5.16 RSPI Transmit Interrupt

5.9.11 RSPI Idle Interrupt

Figure 5.17 shows the RSPI Idle Interrupt.

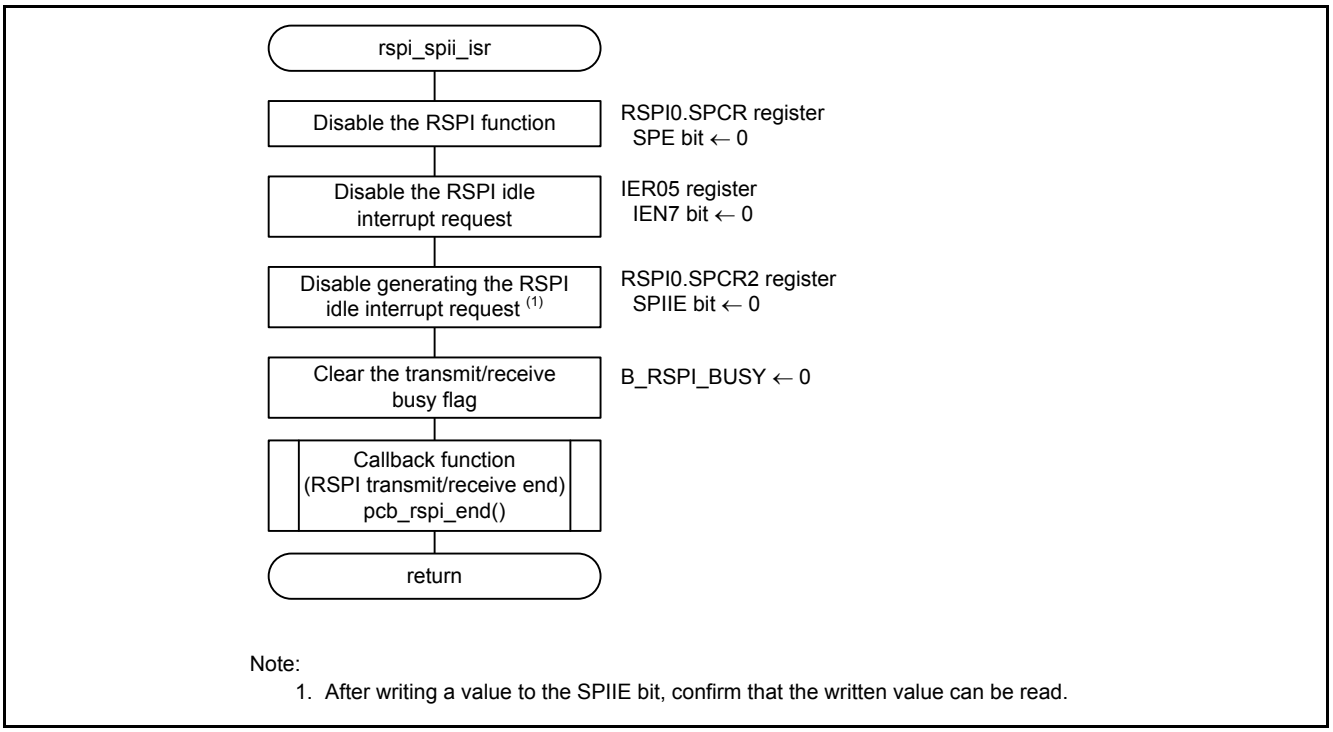
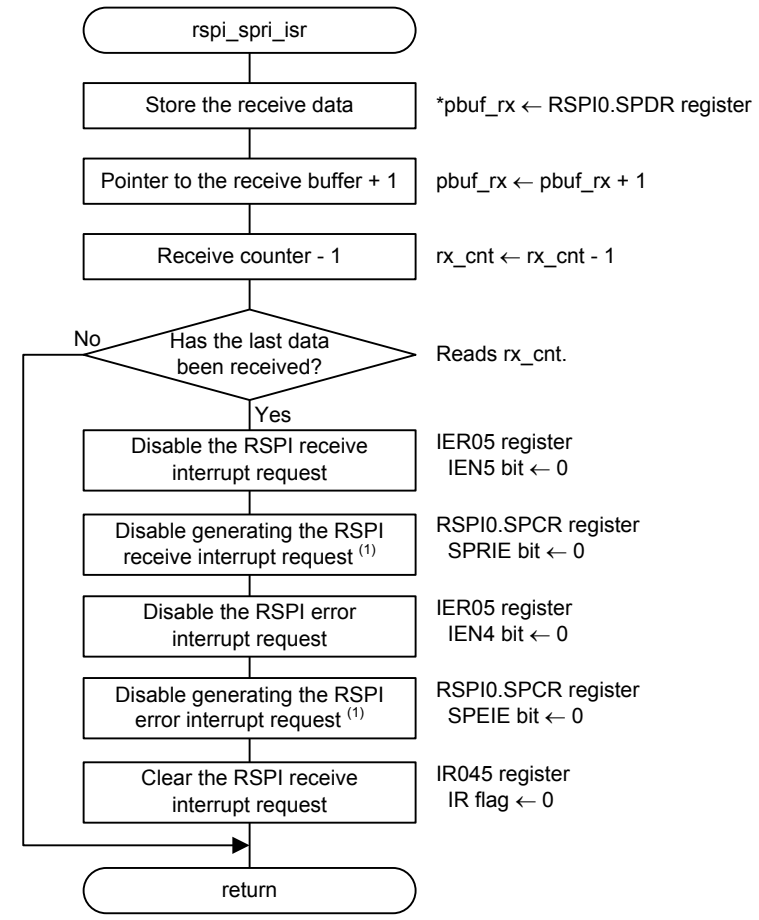


Figure 5.17 RSPI Idle Interrupt

5.9.12 RSPI Receive Interrupt

Figure 5.18 shows the RSPI Receive Interrupt.

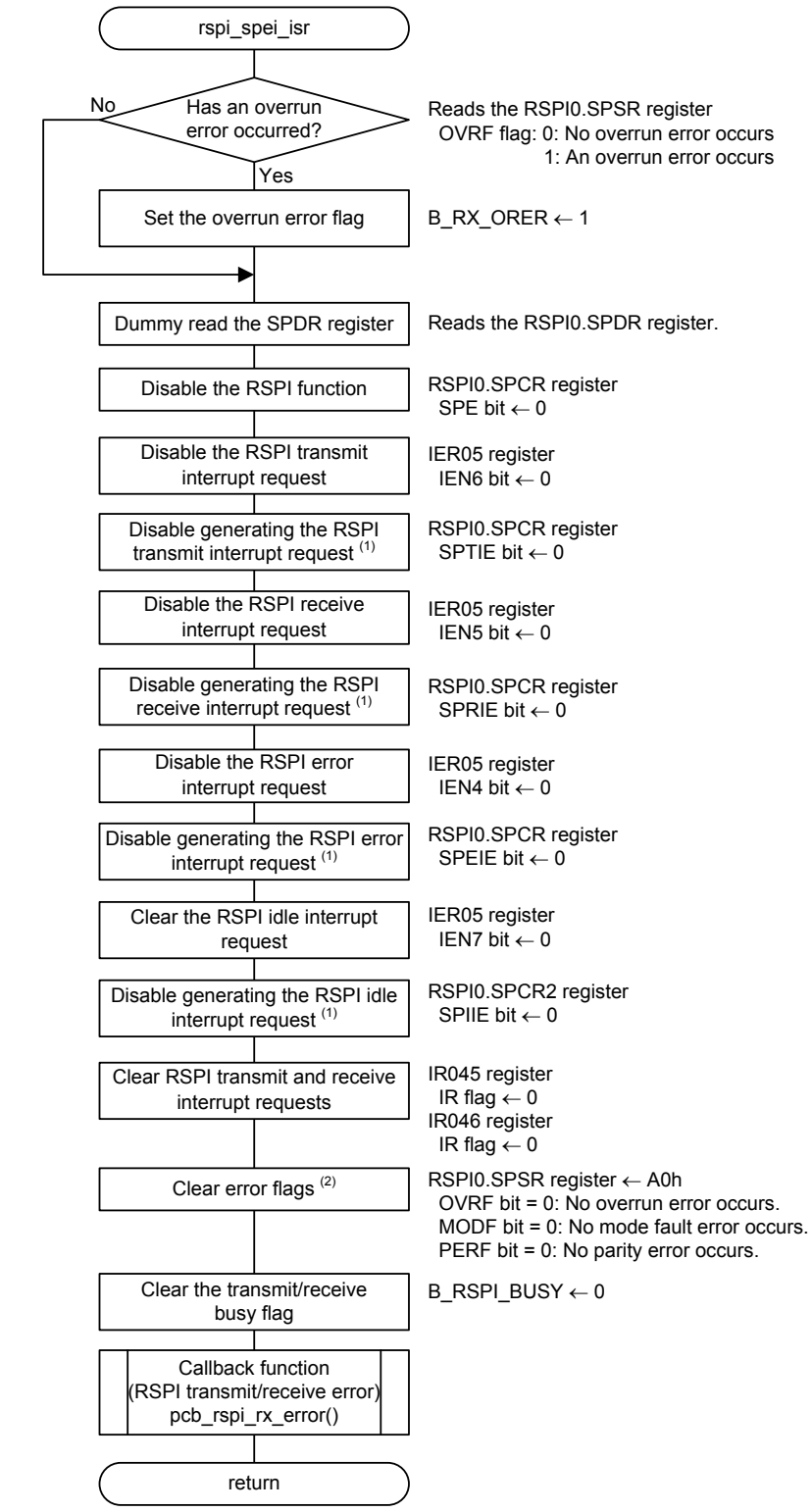


Note:
1. After writing a value to the SPRIE bit or SPEIE bit, confirm that the written value can be read.

Figure 5.18 RSPI Receive Interrupt

5.9.13 RSPI Error Interrupt

Figure 5.19 shows the RSPI Error Interrupt.



Notes:
1. After writing a value to the SPTIE, SPRIE, SPEIE, or SPIIE bit, confirm that the written value can be read.
2. After confirming the value in the flag is 1, write 0 to clear the flag.

Figure 5.19 RSPI Error Interrupt

5.9.14 RSPI0.SPEI0 Interrupt Handling

Figure 5.20 shows the RSPI0.SPEI0 Interrupt Handling.

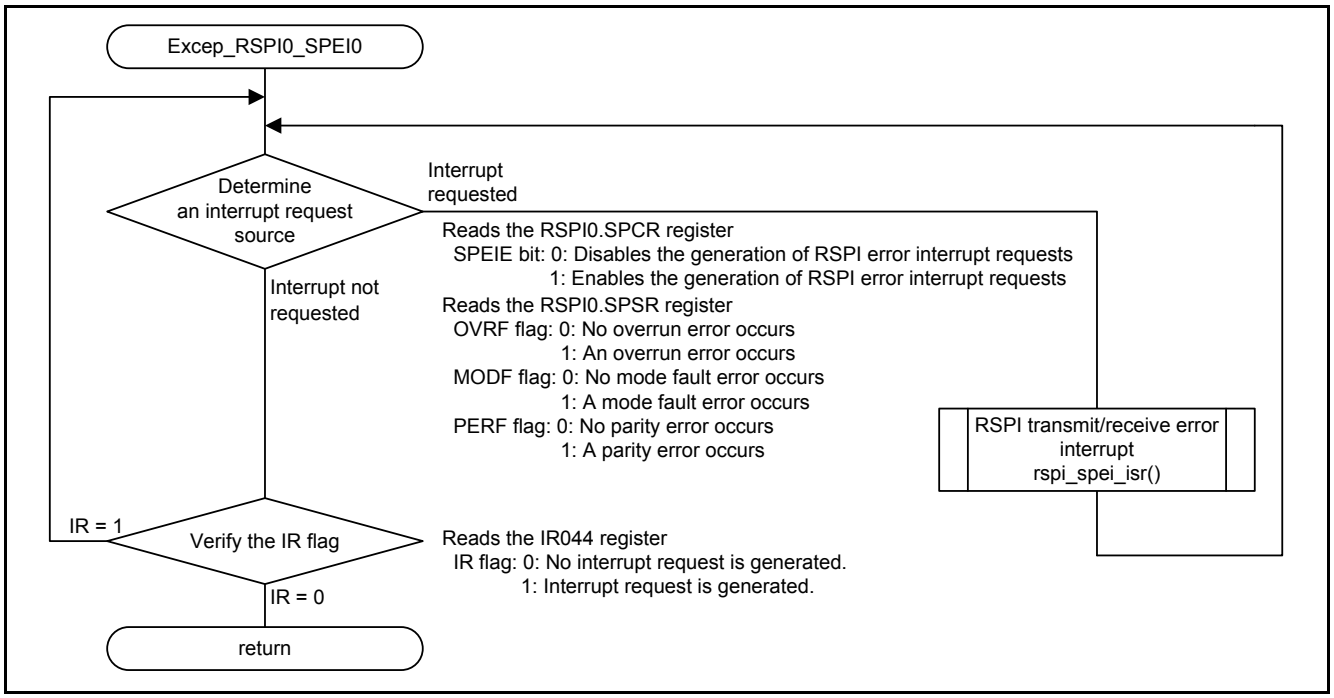


Figure 5.20 RSPI0.SPEI0 Interrupt Handling

5.9.15 RSPI0.SPRI0 Interrupt Handling

Figure 5.21 shows the RSPI0.SPRI0 Interrupt Handling.

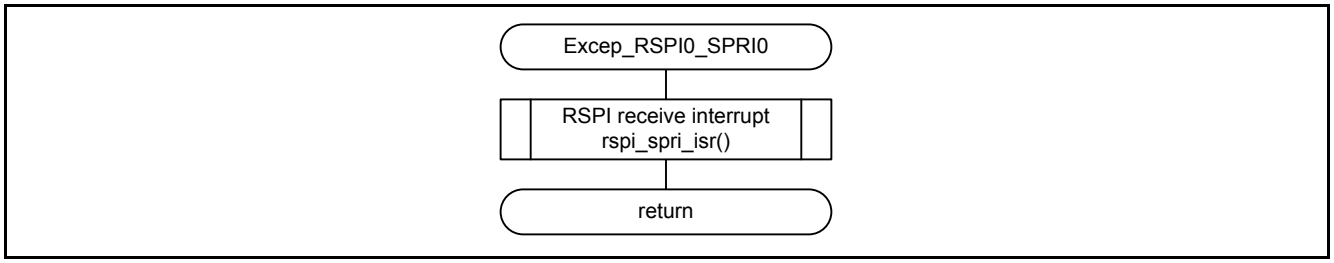


Figure 5.21 RSPI0.SPRI0 Interrupt Handling

5.9.16 RSPI0.SPTI0 Interrupt Handling

Figure 5.22 shows the RSPI0.SPTI0 Interrupt Handling.

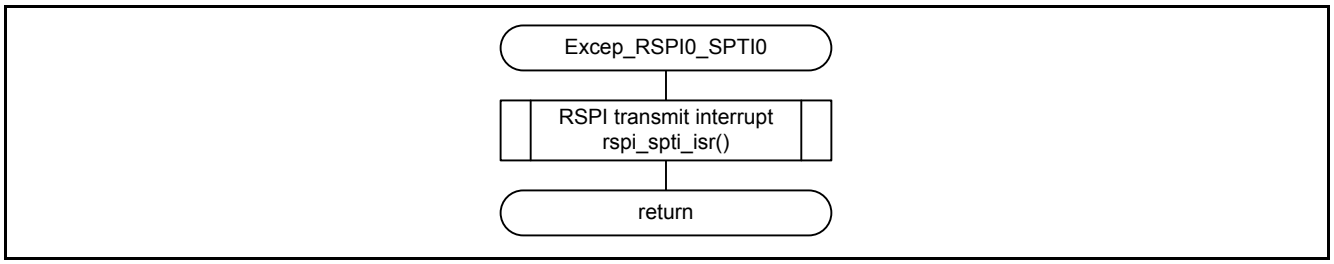


Figure 5.22 RSPI0.SPTI0 Interrupt Handling

5.9.17 RSPI0.SPII0 Interrupt Handling

Figure 5.23 shows the RSPI0.SPII0 Interrupt Handling.

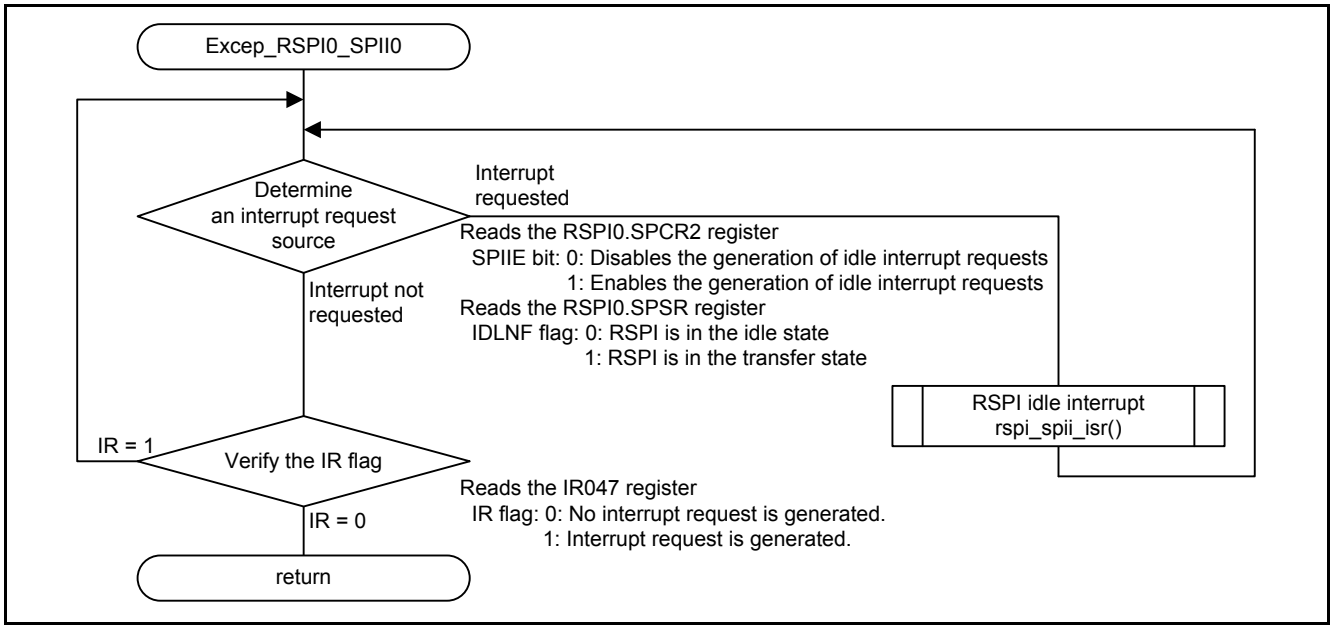


Figure 5.23 RSPI0.SPII0 Interrupt Handling

6. Hardware (Slave)

6.1 Pins Used

Table 6.1 lists the Pins Used and Their Functions.

The number of pins in the sample code is set for the 100-pin package. When using products with less than 100 pins, select pins appropriate to the product used.

Table 6.1 Pins Used and Their Functions

Pin Name	I/O	Function
P15	Output	Outputs a signal for LED1 (completion of RSPI transmission to/reception from the master)
P16	Output	Outputs a signal for LED2 (RSPI transmit/receive error)
PA4/SSLA0	Input	Inputs a signal for slave selection
PC5/RSPCKA	Input	Clock input pin
PC6/MOSIA	Input	Data input pin
PC7/MISOA	Output	Data output pin

7. Software (Slave)

After a reset, the user interface function (RSPI initialization) is called to initialize the RSPI.

After the initialization, the user interface function (RSPI transmit/receive start) is called and transmission and reception are enabled. When the 3-byte transmission and reception have been completed, RSPI transmission and reception are disabled, and the callback function (completion of RSPI transmission to/reception from the master) is called. LED1 is turned on with the callback function.

If a transmit/receive error occurs, RSPI transmission and reception are disabled, the callback function (RSPI transmit/receive error) is called, and LED2 is turned on.

Settings for the peripheral function are as follows and Figure 7.1 shows the Software Configuration.

RSPI

- RSPI mode: SPI operation (four-wire method)
- Communication mode: Full-duplex synchronous serial communications
- Clock source: PCLKB (20 MHz)
- Transfer bit length: 8 bits
- Parity: None
- Sequence length: 1 sequence
- Number of frames: 1 frame
- Error detection: Overrun error
  Mode fault error
- Interrupt source: RSPI error interrupt (SPEI) enabled
  RSPI receive interrupt (SPRI) enabled
  RSPI transmit interrupt (SPTI) enabled
  RSPI idle interrupt (SPII) disabled

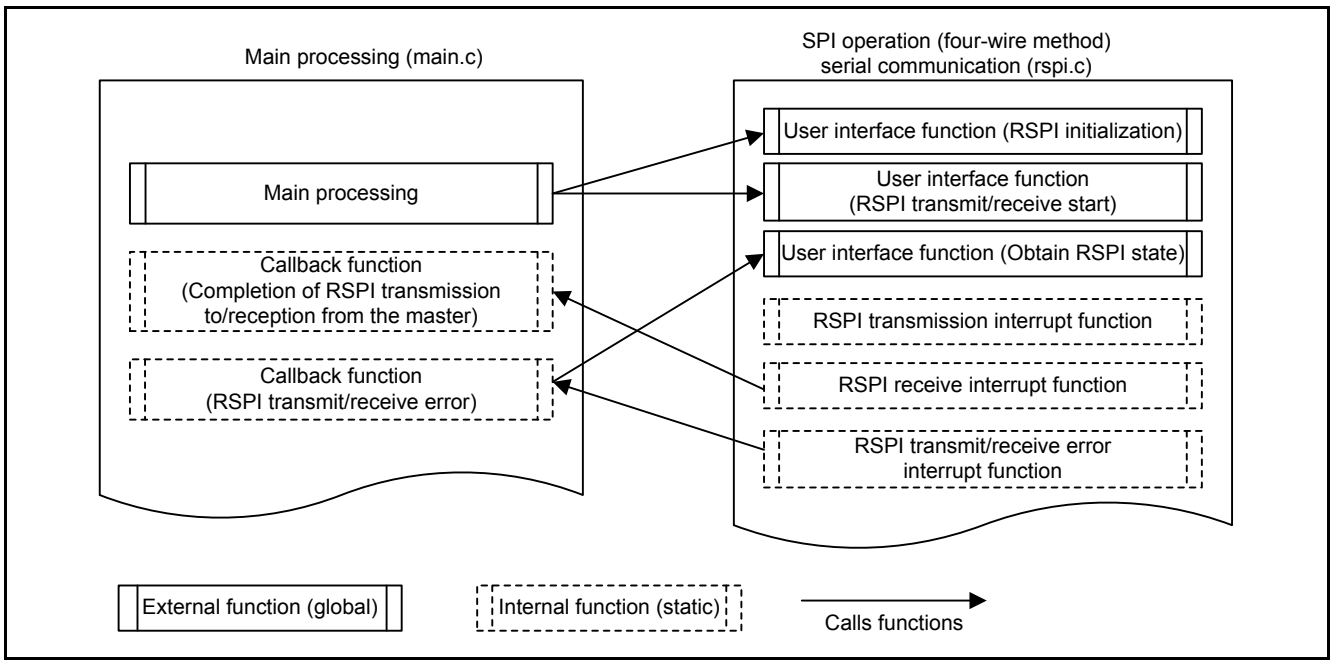


Figure 7.1 Software Configuration

7.1 Operation Overview

Figure 7.2 shows the Timing of Serial Communication with SPI Operation (Four-Wire Method) and (1) to (6) in the figures correspond to numbers in the operation descriptions below.

(1) Initialization
Initializes the RSPI using the user interface function (RSPI initialization).

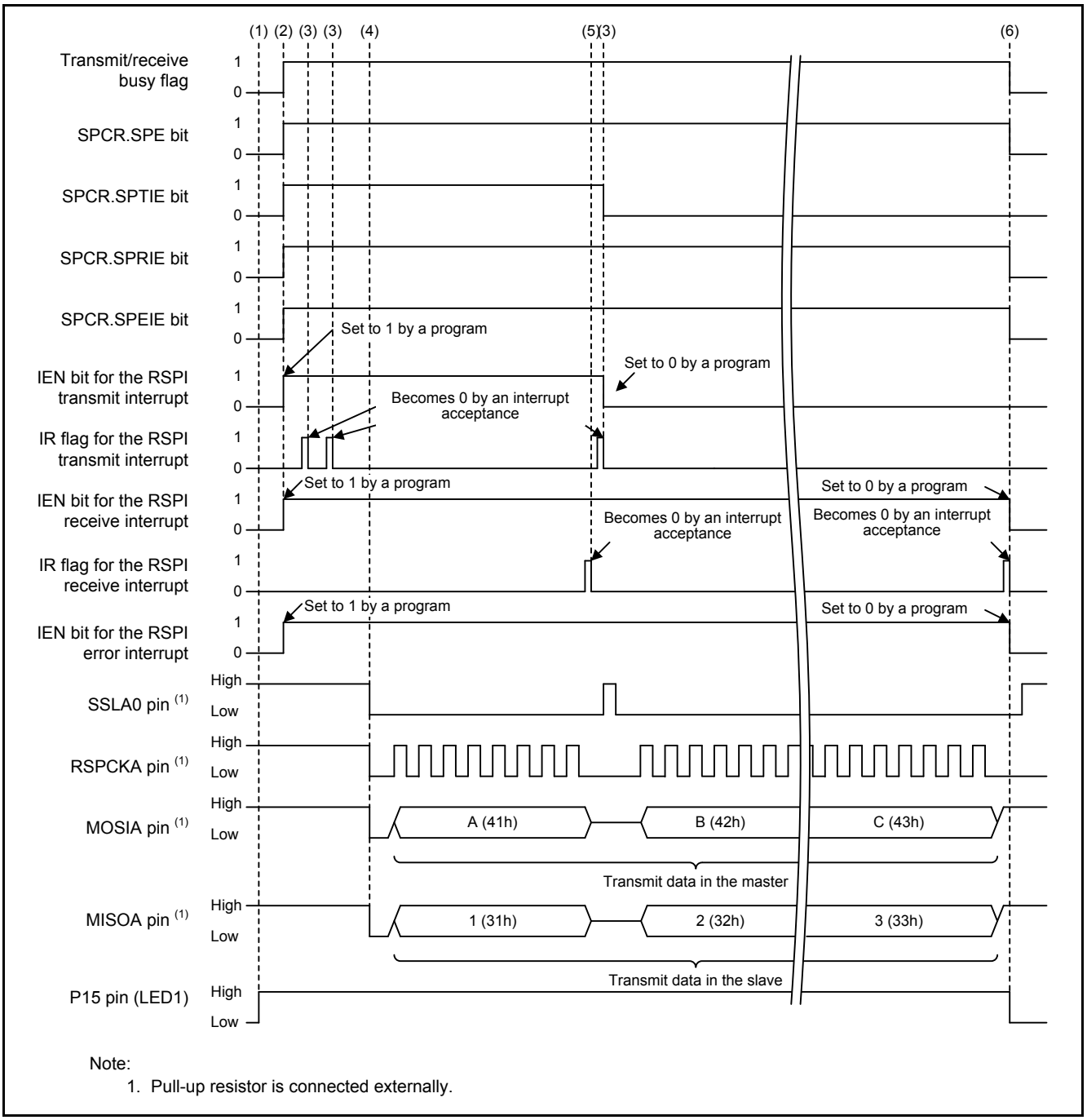
(2) Starting transmission to/reception from the master
With the user interface function (RSPI transmit/receive start), verifies the SPSR.IDLNF bit. When the bit is 1 (RSPI is in the transfer state), returns RSPI_BUSY (RSPI transmission/reception being processed). When the bit is 0 (RSPI is in the idle state), sets the transmit/receive busy flag to 1. Sets the SPCR.SPEIE bit to 1 (enables the generation of RSPI error interrupt requests), the SPCR.SPTIE bit to 1 (enables the generation of RSPI transmit interrupt requests), the SPCR.SPE bit to 1 (enables the RSPI function), and SPCR.SPRIE bit to 1 (enables the generation of RSPI receive interrupt requests). Sets the IEN bits for the RSPI error interrupt, RSPI receive interrupt, and RSPI transmit interrupt to 1. Then waits for an input on the SSLA0 pin and an edge input on the RSPCKA pin from the master.

(3) Setting transmit data
In the RSPI transmit interrupt handling, writes the value in the transmit buffer to the SPDR register. When the last data has been written, sets the SPTIE bit to 0 (disables the generation of RSPI transmit interrupt requests).

(4) Transmitting to and receiving from the master
When an input on the SSLA0 pin and an edge input on the RSPCKA pin from the master are confirmed, performs data transmission and reception.

(5) Receiving data from the master
In the RSPI receive interrupt handling, writes the value in the SPDR register to the receive buffer.

(6) Completing the transmission to/reception from the master
When the reception for the last data has been completed, sets the SPE bit to 0 (disables the RSPI function), the SPRIE bit to 0 (disables the generation of RSPI receive interrupt requests), the SPEIE bit to 0 (disables the generation of RSPI error interrupt requests), the transmit/receive busy flag to 0, and calls the callback function (completion of RSPI transmission to/reception from the master). Then LED1 is turned on.



Note:
1. Pull-up resistor is connected externally.

Figure 7.2 Timing of Serial Communication with SPI Operation (Four-Wire Method)

7.2 File Composition

Table 7.1 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 7.1 Files Used in the Sample Code

File Name	Outline	Remarks
main.c	Main processing	
r_init_stop_module.c	Stop processing for active peripheral functions after a reset	
r_init_stop_module.h	Header file for r_init_stop_module.c	
r_init_non_existent_port.c	Nonexistent port initialization	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Clock initialization	
r_init_clock.h	Header file for r_init_clock.c	
rspi.c	Serial communication with SPI operation (four-wire method)	
rspi.h	Header file for rspi.c	

7.3 Option-Setting Memory

Table 7.2 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 7.2 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	The IWDT is stopped after a reset.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

7.4 Constants

Table 7.3 to Table 7.5 list the Constants Used in the Sample Code.

Table 7.3 Constants Used in the Sample Code (main.c)

Constant Name	Setting Value	Contents
LED1_REG_PODR	PORT1.PODR.BIT.B5	LED1 output data store bit
LED1_REG_PDR	PORT1.PDR.BIT.B5	LED1 I/O select bit
LED1_REG_PMR	PORT1.PMR.BIT.B5	LED1 pin mode control bit
LED2_REG_PODR	PORT1.PODR.BIT.B6	LED2 output data store bit
LED2_REG_PDR	PORT1.PDR.BIT.B6	LED2 I/O select bit
LED2_REG_PMR	PORT1.PMR.BIT.B6	LED2 pin mode control bit
LED_ON	0	LED output data: Turned on
LED_OFF	1	LED output data: Turned off
TR_SIZE	3	Transmission/reception size
BUF_SIZE	TR_SIZE	Buffer size

Table 7.4 Constants Used in the Sample Code (rspi.c)

Constant Name	Setting Value	Contents
SPSR_ERROR_FLAGS	0Dh	Bit pattern of an error flag in the RSPI.SPSR register
B_RSPI_BUSY	state.bit.b_rspi_busy	Transmit/receive busy flag 0: Transmission/reception ready 1: Transmission/reception busy
B_RX_ORER	state.bit.b_rx_orer	Overrun error flag 0: Overrun error not occurred 1: Overrun error occurred
B_RX_MODF	state.bit.b_rx_modf	Mode fault error flag 0: Mode fault error not occurred 1: Mode fault error occurred

Table 7.5 Constants Used in the Sample Code (rspi.h)

Constant Name	Setting Value	Contents
RSPI_OK	00h	Return value of the RSPI_PreTrans function: RSPI transmit/receive start
RSPI_NOT_IDLE	01h	Return value of the RSPI_PreTrans function: RSPI transmission/reception being processed
RSPI_NG	02h	Return value of the RSPI_PreTrans function: Argument error

7.5 Structure/Union List

Figure 7.3 shows the Structure/Union Used in the Sample Code.

```
#pragma bit_order left          /* Bit field order: The bit field members are allocated from upper bits */
#pragma unpack                  /* The boundary alignment value for structure members: Alignment by member type */
typedef   union
{
    uint8_t    byte;
    struct
    {
        uint8_t    b_rspi_busy :1;    /* Transmit/receive busy flag   0: Transmission/reception ready   1: Transmission/reception busy */
        uint8_t    b_rx_orer   :1;    /* Overrun error flag           0: Overrun error not occurred     1: Overrun error occurred */
        uint8_t    b_rx_modf   :1;    /* Mode fault error flag        0: Mode fault error not occurred  1: Mode fault error occurred */
        uint8_t                :5;    /* Not used */
    } bit;
} rspi_state_t;
#pragma packoption              /* End of specification for the boundary alignment value for structure members */
#pragma bit_order               /* End of specification for the bit field order */
```

Figure 7.3 Structure/Union Used in the Sample Code

7.6 Variables

Table 7.6 lists the static Variables.

Table 7.6 static Variables

Type	Variable Name	Contents	Function Used
static uint8_t	tx_buf[]	Transmit buffer	main
static uint8_t	rx_buf[BUF_SIZE]	Receive buffer	main
static const uint8_t *	pbuf_tx	Pointer to the transmit buffer	RSPI_PreTrans
static uint8_t	tx_cnt	Transmit counter	rspi_spti_isr
static uint8_t *	pbuf_rx	Pointer to the receive buffer	RSPI_PreTrans
static uint8_t	rx_cnt	Receive counter	rspi_spri_isr
static rspi_state_t	state	RSPI state	RSPI_PreTrans RSPI_GetState rspi_spri_isr rspi_spei_isr

7.7 Functions

Table 7.7 lists the Functions.

Table 7.7 Functions

Function Name	Outline
main	Main processing
port_init	Port initialization
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
peripheral_init	Peripheral function initialization
cb_rspi_end	Callback function (completion of RSPI transmission to/reception from the master)
cb_rspi_rx_error	Callback function (RSPI transmit/receive error)
RSPI_Init	User interface function (RSPI initialization)
RSPI_PreTrans	User interface function (RSPI transmit/receive start)
RSPI_GetState	User interface function (obtain RSPI state)
rspi_spti_isr	RSPI transmit interrupt
rspi_spri_isr	RSPI receive interrupt
rspi_spei_isr	RSPI error interrupt
Excep_RSPI0_SPEI0	RSPI0_SPEI0 interrupt handling
Excep_RSPI0_SPRI0	RSPI0_SPRI0 interrupt handling
Excep_RSPI0_SPTI0	RSPI0_SPTI0 interrupt handling

7.8 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	After initialization, waits for an input signal on the SSLA0 pin from the master. When a signal is input through the SSLA0 pin and an edge input is detected on the RSPCKA pin, starts RSPI transmission and reception.
Arguments	None
Return Value	None

port_init	
Outline	Port initialization
Header	None
Declaration	static void port_init(void)
Description	Initializes the ports.
Arguments	None
Return Value	None

R_INIT_StopModule	
Outline	Stop processing for active peripheral functions after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	Configures the setting to enter the module stop state.
Arguments	None
Return Value	None
Remarks	Transition to the module stop state is not performed in the sample code. Refer to the RX220 Group Initial Setting Rev. 1.00 application note for details on this function.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Initializes port direction registers for ports that do not exist in products with less than 100 pins.
Arguments	None
Return Value	None
Remarks	The number of pins in the sample code is set for the 100-pin package (PIN_SIZE=100). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. Refer to the RX220 Group Initial Setting Rev. 1.00 application note for details on this function.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Initializes the clock.
Arguments	None
Return Value	None
Remarks	The sample code selects processing which uses PLL as the system clock without using the sub-clock. Refer to the RX220 Group Initial Setting Rev. 1.00 application note for details on this function.

peripheral_init	
Outline	Peripheral function initialization
Header	None
Declaration	static void peripheral_init(void)
Description	Initializes peripheral functions used.
Arguments	None
Return Value	None

cb_rspi_end	
Outline	Callback function (completion of RSPI transmission to/reception from the master)
Header	None
Declaration	static void cb_rspi_end(void)
Description	This function is called when an RSPI transmission and reception have been completed between the slave and master.
Arguments	None
Return Value	None

cb_rspi_rx_error	
Outline	Callback function (RSPI transmit/receive error)
Header	None
Declaration	static void cb_rspi_rx_error(void)
Description	This function is called when an RSPI transmit/receive error occurs.
Arguments	None
Return Value	None
Remarks	Error processing is not performed in the sample code. Add a program as required.

RSPI_Init	
Outline	User interface function (RSPI initialization)
Header	rspi.h
Declaration	void RSPI_Init(void)
Description	Initializes the RSPI.
Arguments	None
Return Value	None

RSPI_PreTrans	
Outline	User interface function (RSPI transmit/receive start)
Header	rspi.h
Declaration	uint8_t RSPI_PreTrans(const uint8_t * pbuf_t, uint8_t * pbuf_r, uint8_t num, CallBackFunc pcb_end, CallBackFunc pcb_rx_error)
Description	Verifies that the RSPI is in idle state. Enables the RSPI functions, RSPI transmit interrupt, RSPI receive interrupt, and RSPI error interrupt, waits for an input on the SSLA0 pin and an edge input on the RSPCKA pin from the master.
Arguments	const uint8_t * pbuf_t: Pointer to the transmit data store buffer uint8_t * pbuf_r: Pointer to the receive data store buffer uint8_t num: Number of bytes to be transmitted/received CallBackFunc pcb_end: Pointer to the callback function (completion of RSPI transmission/reception) CallBackFunc pcb_rx_error: Pointer to the callback function (RSPI transmit/receive error)
Return Value	RSPI_NG: Argument error (number of bytes to be transmitted/received is 0) RSPI_NOT_IDLE: RSPI transmission/reception being processed RSPI_OK: RSPI transmission/reception started

RSPI_GetState	
Outline	User interface function (obtain RSPI state)
Header	rspi.h
Declaration	rspi_state_t RSPI_GetState(void)
Description	Returns the RSPI state.
Arguments	None
Return Value	rspi_state_t.bit.b_rspi_busy: Transmit/receive busy flag 0: Transmission/reception ready 1: Transmission/reception busy rspi_state_t.bit.b_rx_orer: Overrun error flag 0: Overrun error not occurred 1: Overrun error occurred rspi_state_t.bit.b_rx_modf: Mode fault error flag 0: Mode fault error not occurred 1: Mode fault error occurred

rspi_spti_isr	
Outline	RSPI transmit interrupt
Header	None
Declaration	static void rspi_spti_isr(void)
Description	This function is called in the RSPI0.SPTI0 interrupt handling. Writes the transmit data. After transmitting the last data, disables generating the RSPI transmit interrupt request.
Arguments	None
Return Value	None

rspi_spri_isr	
Outline	RSPI receive interrupt
Header	None
Declaration	static void rspi_spri_isr(void)
Description	This function is called in the RSPI0.SPRI0 interrupt handling. Stores the receive data. After receiving the last data, disables the RSPI functions and calls the callback function (completion of RSPI transmission to/reception from the master).
Arguments	None
Return Value	None

rspi_spei_isr	
Outline	RSPI error interrupt
Header	None
Declaration	static void rspi_spei_isr(void)
Description	This function is called in the RSPI0.SPEI0 interrupt handling. Disables the RSPI function and calls the callback function (RSPI transmit/receive error).
Arguments	None
Return Value	None

Excep_RSPI0_SPEI0	
Outline	RSPI0.SPEI0 interrupt handling
Header	None
Declaration	static void Excep_RSPI0_SPEI0(void)
Description	Performs processing for the RSPI error interrupt.
Arguments	None
Return Value	None

Excep_RSPI0_SPRI0	
Outline	RSPI0.SPRI0 interrupt handling
Header	None
Declaration	static void Excep_RSPI0_SPRI0(void)
Description	Performs processing for the RSPI receive interrupt.
Arguments	None
Return Value	None

Excep_RSPI0_SPTI0	
Outline	RSPI0.SPTI0 interrupt handling
Header	None
Declaration	static void Excep_RSPI0_SPTI0(void)
Description	Performs processing for the RSPI transmit interrupt.
Arguments	None
Return Value	None

7.9 Flowcharts

7.9.1 Main Processing

Figure 7.4 shows the Main Processing.

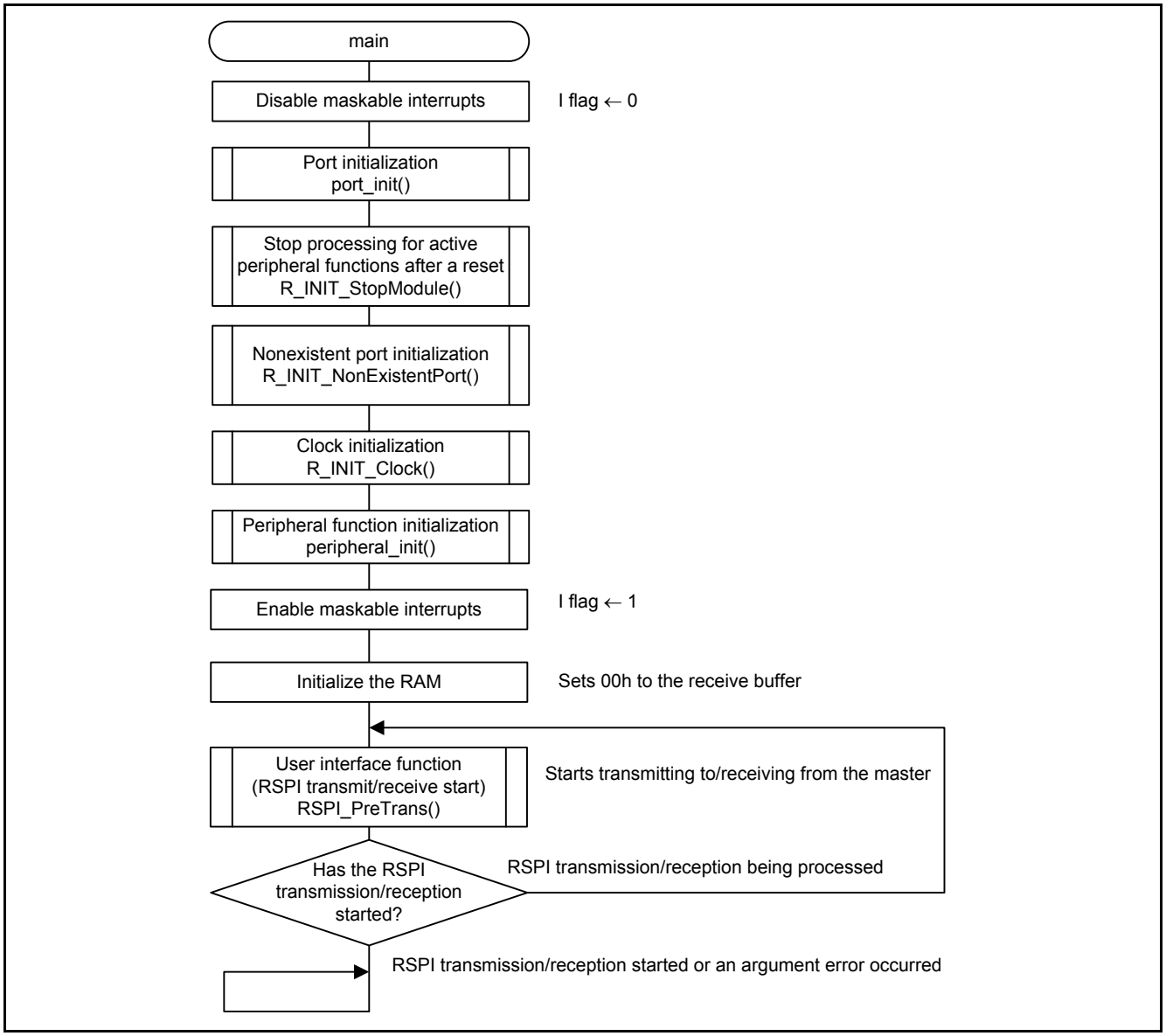


Figure 7.4 Main Processing

7.9.2 Port Initialization

Figure 7.5 shows the Port Initialization.

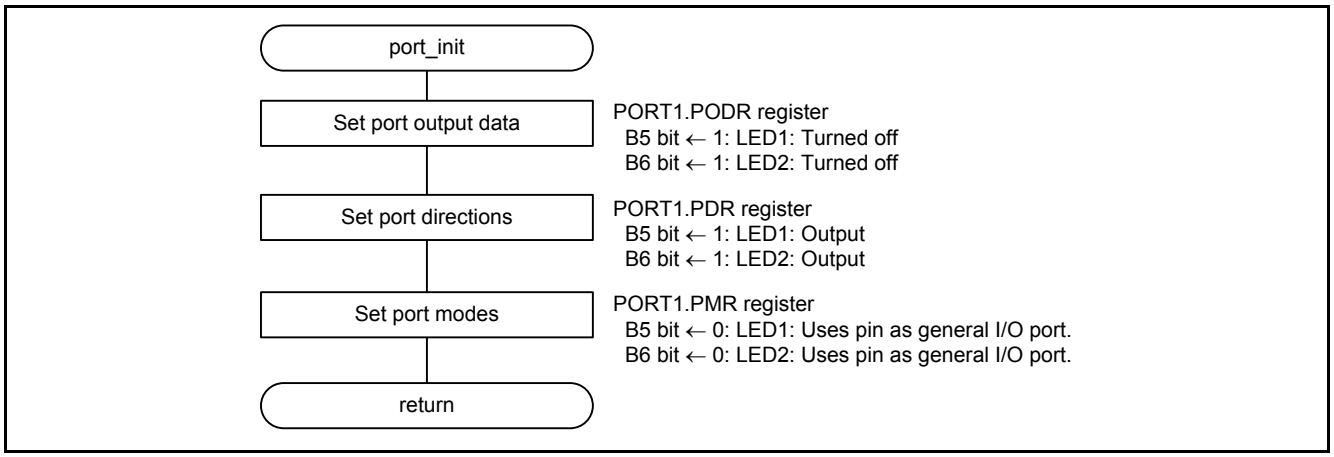


Figure 7.5 Port Initialization

7.9.3 Peripheral Function Initialization

Figure 7.6 shows the Peripheral Function Initialization.

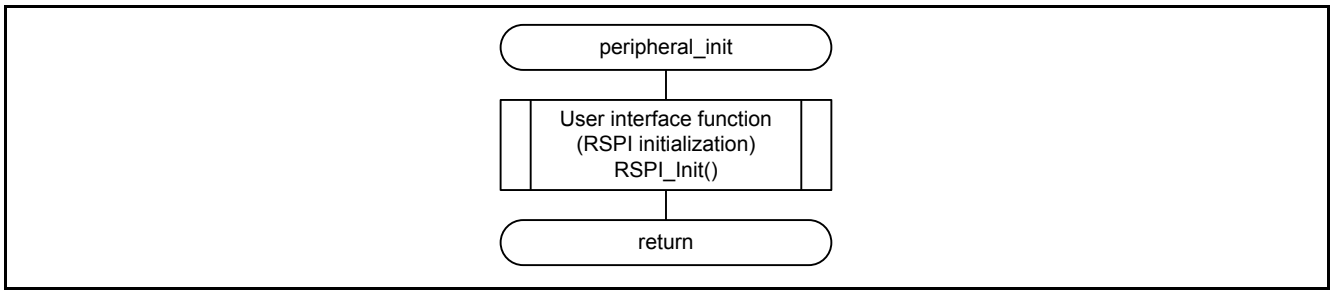


Figure 7.6 Peripheral Function Initialization

7.9.4 Callback Function (Completion of RSPI Transmission to/Reception from the Master)

Figure 7.7 shows the Callback Function (Completion of RSPI Transmission to/Reception from the Master).

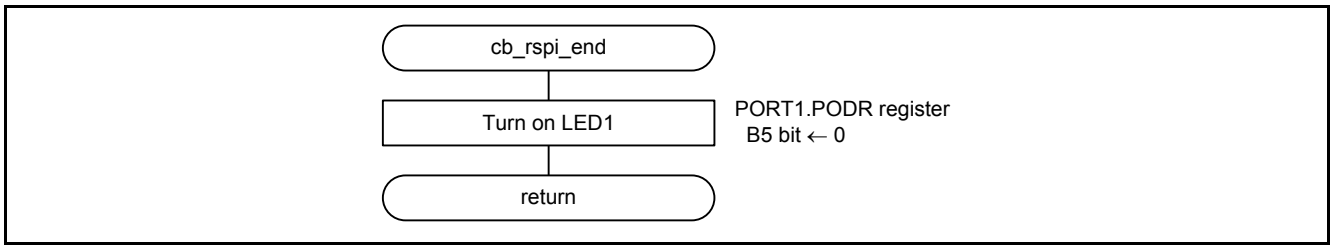
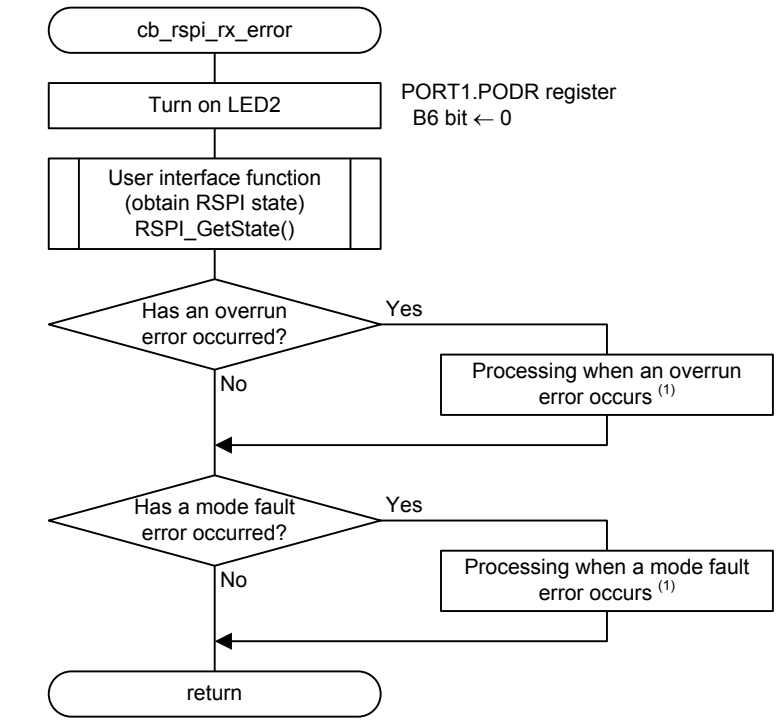


Figure 7.7 Callback Function (Completion of RSPI Transmission to/Reception from the Master)

7.9.5 Callback Function (RSPI Transmit/Receive Error)

Figure 7.8 shows the Callback Function (RSPI Transmit/Receive Error).



Note:
1. Error processing is not performed in the sample code. Add a program as required.

Figure 7.8 Callback Function (RSPI Transmit/Receive Error)

7.9.6 User Interface Function (RSPI Initialization)

Figure 7.9 and Figure 7.10 show the User Interface Function (RSPI Initialization).

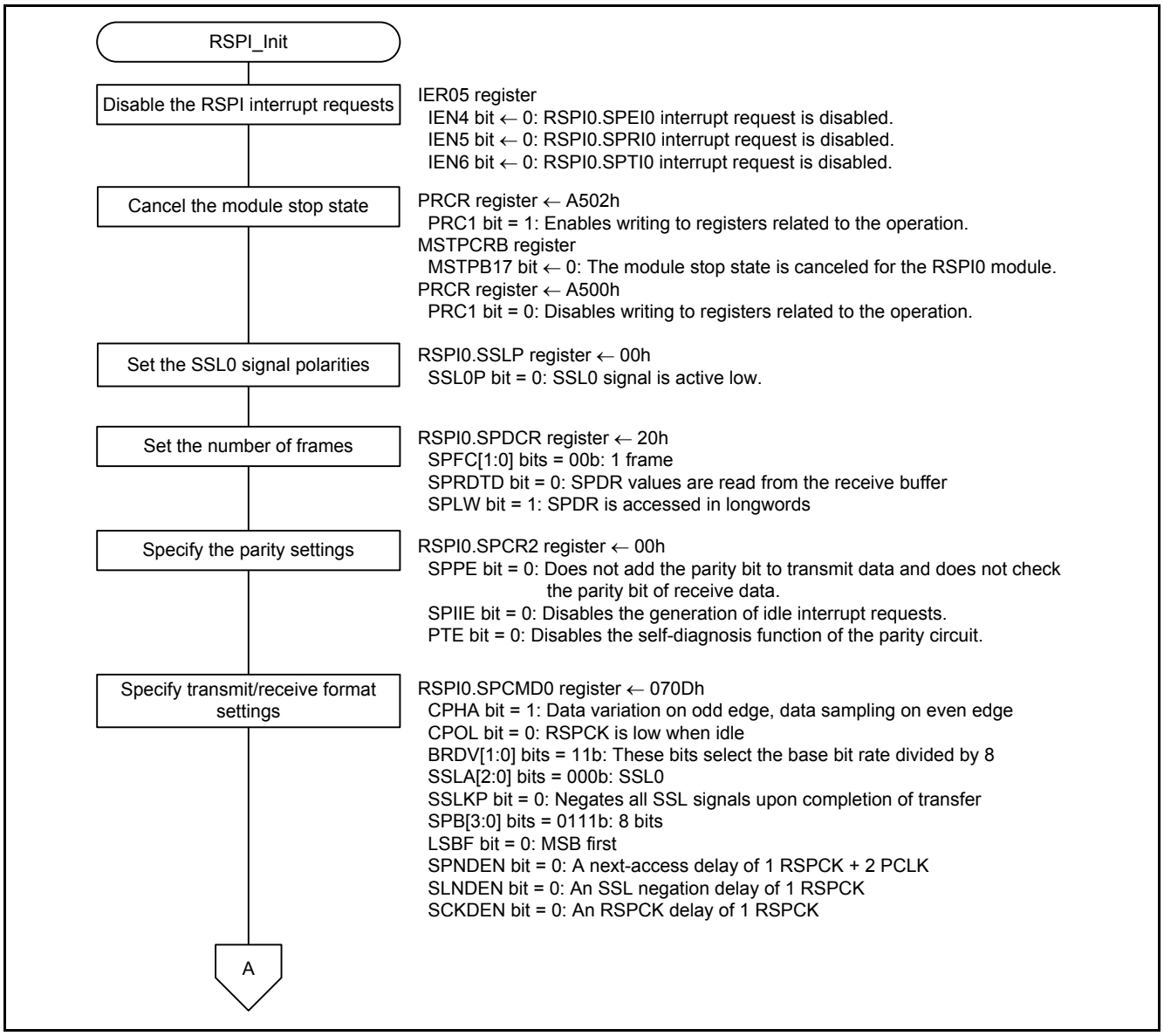


Figure 7.9 User Interface Function (RSPI Initialization) (1/2)

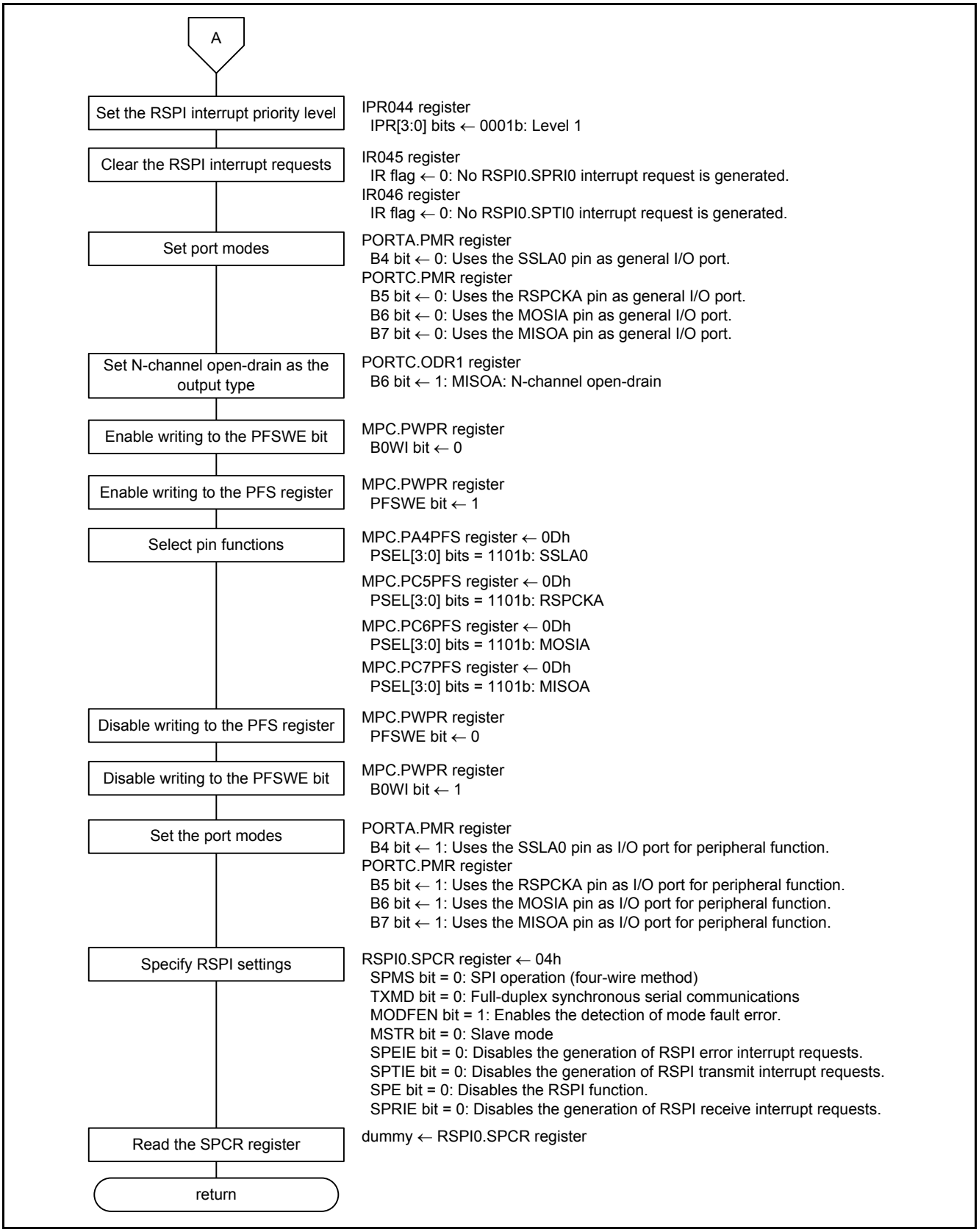


Figure 7.10 User Interface Function (RSPI Initialization) (2/2)

7.9.7 User Interface Function (RSPI Transmit/Receive Start)

Figure 7.11 and Figure 7.12 show the RSPI Transmit/Receive Start.

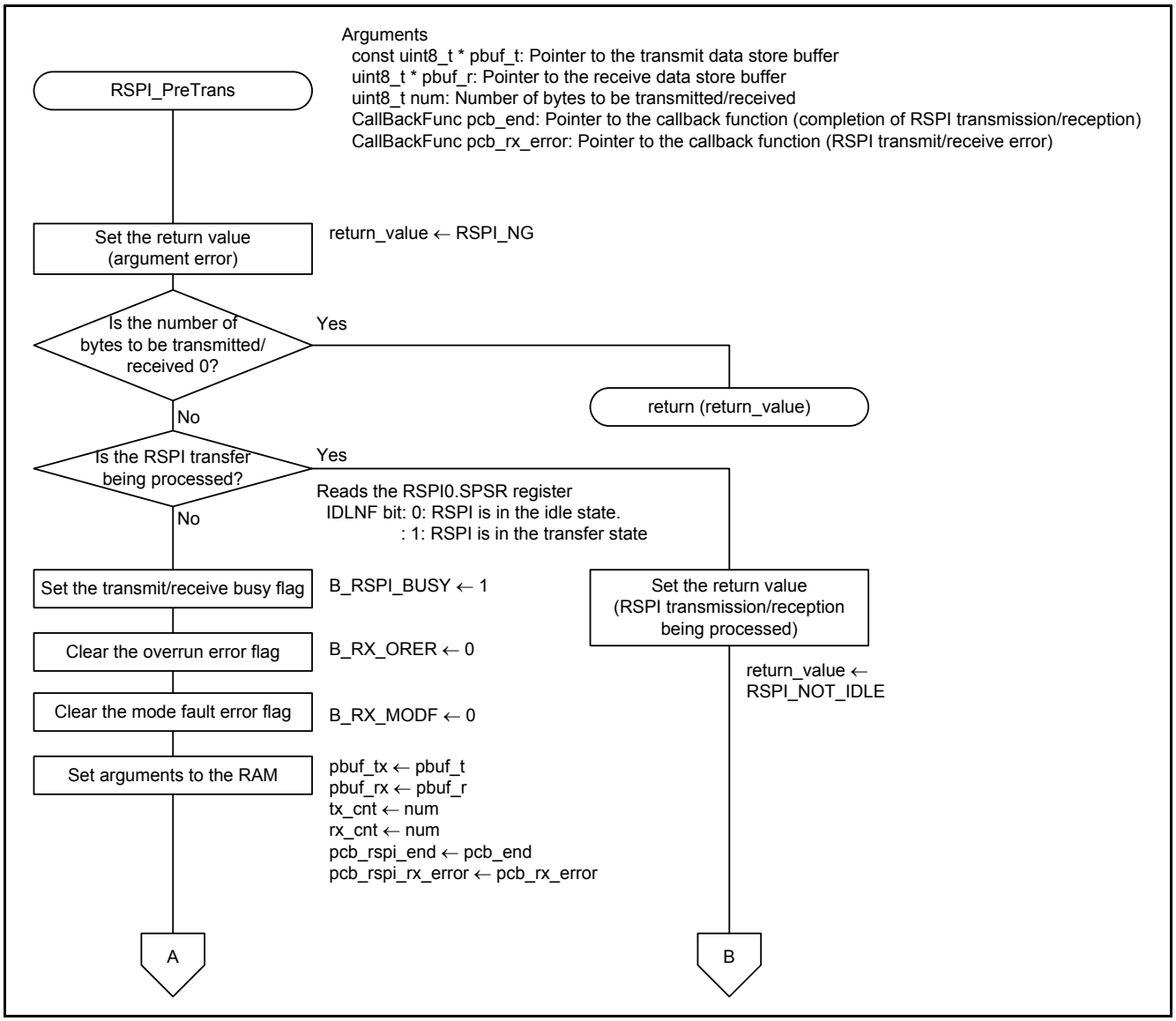
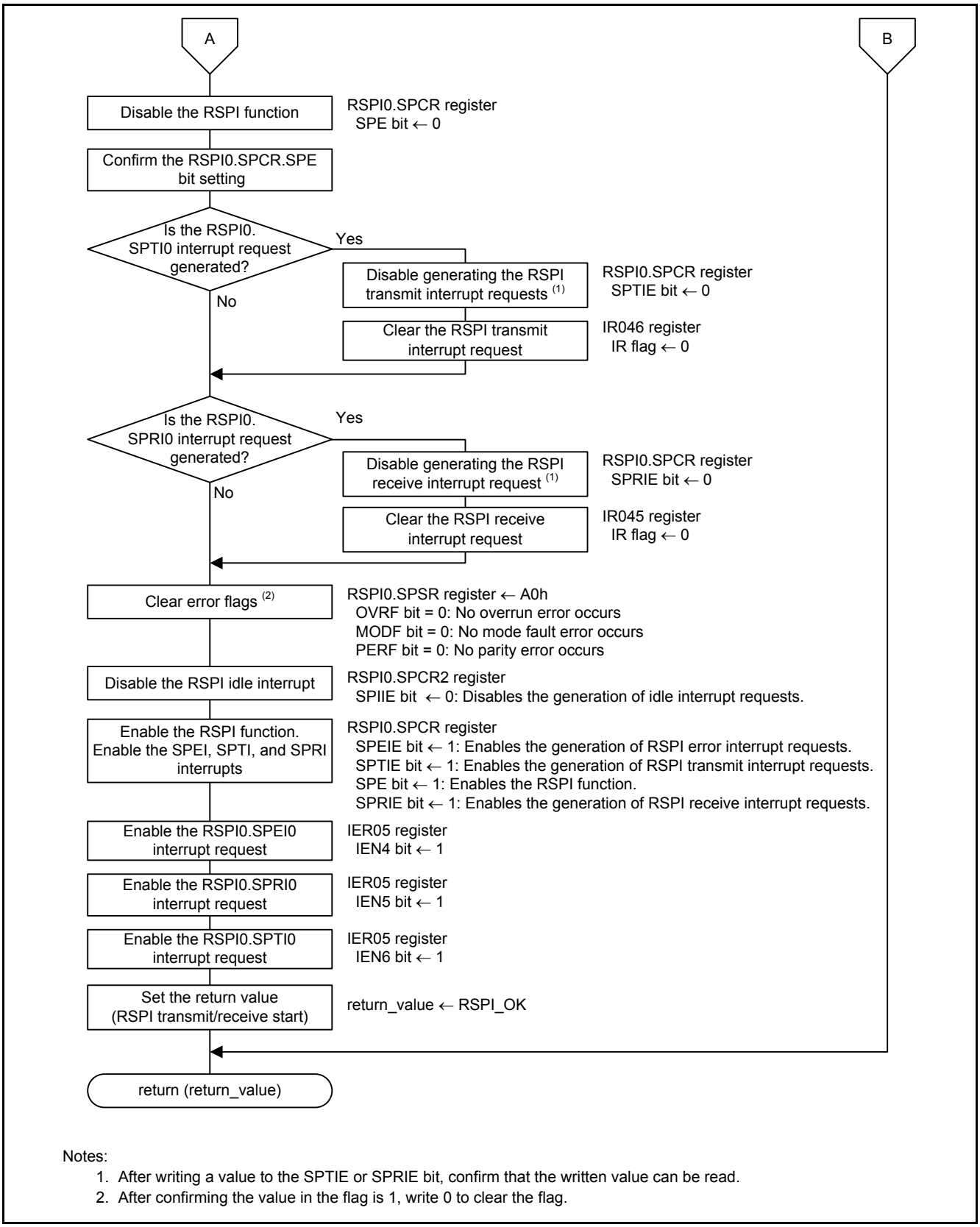


Figure 7.11 User Interface Function (RSPI Transmit/Receive Start) (1/2)



Notes:
1. After writing a value to the SPTIE or SPRIE bit, confirm that the written value can be read.
2. After confirming the value in the flag is 1, write 0 to clear the flag.

Figure 7.12 User Interface Function (RSPI Transmit/Receive Start) (2/2)

7.9.8 User Interface Function (Obtain RSPI State)

Figure 7.13 shows the User Interface Function (Obtain RSPI State).

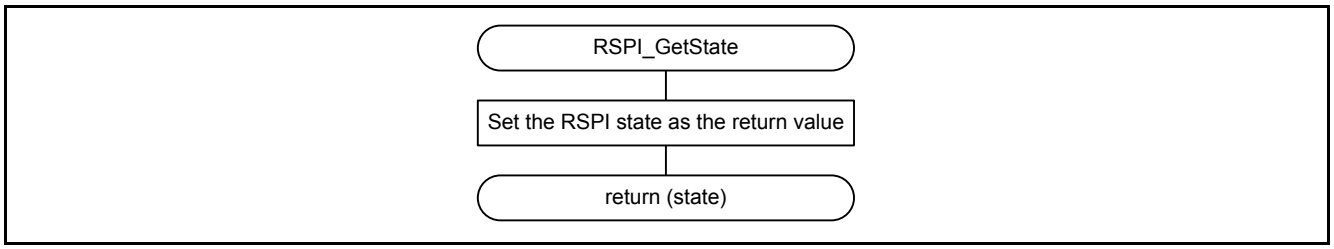


Figure 7.13 User Interface Function (Obtain RSPI State)

7.9.9 RSPI Transmit Interrupt

Figure 7.14 shows the RSPI Transmit Interrupt.

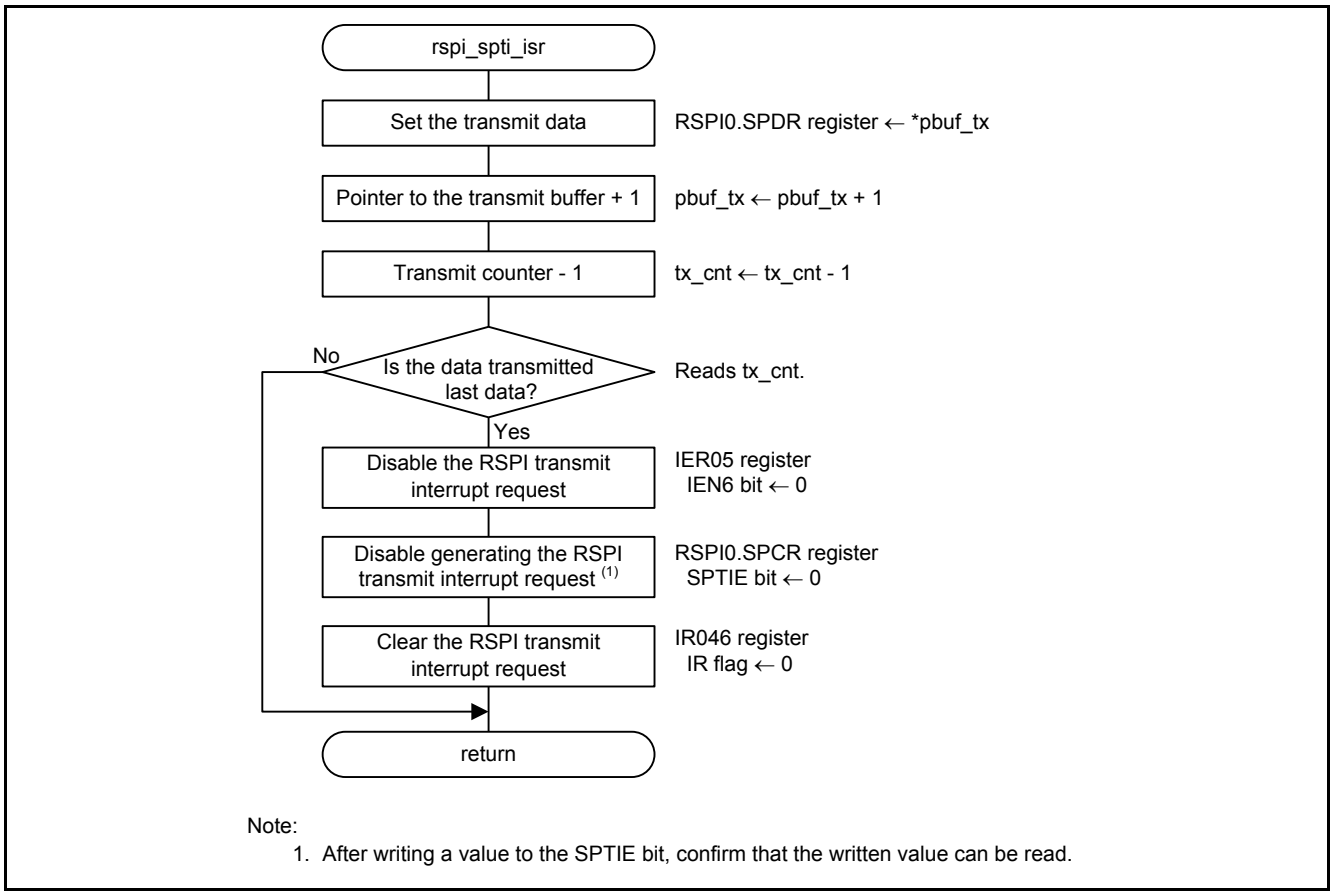
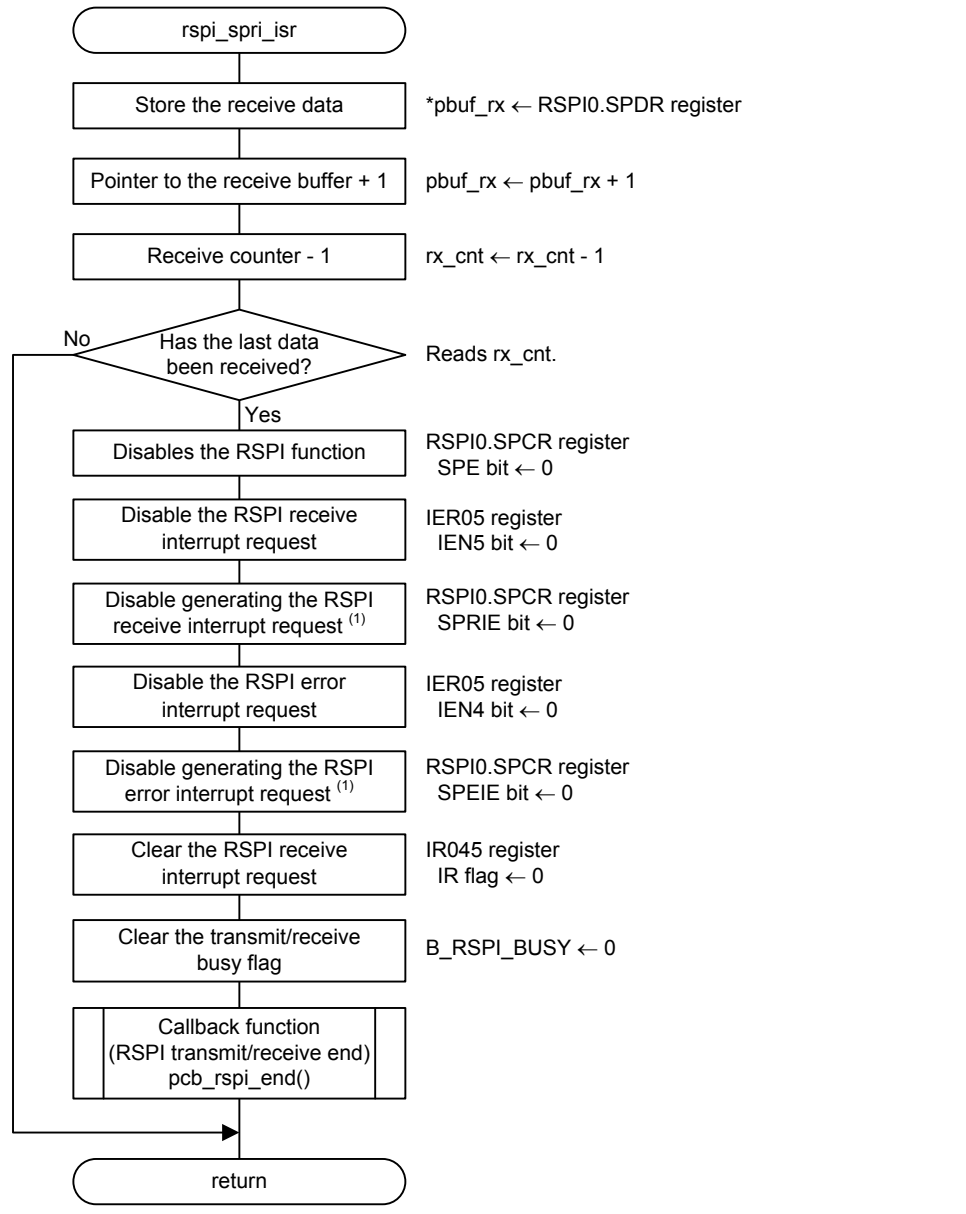


Figure 7.14 RSPI Transmit Interrupt

7.9.10 RSPI Receive Interrupt

Figure 7.15 shows the RSPI Receive Interrupt.

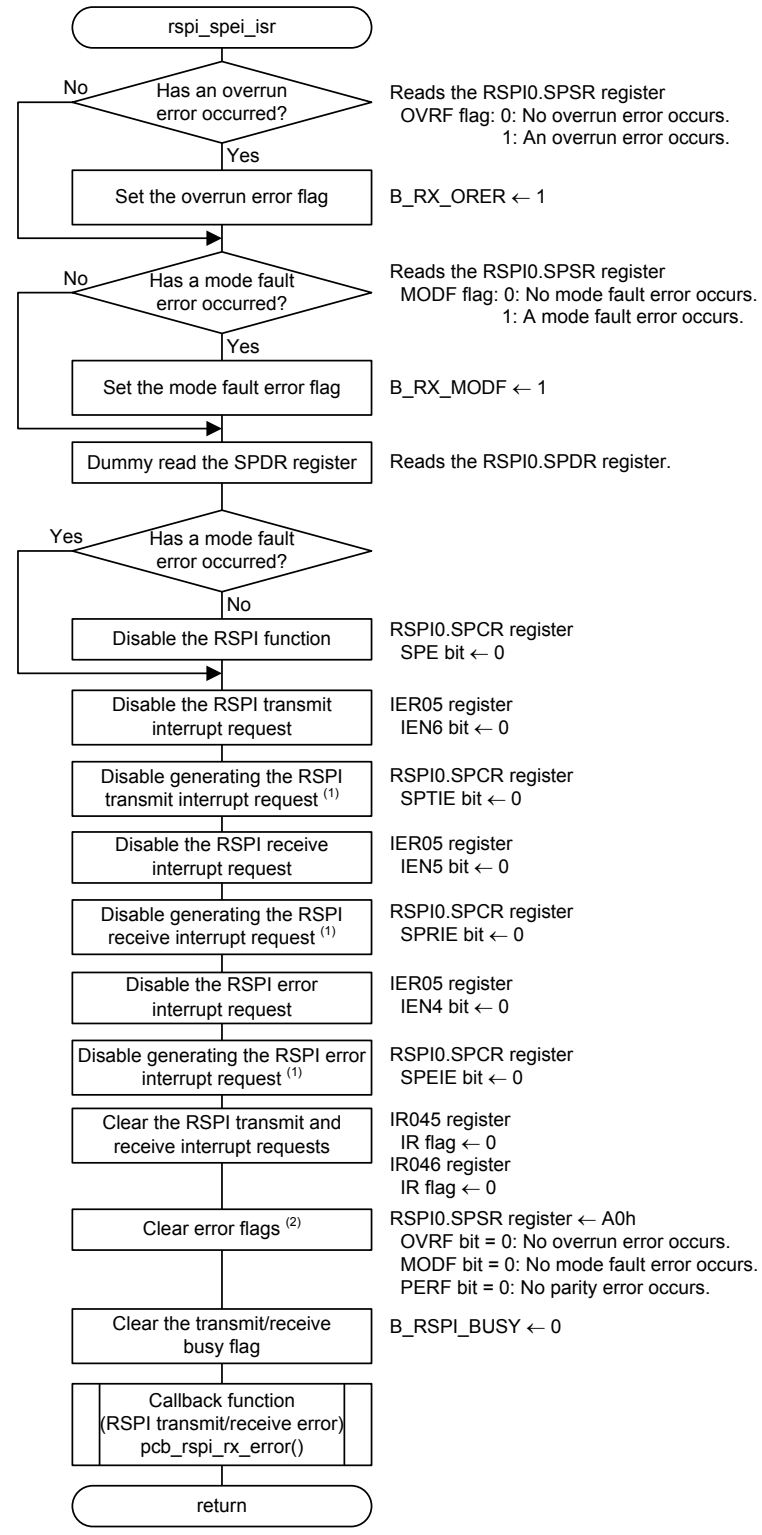


Note:
1. After writing a value to the SPRIE bit or SPEIE bit, confirm that the written value can be read.

Figure 7.15 RSPI Receive Interrupt

7.9.11 RSPI Error Interrupt

Figure 7.16 shows the RSPI Error Interrupt.



Notes:
1. After writing a value to the SPTIE, SPRIE, or SPEIE bit, confirm that the written value can be read.
2. After confirming the value in the flag is 1, write 0 to clear the flag.

Figure 7.16 RSPI Error Interrupt

7.9.12 RSPI0.SPEI0 Interrupt Handling

Figure 7.17 shows the RSPI0.SPEI0 Interrupt Handling.

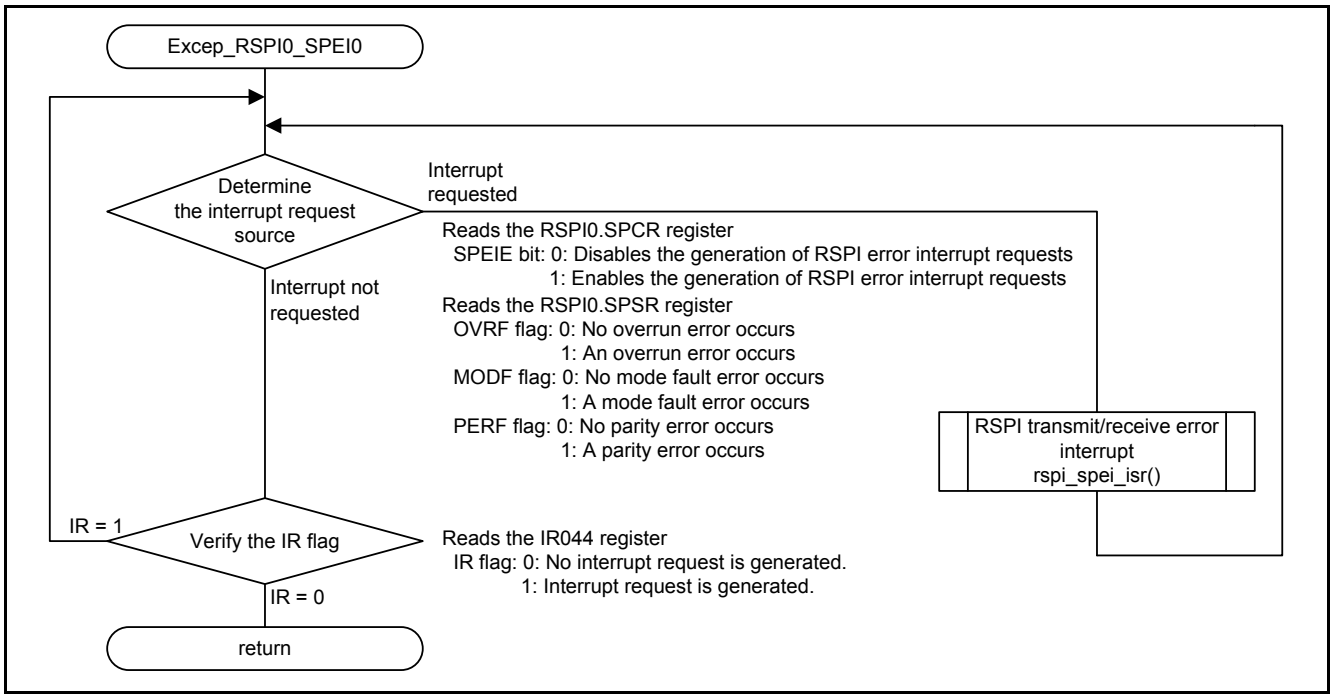


Figure 7.17 RSPI0.SPEI0 Interrupt Handling

7.9.13 RSPI0.SPRI0 Interrupt Handling

Figure 7.18 shows the RSPI0.SPRI0 Interrupt Handling.

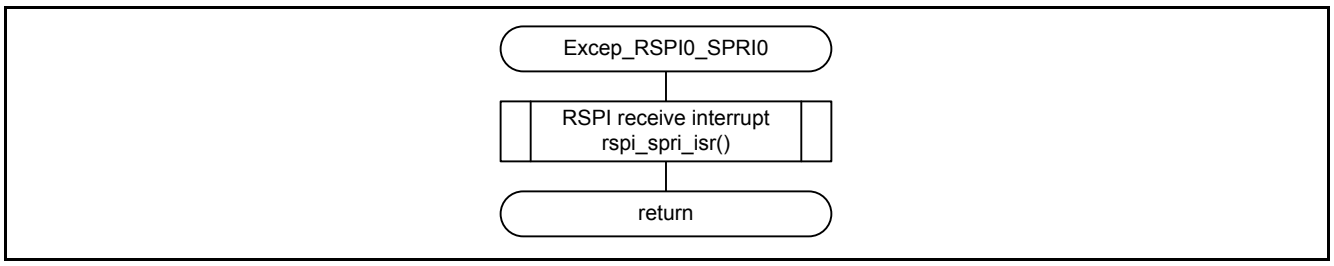


Figure 7.18 RSPI0.SPRI0 Interrupt Handling

7.9.14 RSPI0.SPTI0 Interrupt Handling

Figure 7.19 shows the RSPI0.SPTI0 Interrupt Handling.

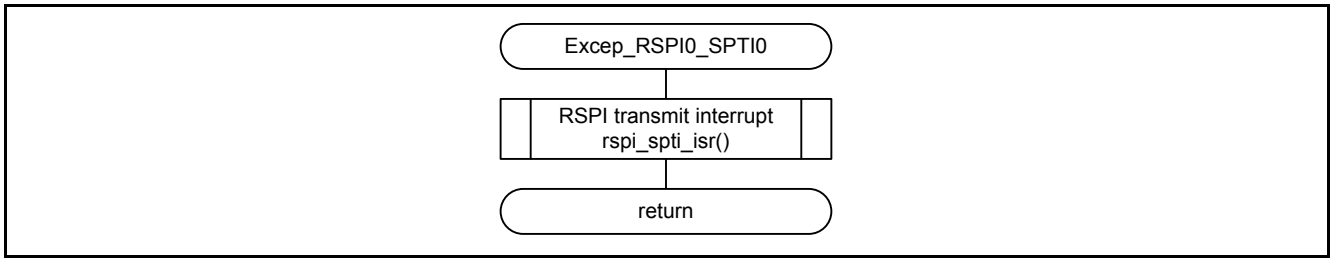


Figure 7.19 RSPI0.SPTI0 Interrupt Handling

8. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

9. Reference Documents

User's Manual: Hardware
RX220 Group User's Manual: Hardware Rev.1.00 (R01UH0292EJ)
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools
RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (R20UT0570EJ)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website
http://www.renesas.com

Inquiries
http://www.renesas.com/contact/

REVISION HISTORY	RX220 Group Application Note Communication Example Using the RSPI

Rev.	Date	Description	
		Page	Summary
1.00	Dec. 16, 2013	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 LanGao Rd., Putuo District, Shanghai, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, JIn Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

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

Colophon 3.0