RX220 Group

Asynchronous Communication Using the SCI

R01AN1708EJ0100
Rev. 1.00
Dec 16, 2013

Abstract

This document describes the method to perform asynchronous serial transmission and reception using the serial communications interface (SCI) in the RX220 Group.

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

Asynchronous serial transmission and reception are performed using the SCI.

After a reset, transmission and reception are only performed once each. 12 bytes of character code spelling out "Hello world!" is transmitted from the transmit buffer. When the 12-byte transmission is completed, LED0 is turned on. The 12-byte data is received and stored in the receive buffer. When the 12-byte reception is completed, LED1 is turned on. If an error occurs during reception, the receive operation is terminated and LED2 is turned on.

The SCI channel used is selected in the configuration file. Channel 1 (SCI1) is selected in the sample code.

- Transfer rate: 57600 bps
- Data length: 8 bits
- Stop bit: 2 bits
- Parity: None
- Hardware flow control: None

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
SCI (selectable from channels 1, 5, 6, 9, and 12)	Asynchronous serial transmission and reception
I/O ports	Turn on LEDs

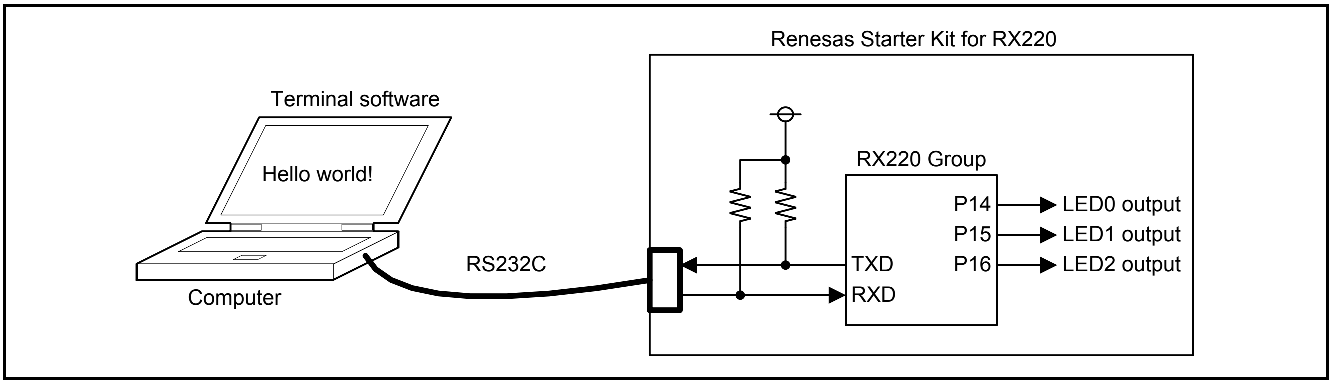


Figure 1.1 Usage Example

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.: R0K505220C000BE)
Tool used	Terminal software

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

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 package used.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P14	Output	LED0 output (completion of SCI transmission)
P15	Output	LED1 output (completion of SCI reception)
P16	Output	LED2 output (SCI reception error)
P15/RXD1	Input	Input pin for SCI1 receive data (1)
P16/TXD1	Output	Output pin for SCI1 transmit data (1)
PA3/RXD5	Input	Input pin for SCI5 receive data (1)
PA4/TXD5	Output	Output pin for SCI5 transmit data (1)
PB0/RXD6	Input	Input pin for SCI6 receive data (1)
PB1/TXD6	Output	Output pin for SCI6 transmit data (1)
PB6/RXD9	Input	Input pin for SCI9 receive data (1)
PB7/TXD9	Output	Output pin for SCI9 transmit data (1)
PE2/RXD12	Input	Input pin for SCI12 receive data (1)
PE1/TXD12	Output	Output pin for SCI12 transmit data (1)

Note:

1. The SCI pins used depend on the SCI channel selected in the configuration file. Unused SCI pins can be used as general I/O ports.

5. Software

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

When the user interface function (SCI receive start) is called, receive operation is enabled. When data for the specified number of bytes have been received, the SCI receive operation is disabled and the callback function (SCI receive end) is called. LED1 is turned on with the callback function (SCI receive end).

When a receive error occurs, the SCI receive operation is disabled and the callback function (SCI reception error) is called. LED2 is turned on with the callback function (SCI receive error).

When the user interface function (SCI transmit start) is called, the transmit operation is enabled. When data for the specified number of bytes have been transmitted, the SCI transmit operation is disabled and the callback function (SCI transmit end) is called. LED0 is turned on with the callback function (SCI transmit end).

Peripheral function settings are shown below and Figure 5.1 shows the Software Configuration.

SCI

- Serial communication mode: Asynchronous operation
- Transfer rate: 57600 bps
- Clock source: PCLKB (20 MHz)
- Data length: 8 bits
- Stop bit: 2 bits
- Parity: None
- Interrupts: Receive error interrupt (ERI) enabled
  Receive data full interrupt (RXI) enabled
  Transmit data empty interrupt (TXI) enabled
  Transmit end interrupt (TEI) enabled

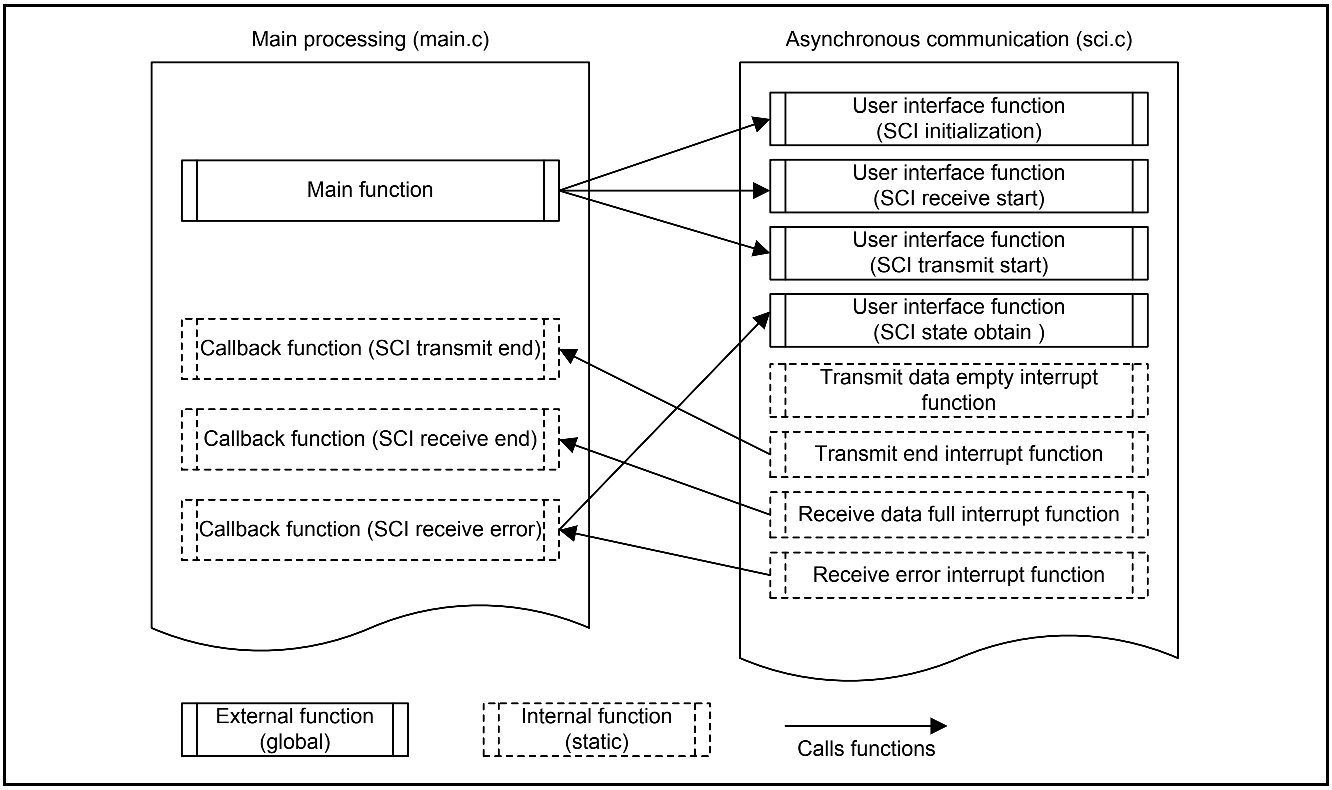


Figure 5.1 Software Configuration

5.1 Operation Overview

5.1.1 Serial Transmission

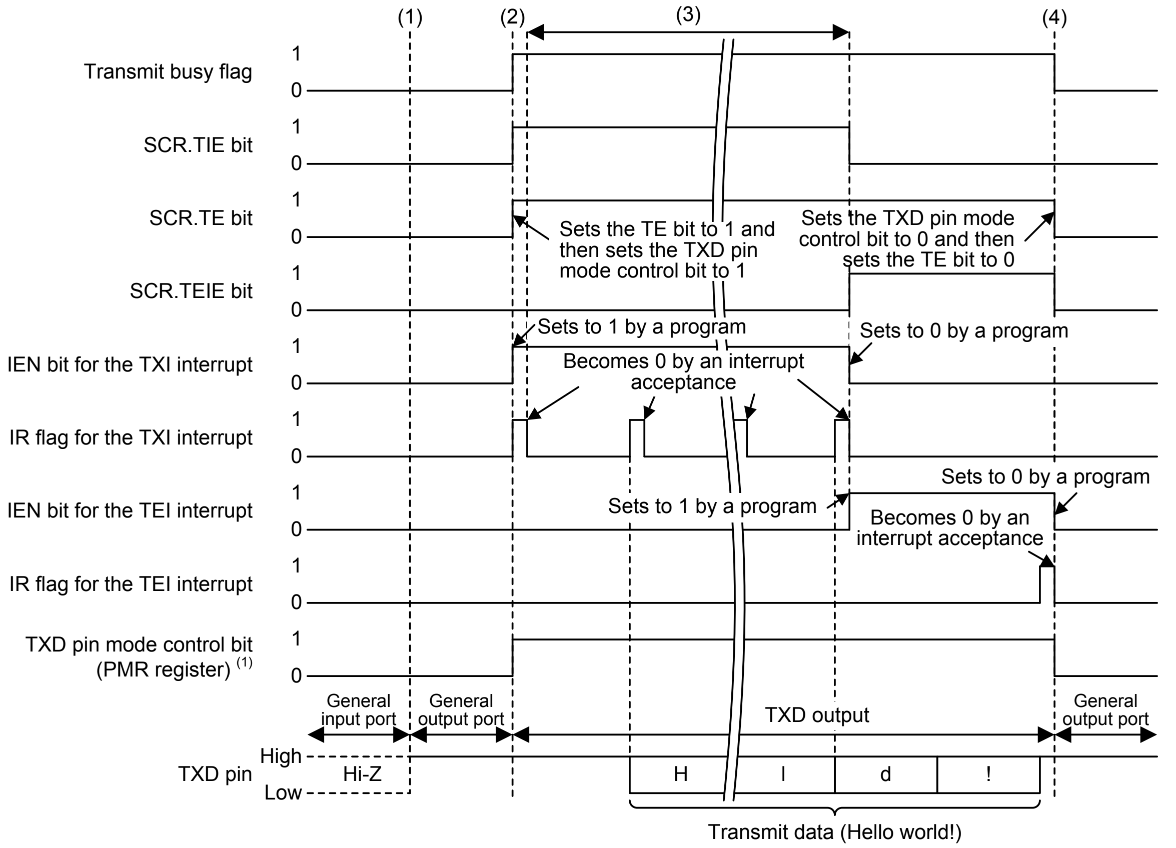
Figure 5.2 shows the Timing of Serial Transmission and (1) to (4) in the figure correspond to numbers in the operation descriptions below.

(1) Initialization
Initializes the SCI using the user interface function (SCI initialization) and switches the output level on the TXD pin to high.

(2) Starting a transmission
Verifies the transmit busy flag (B_TX_BUSY) using the user interface function (SCI transmit start). When the flag is 1 (transmission busy), SCI_BUSY (SCI transmission being processed) is returned. When the flag is 0 (transmission ready), sets the transmit busy flag to 1, the SCR.TIE bit to 1 (a TXI interrupt request is enabled), and the SCR.TE bit to 1 (serial transmission is enabled). When the TE bit is set to 1, the TXD pin becomes enabled. Then sets the TXD pin mode control bit to 1 (use pin as I/O port for peripheral functions) to switch the pin function to TXD output. When the IEN bit for the TXI interrupt is set to 1 (interrupt request is enabled), the TXI interrupt request is generated.

(3) Transmitting data
In the TXI interrupt handling, the value in the transmit buffer is written to the TDR register. When the last data is written, sets the TIE bit to 0 (a TXI interrupt request is disabled) and the SCR.TEIE bit to 1 (a TEI interrupt request is enabled).

(4) Completing the transmission
When the last data has been transmitted, a TEI interrupt request is generated. In the TEI interrupt handling, sets the TXD pin mode control bit to 0 (use pin as general I/O port), the TE bit to 0 (serial transmission is disabled), and the TEIE bit to 0 (a TEI interrupt request is disabled). Sets the transmit busy flag to 0 and calls the callback function (SCI transmit end).



Note:
1. The bit specified depends on the SCI channel selected.

Figure 5.2 Timing of Serial Transmission

5.1.2 Serial Reception

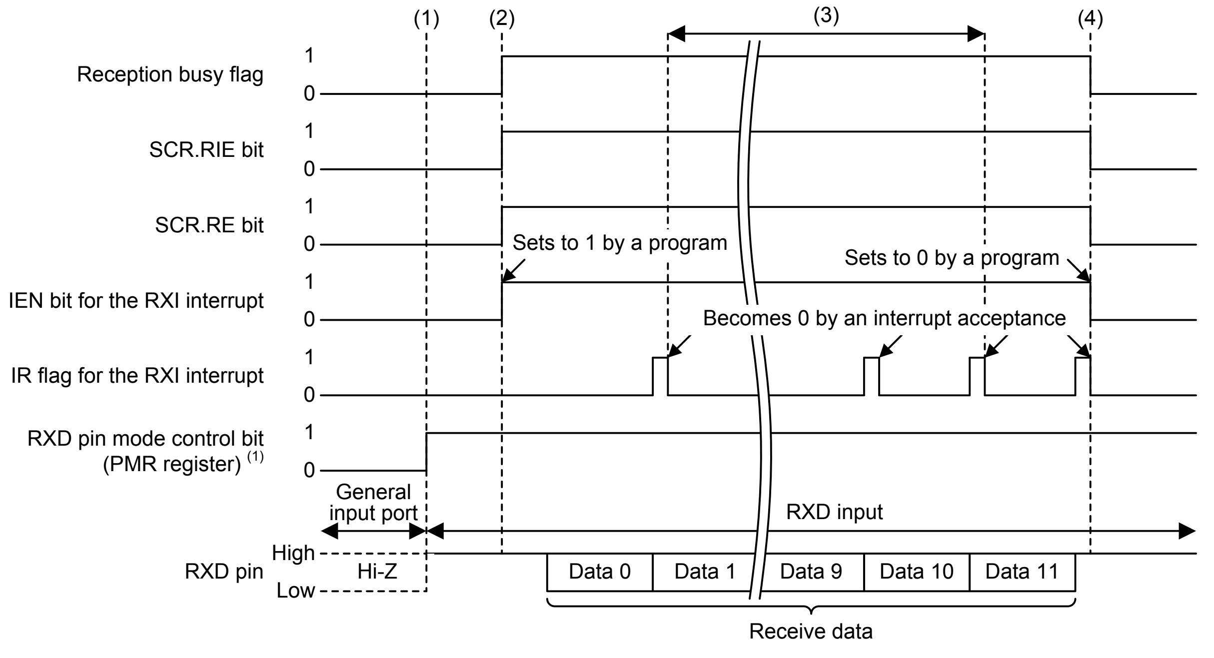
Figure 5.3 shows the Timing of Serial Reception and (1) to (4) in the figure correspond to numbers in the operation descriptions below.

(1) Initialization
Initializes the SCI using the user interface function (SCI initialization) and switches the RXD pin function to RXD input.

(2) Starting a reception
Verifies the receive busy flag (B_RX_BUSY) using the user interface function (SCI receive start). When the flag is 1 (reception busy), SCI_BUSY (SCI reception being processed) is returned. When the flag is 0 (reception ready), sets the receive busy flag to 1 and clears the error flags. Sets the SCR.RIE bit to 1 (RXI and ERI interrupt requests are enabled), the SCR.RE bit to 1 (serial reception is enabled), and the IEN bit for the RXI and ERI interrupts to 1 (interrupt request is enabled).

(3) Receiving data
When data is received, an RXI interrupt request is generated. In the RXI interrupt handling, stores the RDR register value in the receive buffer.
When a reception error occurs, an ERI interrupt request is generated. In the ERI interrupt handling, sets the error flag variable and dummy reads the RDR register. Sets the RE bit to 0 and clears the error flags in the SSR register. Sets the RIE bit to 0, the receive busy flag to 0, and calls the callback function (SCI reception error).

(4) Completing the reception
When the last data has been received, in the RXI interrupt handling, sets the RE bit to 0 (serial reception is disabled) and RIE bit to 0 (RXI and ERI interrupt requests are disabled). Sets the receive busy flag to 0 and calls the callback function (SCI receive end).



Note:
1. The bit specified depends on the SCI channel selected.

Figure 5.3 Timing of Serial Reception

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	
sci.c	Asynchronous communication	
sci.h	Header file for sci.c	
sci_cfg.h	Header file for sci.c configuration file	SCI channel selection

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 5.10 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
BUF_SIZE	12	Buffer size
NULL_SIZE	1	NULL code size
SCI_B_TX_BUSY	sci_state.bit.b_tx_busy	Transmit busy flag 0: Transmission ready 1: Transmission busy
SCI_B_RX_BUSY	sci_state.bit.b_rx_busy	Receive busy flag 0: Reception ready 1: Reception busy
SCI_B_RX_ORER	sci_state.bit.b_rx_orer	Overrun error flag 0: Overrun error not occurred 1: Overrun error occurred
SCI_B_RX_FER	sci_state.bit.b_rx_fer	Framing error flag 0: Framing error not occurred 1: Framing error occurred

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

Constant Name	Setting Value	Contents
SSR_ERROR_FLAGS	38h	Bit pattern of an error flag in the SCI.SSR register
B_TX_BUSY	state.bit.b_tx_busy	Transmit busy flag 0: Transmission ready 1: Transmission busy
B_RX_BUSY	state.bit.b_rx_busy	Receive busy flag 0: Reception ready 1: Reception busy
B_RX_ORER	state.bit.b_rx_orer	Overrun error flag 0: Overrun error not occurred 1: Overrun error occurred
B_RX_FER	state.bit.b_rx_fer	Framing error flag 0: Framing error not occurred 1: Framing error occurred

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

Constant Name	Setting Value	Contents
SCI_OK	00h	Return value of the SCI_StartTransmit and SCI_StartReceive functions: SCI transmit/receive start
SCI_BUSY	01h	Return value of the SCI_StartTransmit and SCI_StartReceive functions: SCI transmission or reception being processed
SCI_NG	02h	Return value of the SCI_StartTransmit and SCI_StartReceive functions: Argument error (number of bytes to be transmitted/received is 0)

Table 5.6 Constants Used in the Sample Code (when SELECT_SCI1 is Selected in sci_cfg.h)

Constant Name	Setting Value	Contents
SCIn	SCI1	SCI channel: SCI1
MSTP_SCIn	MSTP(SCI1)	SCI1 module stop setting bit
IPR_SCIn	IPR(SCI1,)	SCI1 interrupt priority level setting bit
IR_SCIn_ERIn	IR(SCI1,ERI1)	SCI1.ERI1 interrupt status flag
IR_SCIn_RXIn	IR(SCI1,RXI1)	SCI1.RXI1 interrupt status flag
IR_SCIn_TXIn	IR(SCI1,TXI1)	SCI1.TXI1 interrupt status flag
IR_SCIn_TEIn	IR(SCI1,TEI1)	SCI1.TEI1 interrupt status flag
IEN_SCIn_ERIn	IEN(SCI1,ERI1)	SCI1.ERI1 interrupt request enable bit
IEN_SCIn_RXIn	IEN(SCI1,RXI1)	SCI1.RXI1 interrupt request enable bit
IEN_SCIn_TXIn	IEN(SCI1,TXI1)	SCI1.TXI1 interrupt request enable bit
IEN_SCIn_TEIn	IEN(SCI1,TEI1)	SCI1.TEI1 interrupt request enable bit
RXDn_PDR	PORT1.PDR.BIT.B5	P15 I/O select bit
RXDn_PMR	PORT1.PMR.BIT.B5	P15 pin mode control bit
RXDnPFS	P15PFS	P15 pin function control register
TXDn_PODR	PORT1.PODR.BIT.B6	P16 output data store bit
TXDn_PDR	PORT1.PDR.BIT.B6	P16 I/O select bit
TXDn_PMR	PORT1.PMR.BIT.B6	P16 pin mode control bit
TXDnPFS	P16PFS	P16 pin function control register
PSEL_SETTING	0Ah	Setting value of the pin function select bit: RXD1, TXD1

Table 5.7 Constants Used in the Sample Code (when SELECT_SCI5 is Selected in sci_cfg.h)

Constant Name	Setting Value	Contents
SCIn	SCI5	SCI channel: SCI5
MSTP_SCIn	MSTP(SCI5)	SCI5 module stop setting bit
IPR_SCIn	IPR(SCI5,)	SCI5 interrupt priority level setting bit
IR_SCIn_ERIn	IR(SCI5,ERI5)	SCI5.ERI5 interrupt status flag
IR_SCIn_RXIn	IR(SCI5,RXI5)	SCI5.RXI5 interrupt status flag
IR_SCIn_TXIn	IR(SCI5,TXI5)	SCI5.TXI5 interrupt status flag
IR_SCIn_TEIn	IR(SCI5,TEI5)	SCI5.TEI5 interrupt status flag
IEN_SCIn_ERIn	IEN(SCI5,ERI5)	SCI5.ERI5 interrupt request enable bit
IEN_SCIn_RXIn	IEN(SCI5,RXI5)	SCI5.RXI5 interrupt request enable bit
IEN_SCIn_TXIn	IEN(SCI5,TXI5)	SCI5.TXI5 interrupt request enable bit
IEN_SCIn_TEIn	IEN(SCI5,TEI5)	SCI5.TEI5 interrupt request enable bit
RXDn_PDR	PORTA.PDR.BIT.B3	PA3 I/O select bit
RXDn_PMR	PORTA.PMR.BIT.B3	PA3 pin mode control bit
RXDnPFS	PA3PFS	PA3 pin function control register
TXDn_PODR	PORTA.PODR.BIT.B4	PA4 output data store bit
TXDn_PDR	PORTA.PDR.BIT.B4	PA4 I/O select bit
TXDn_PMR	PORTA.PMR.BIT.B4	PA4 pin mode control bit
TXDnPFS	PA4PFS	PA4 pin function control register
PSEL_SETTING	0Ah	Setting value of the pin function select bit: RXD5, TXD5

Table 5.8 Constants Used in the Sample Code (when SELECT_SCI6 is Selected in sci_cfg.h)

Constant Name	Setting Value	Contents
SCIn	SCI6	SCI channel: SCI6
MSTP_SCIn	MSTP(SCI6)	SCI6 module stop setting bit
IPR_SCIn	IPR(SCI6,)	SCI6 interrupt priority level setting bit
IR_SCIn_ERIn	IR(SCI6,ERI6)	SCI6.ERI6 interrupt status flag
IR_SCIn_RXIn	IR(SCI6,RXI6)	SCI6.RXI6 interrupt status flag
IR_SCIn_TXIn	IR(SCI6,TXI6)	SCI6.TXI6 interrupt status flag
IR_SCIn_TEIn	IR(SCI6,TEI6)	SCI6.TEI6 interrupt status flag
IEN_SCIn_ERIn	IEN(SCI6,ERI6)	SCI6.ERI6 interrupt request enable bit
IEN_SCIn_RXIn	IEN(SCI6,RXI6)	SCI6.RXI6 interrupt request enable bit
IEN_SCIn_TXIn	IEN(SCI6,TXI6)	SCI6.TXI6 interrupt request enable bit
IEN_SCIn_TEIn	IEN(SCI6,TEI6)	SCI6.TEI6 interrupt request enable bit
RXDn_PDR	PORTB.PDR.BIT.B0	PB0 I/O select bit
RXDn_PMR	PORTB.PMR.BIT.B0	PB0 pin mode control bit
RXDnPFS	PB0PFS	PB0 pin function control register
TXDn_PODR	PORTB.PODR.BIT.B1	PB1 output data store bit
TXDn_PDR	PORTB.PDR.BIT.B1	PB1 I/O select bit
TXDn_PMR	PORTB.PMR.BIT.B1	PB1 pin mode control bit
TXDnPFS	PB1PFS	PB1 pin function control register
PSEL_SETTING	0Bh	Setting value of the pin function select bit: RXD6, TXD6

Table 5.9 Constants Used in the Sample Code (when SELECT_SCI9 is Selected in sci_cfg.h) (1)

Constant Name	Setting Value	Contents
SCIn	SCI9	SCI channel: SCI9
MSTP_SCIn	MSTP(SCI9)	SCI9 module stop setting bit
IPR_SCIn	IPR(SCI9,)	SCI9 interrupt priority level setting bit
IR_SCIn_ERIn	IR(SCI9,ERI9)	SCI9.ERI9 interrupt status flag
IR_SCIn_RXIn	IR(SCI9,RXI9)	SCI9.RXI9 interrupt status flag
IR_SCIn_TXIn	IR(SCI9,TXI9)	SCI9.TXI9 interrupt status flag
IR_SCIn_TEIn	IR(SCI9,TEI9)	SCI9.TEI9 interrupt status flag
IEN_SCIn_ERIn	IEN(SCI9,ERI9)	SCI9.ERI9 interrupt request enable bit
IEN_SCIn_RXIn	IEN(SCI9,RXI9)	SCI9.RXI9 interrupt request enable bit
IEN_SCIn_TXIn	IEN(SCI9,TXI9)	SCI9.TXI9 interrupt request enable bit
IEN_SCIn_TEIn	IEN(SCI9,TEI9)	SCI9.TEI9 interrupt request enable bit
RXDn_PDR	PORTB.PDR.BIT.B6	PB6 I/O select bit
RXDn_PMR	PORTB.PMR.BIT.B6	PB6 pin mode control bit
RXDnPFS	PB6PFS	PB6 pin function control register
TXDn_PODR	PORTB.PODR.BIT.B7	PB7 output data store bit
TXDn_PDR	PORTB.PDR.BIT.B7	PB7 I/O select bit
TXDn_PMR	PORTB.PMR.BIT.B7	PB7 pin mode control bit
TXDnPFS	PB7PFS	PB7 pin function control register
PSEL_SETTING	0Ah	Setting value of the pin function select bit: RXD9, TXD9

Note:

1. SCI9 is not available in 48-pin products.

Table 5.10 Constants Used in the Sample Code (when SELECT_SCI12 is Selected in sci_cfg.h)

Constant Name	Setting Value	Contents
SCIn	SCI12	SCI channel: SCI12
MSTP_SCIn	MSTP(SCI12)	SCI12 module stop setting bit
IPR_SCIn	IPR(SCI12,)	SCI12 interrupt priority level setting bit
IR_SCIn_ERIn	IR(SCI12,ERI12)	SCI12.ERI12 interrupt status flag
IR_SCIn_RXIn	IR(SCI12,RXI12)	SCI12.RXI12 interrupt status flag
IR_SCIn_TXIn	IR(SCI12,TXI12)	SCI12.TXI12 interrupt status flag
IR_SCIn_TEIn	IR(SCI12,TEI12)	SCI12.TEI12 interrupt status flag
IEN_SCIn_ERIn	IEN(SCI12,ERI12)	SCI12.ERI12 interrupt request enable bit
IEN_SCIn_RXIn	IEN(SCI12,RXI12)	SCI12.RXI12 interrupt request enable bit
IEN_SCIn_TXIn	IEN(SCI12,TXI12)	SCI12.TXI12 interrupt request enable bit
IEN_SCIn_TEIn	IEN(SCI12,TEI12)	SCI12.TEI12 interrupt request enable bit
RXDn_PDR	PORTE.PDR.BIT.B2	PE2 I/O select bit
RXDn_PMR	PORTE.PMR.BIT.B2	PE2 pin mode control bit
RXDnPFS	PE2PFS	PE2 pin function control register
TXDn_PODR	PORTE.PODR.BIT.B1	PE1 output data store bit
TXDn_PDR	PORTE.PDR.BIT.B1	PE1 I/O select bit
TXDn_PMR	PORTE.PMR.BIT.B1	PE1 pin mode control bit
TXDnPFS	PE1PFS	PE1 pin function control register
PSEL_SETTING	0Ch	Setting value of the pin function select bit: RXD12, TXD12

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_tx_busy   :1;  /* Transmit busy flag   0: Transmission ready           1: Transmission busy */
        uint8_t    b_rx_busy   :1;  /* Receive busy flag    0: Reception ready              1: Reception busy */
        uint8_t    b_rx_orer   :1;  /* Overrun error flag   0: Overrun error not occurred   1: Overrun error occurred */
        uint8_t    b_rx_fer    :1;  /* Framing error flag   0: Framing error not occurred   1: Framing error occurred */
        uint8_t                :4;  /* Not used */
    } bit;
} sci_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 5.11 lists the static Variables.

Table 5.11 static Variables

Type	Variable Name	Contents	Function Used
static uint8_t	rx_buf[BUF_SIZE]	Receive buffer	main
static uint8_t	tx_buf[]	Transmit buffer	main
static sci_state_t	sci_state	SCI state	cb_sci_rx_error
static const uint8_t *	pbuf_tx	Pointer to the transmit buffer	SCI_StartTransmit sci_txi_isr
static uint8_t	tx_cnt	Transmit counter	
static uint8_t *	pbuf_rx	Pointer to the reception buffer	SCI_StartReceive sci_rxi_isr
static uint8_t	rx_cnt	Receive counter	
static sci_state_t	state	SCI state	SCI_StartReceive SCI_StartTransmit SCI_GetState sci_tei_isr sci_rxi_isr sci_eri_isr

5.7 Functions

Table 5.12 lists the Functions Used in the Sample Code.

Table 5.12 Functions Used in the Sample Code

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_sci_tx_end	Callback function (SCI transmit end)
cb_sci_rx_end	Callback function (SCI receive end)
cb_sci_rx_error	Callback function (SCI receive error)
SCI_Init	User interface function (SCI initialization)
SCI_StartReceive	User interface function (SCI receive start)
SCI_StartTransmit	User interface function (SCI transmit start)
SCI_GetState	User interface function (SCI state obtain)
sci_txi_isr	Transmit data empty interrupt
sci_tei_isr	Transmit end interrupt
sci_rxi_isr	Receive data full interrupt
sci_eri_isr	Receive error interrupt
Excep_SCIn_ERIn	SCI.ERI interrupt handling
Excep_SCIn_RXIn	SCI.RXI interrupt handling
Excep_SCIn_TXIn	SCI.TXI interrupt handling
Excep_SCIn_TEIn	SCI.TEI 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 SCI reception and then starts transmission.
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 the main clock 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_sci_tx_end	
Outline	Callback function (SCI transmit end)
Header	None
Declaration	static void cb_sci_tx_end(void)
Description	This function is called when the SCI transmission has been completed.
Arguments	None
Return Value	None

cb_sci_rx_end	
Outline	Callback function (SCI receive end)
Header	None
Declaration	static void cb_sci_rx_end(void)
Description	This function is called when the SCI reception has been completed.
Arguments	None
Return Value	None

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

SCI_Init	
Outline	User interface function (SCI initialization)
Header	sci.h
Declaration	void SCI_Init(void)
Description	Initializes the SCI.
Arguments	None
Return Value	None

SCI_StartReceive	
Outline	User interface function (SCI receive start)
Header	sci.h
Declaration	uint8_t SCI_StartReceive(uint8_t * pbuf, uint8_t num, CallBackFunc pcb_rx_end, CallBackFunc pcb_rx_error)
Description	Starts SCI reception.
Arguments	uint8_t * pbuf: Pointer to the receive data storage uint8_t num: Number of bytes to be received CallBackFunc pcb_rx_end: Pointer to the callback function (SCI receive end) CallBackFunc pcb_rx_error: Pointer to the callback function (SCI receive error)
Return Value	SCI_NG: Argument error (number of bytes to be received is 0) SCI_BUSY: SCI reception being processed SCI_OK: SCI reception started

SCI_StartTransmit	
Outline	User interface function (SCI transmit start)
Header	sci.h
Declaration	uint8_t SCI_StartTransmit(const uint8_t * pbuf, uint8_t num, CallBackFunc pcb_tx_end)
Description	Starts SCI transmission.
Arguments	const uint8_t * pbuf: Pointer to the transmit data storage uint8_t num: Number of bytes to be transmitted CallBackFunc pcb_tx_end: Pointer to the callback function (transmit end)
Return Value	SCI_NG: Argument error (number of bytes to be transmitted is 0) SCI_BUSY: SCI transmission being processed SCI_OK: SCI transmission started

SCI_GetState	
Outline	User interface function (SCI state obtain)
Header	sci.h
Declaration	sci_state_t SCI_GetState(void)
Description	Returns the SCI state.
Arguments	None
Return Value	sci_state_t.bit.b_tx_busy: Transmit busy flag 0: Transmission ready 1: Transmission busy sci_state_t.bit.b_rx_busy: Receive busy flag 0: Reception ready 1: Reception busy sci_state_t.bit.b_rx_orer: Overrun error flag 0: Overrun error not occurred 1: Overrun error occurred sci_state_t.bit.b_rx_fer: Framing error flag 0: Framing error not occurred 1: Framing error occurred

sci_txi_isr	
Outline	Transmit data empty interrupt
Header	None
Declaration	static void sci_txi_isr(void)
Description	This function is called in the SCI.TXI interrupt handling. Writes the transmit data. After transmitting the last data, disables the TXI interrupt request and enables TEI interrupt request.
Arguments	None
Return Value	None

sci_tei_isr	
Outline	Transmit end interrupt
Header	None
Declaration	static void sci_tei_isr(void)
Description	This function is called in the SCI.TEI interrupt handling. Disables the serial transmission and calls the callback function (SCI transmit end).
Arguments	None
Return Value	None

sci_rxi_isr	
Outline	Receive data full interrupt
Header	None
Declaration	static void sci_rxi_isr(void)
Description	This function is called in the SCI.RXI interrupt handling. Stores the receive data. After receiving the last data, disables the serial reception and calls the callback function (SCI receive end).
Arguments	None
Return Value	None

sci_eri_isr	
Outline	Receive error interrupt
Header	None
Declaration	static void sci_eri_isr(void)
Description	This function is called in the SCI.ERI interrupt handling. Disables the serial reception and calls the callback function (SCI receive error).
Arguments	None
Return Value	None

Excep_SCIn_ERIn	
Outline	SCI.ERI interrupt handling
Header	None
Declaration	static void Excep_SCIn_ERIn(void)
Description	Performs processing for the receive error interrupt.
Arguments	None
Return Value	None

Excep_SCIn_RXIn	
Outline	SCI.RXI interrupt handling
Header	None
Declaration	static void Excep_SCIn_RXIn(void)
Description	Performs processing for the reception data full interrupt.
Arguments	None
Return Value	None

Excep_SCIn_TXIn	
Outline	SCI.TXI interrupt handling
Header	None
Declaration	static void Excep_SCIn_TXIn(void)
Description	Performs processing for the transmit data empty interrupt.
Arguments	None
Return Value	None

Excep_SCIn_TEIn	
Outline	SCI.TEI interrupt handling
Header	None
Declaration	static void Excep_SCIn_TEIn(void)
Description	Performs processing for the transmit end interrupt.
Arguments	None
Return Value	None

5.9 Flowcharts

5.9.1 Main Processing

Figure 5.5 shows the Main Processing.

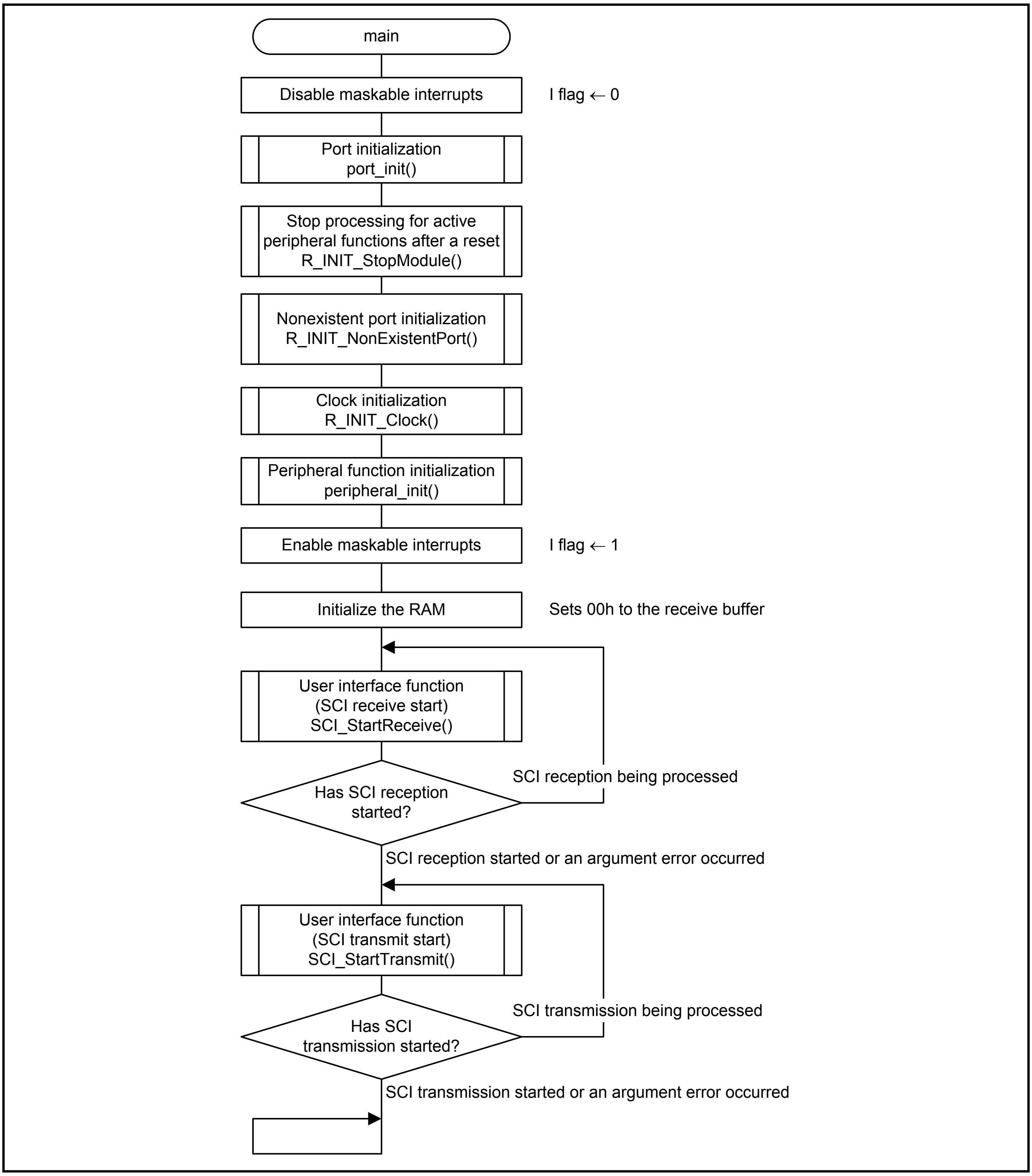


Figure 5.5 Main Processing

5.9.2 Port Initialization

Figure 5.6 shows the Port Initialization.

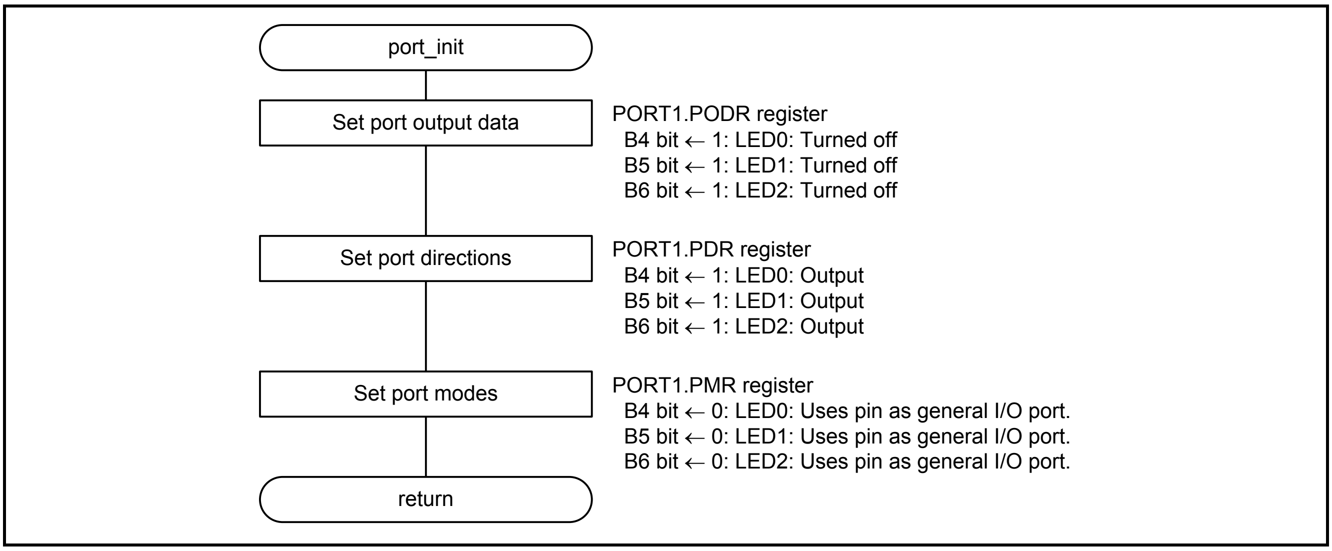


Figure 5.6 Port Initialization

5.9.3 Peripheral Function Initialization

Figure 5.7 shows the Peripheral Function Initialization.

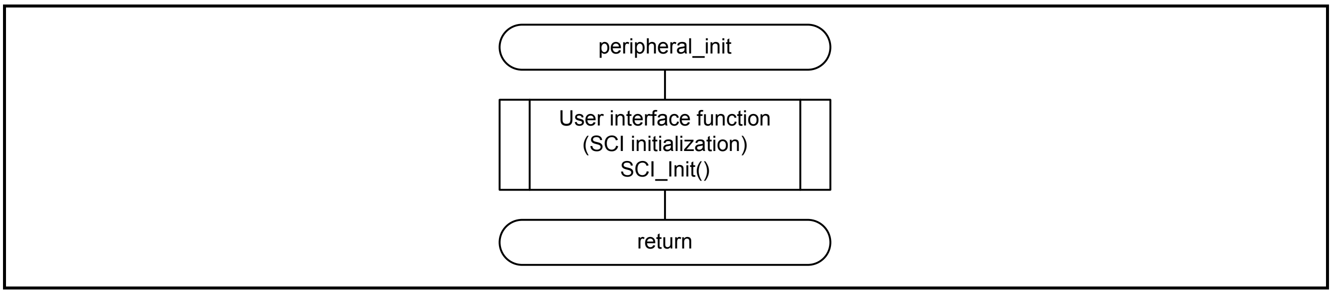


Figure 5.7 Peripheral Function Initialization

5.9.4 Callback Function (SCI Transmit End)

Figure 5.8 shows the Callback Function (SCI Transmit End).

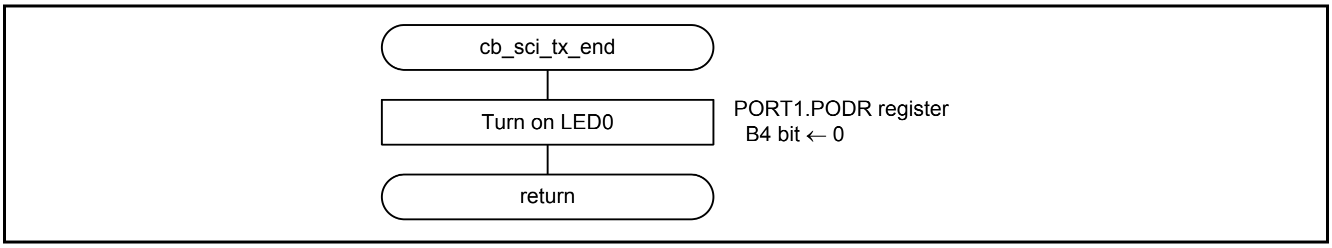


Figure 5.8 Callback Function (SCI Transmit End)

5.9.5 Callback Function (SCI Receive End)

Figure 5.9 shows the Callback Function (SCI Receive End).

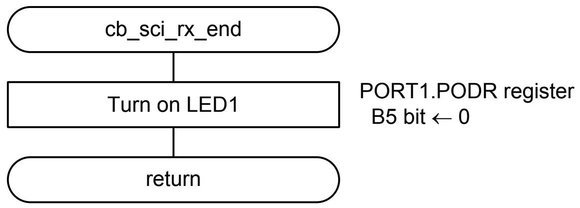
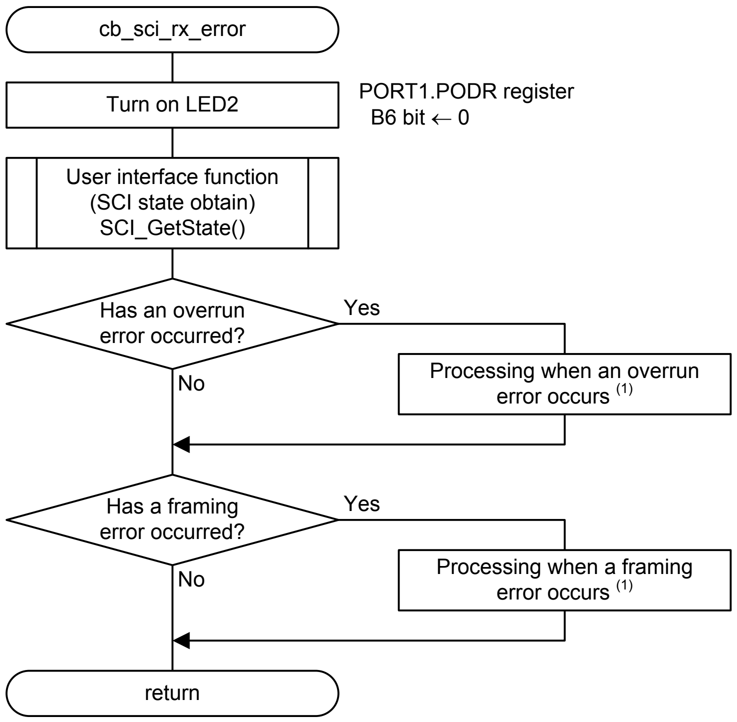


Figure 5.9 Callback Function (SCI Receive End)

5.9.6 Callback Function (SCI Receive Error)

Figure 5.10 shows the Callback Function (SCI Receive Error).



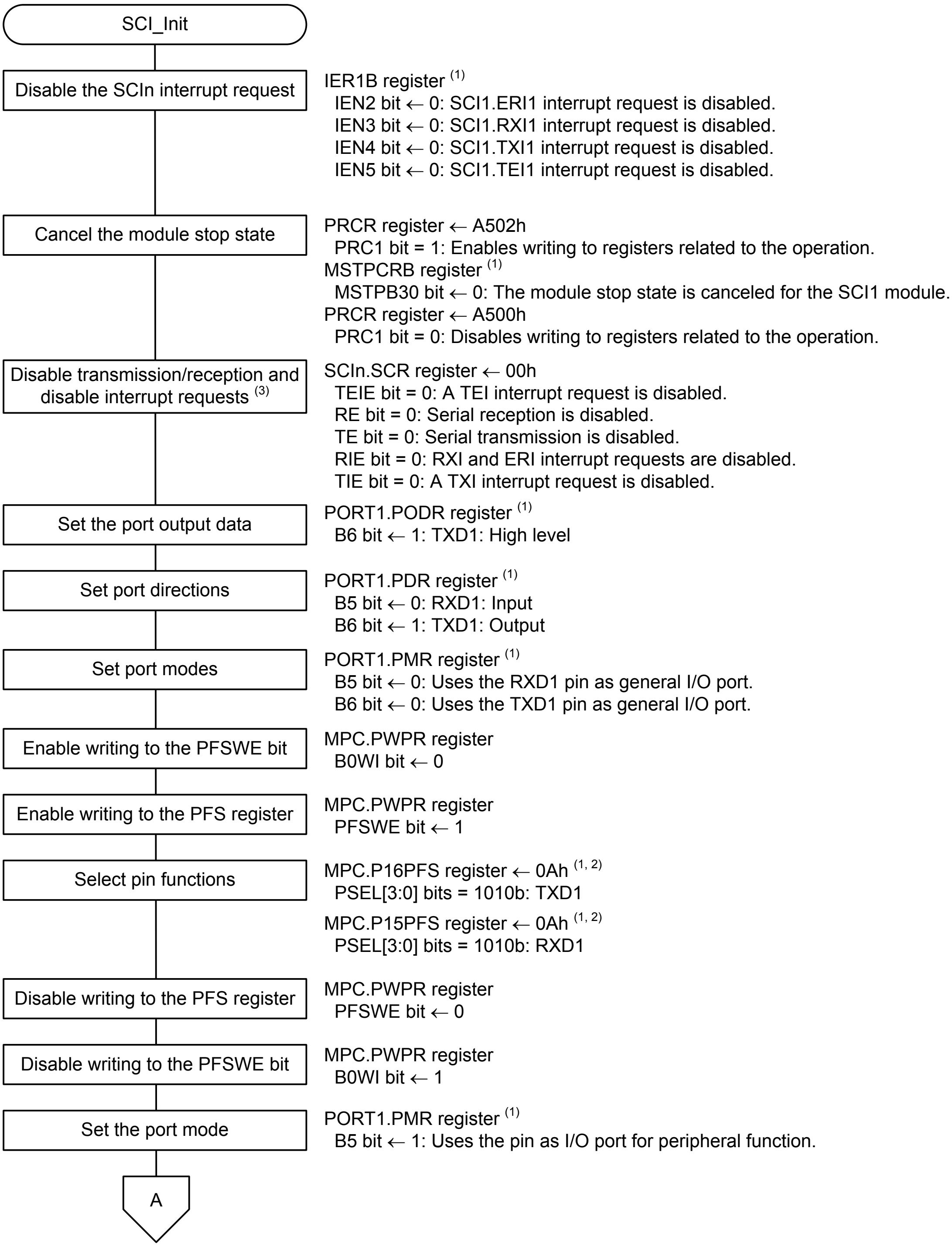
Note:

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

Figure 5.10 Callback Function (SCI Receive Error)

5.9.7 User Interface Function (SCI Initialization)

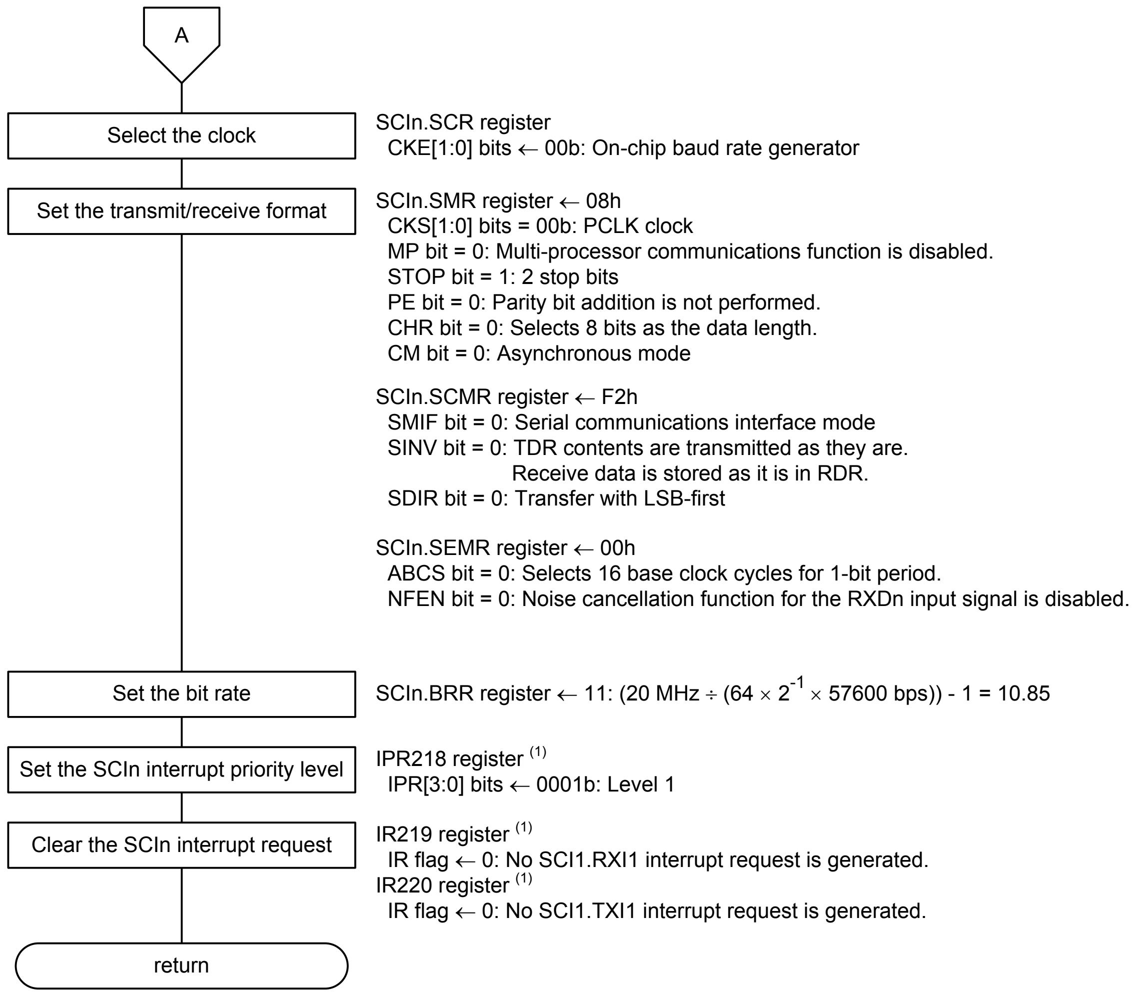
Figure 5.11 and Figure 5.12 show the User Interface Function (SCI Initialization).



Notes:

1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1).
   The I/O registers specified depend on the SCI channel selected in the configuration file.
2. The setting value described here is the value when SCI channel 1 is selected.
   The setting values specified depend on the SCI channel selected in the configuration file.
3. After writing values to bits RE, TE, RIE and TIE, read these bits to confirm that the written values can be read.

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



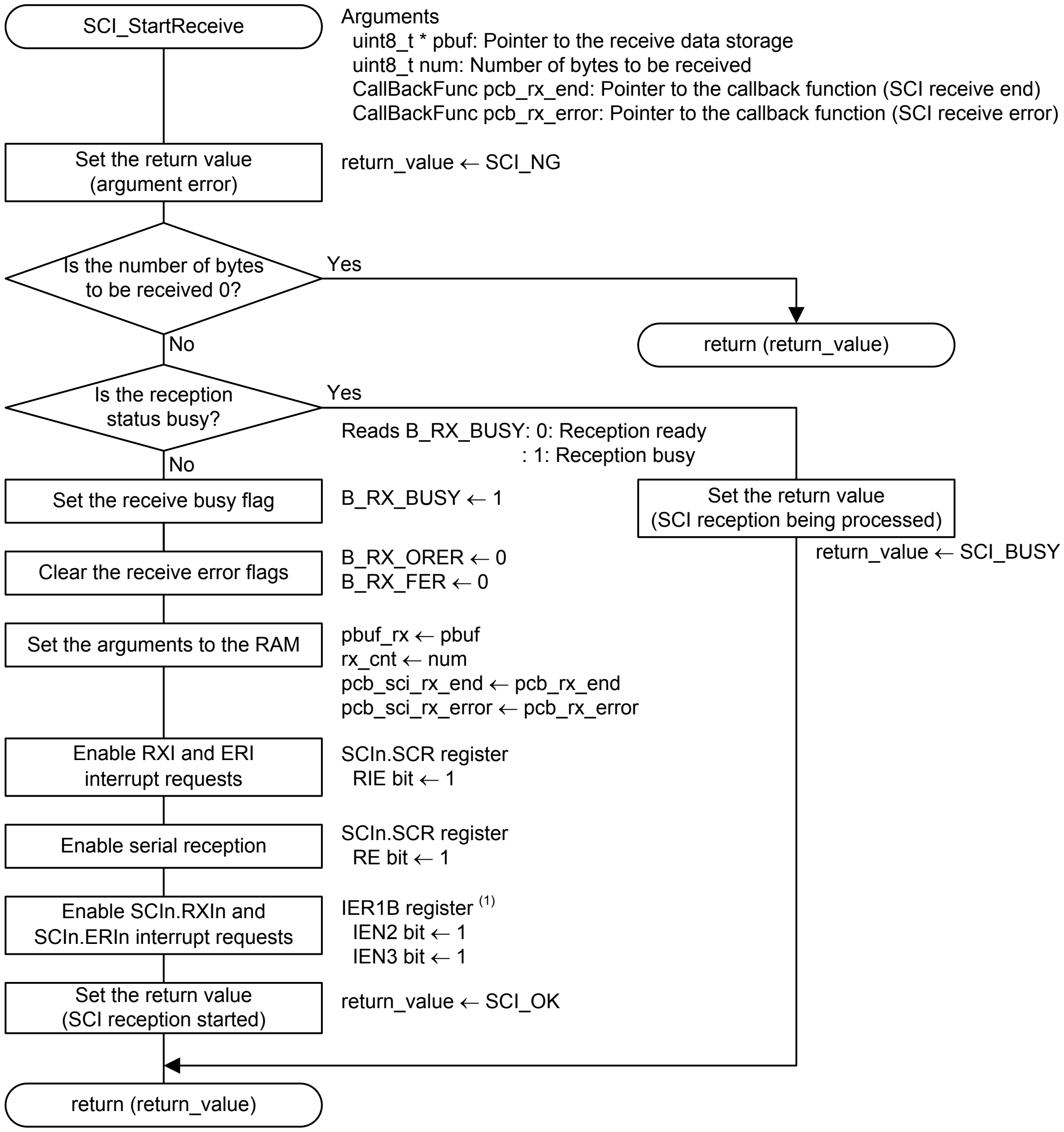
Note:

1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1).
The I/O registers specified depend on the SCI channel selected in the configuration file.

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

5.9.8 User Interface Function (SCI Receive Start)

Figure 5.13 shows the User Interface Function (SCI Receive Start).



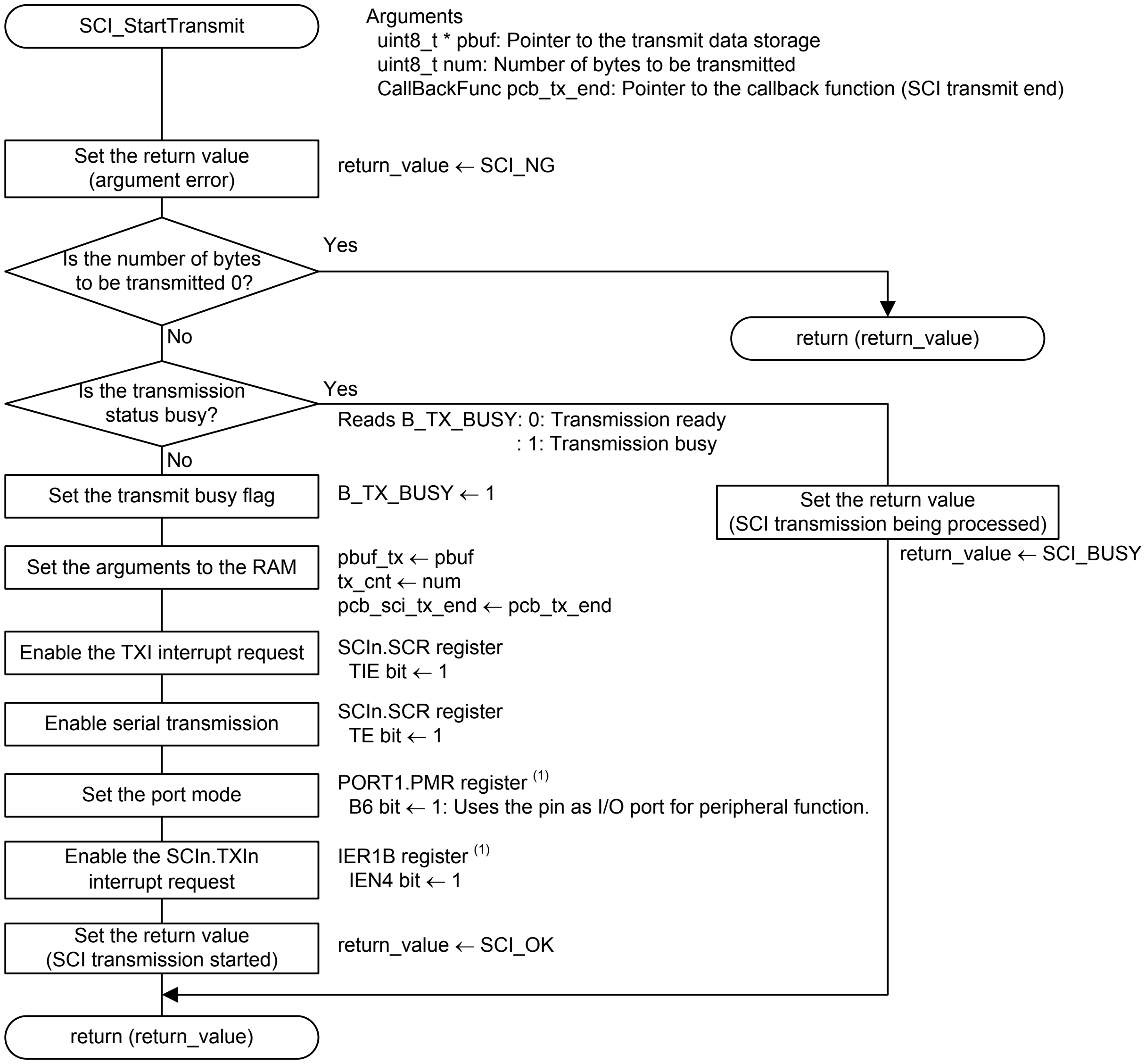
Note:

1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1). The I/O register specified depends on the SCI channel selected in the configuration file.

Figure 5.13 User Interface Function (SCI Receive Start)

5.9.9 User Interface Function (SCI Transmit Start)

Figure 5.14 shows the User Interface Function (SCI Transmit Start).



Note:
1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1).
The I/O registers specified depend on the SCI channel selected in the configuration file.

Figure 5.14 User Interface Function (SCI Transmit Start)

5.9.10 User Interface Function (SCI State Obtain)

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

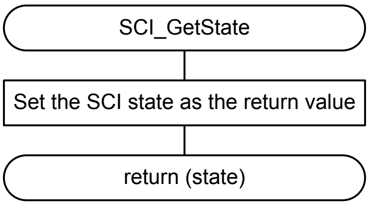
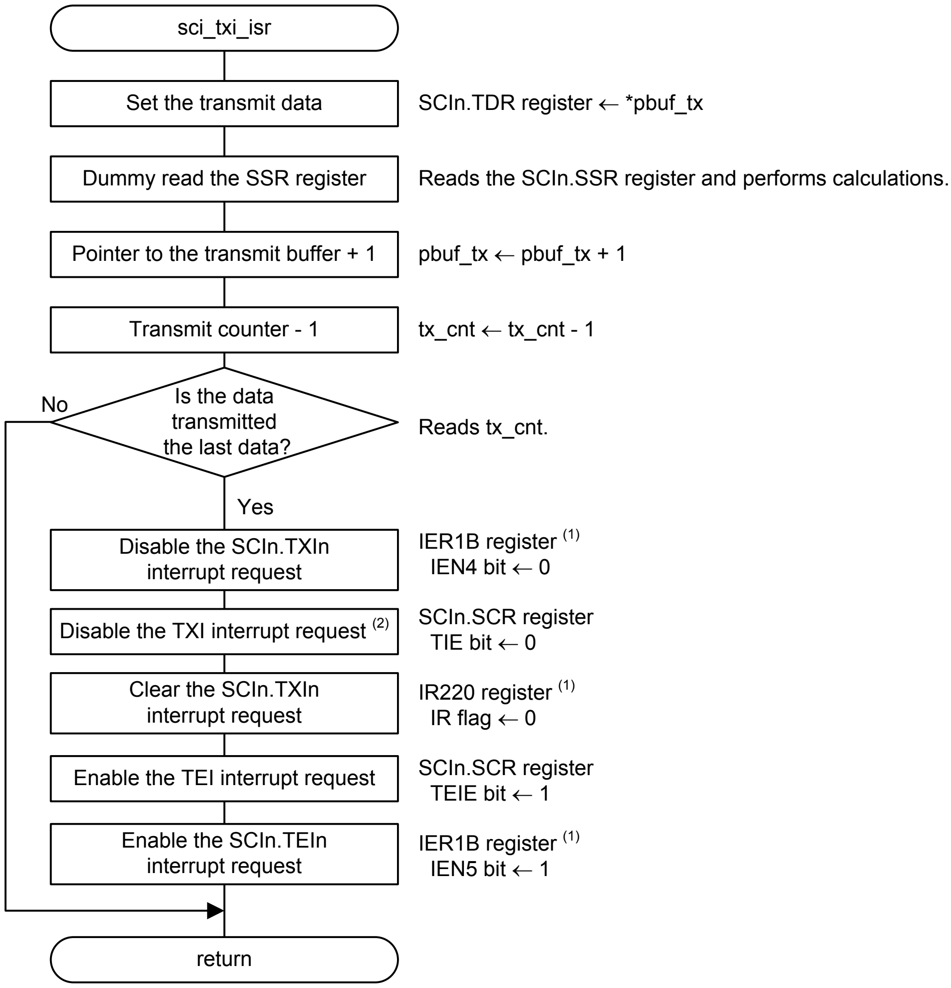


Figure 5.15 User Interface Function (SCI State Obtain)

5.9.11 Transmit Data Empty Interrupt

Figure 5.16 shows the Transmit Data Empty Interrupt.

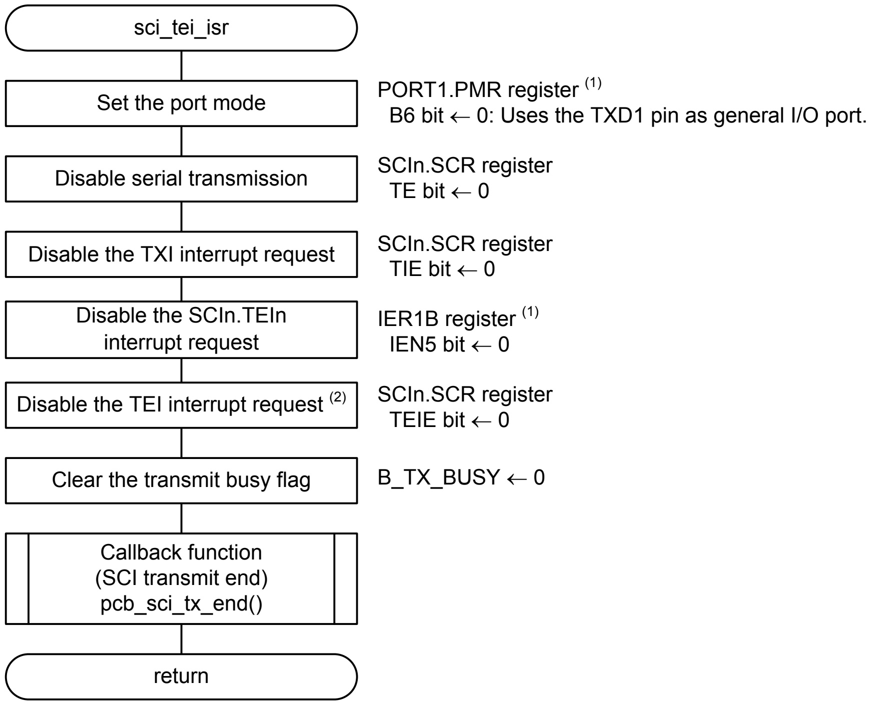


Notes:
1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1). The I/O registers specified depend on the SCI channel selected in the configuration file.
2. After writing a value to the TIE bit, read the bit to confirm that the written value can be read.

Figure 5.16 Transmit Data Empty Interrupt

5.9.12 Transmit End Interrupt

Figure 5.17 shows the Transmit End Interrupt.



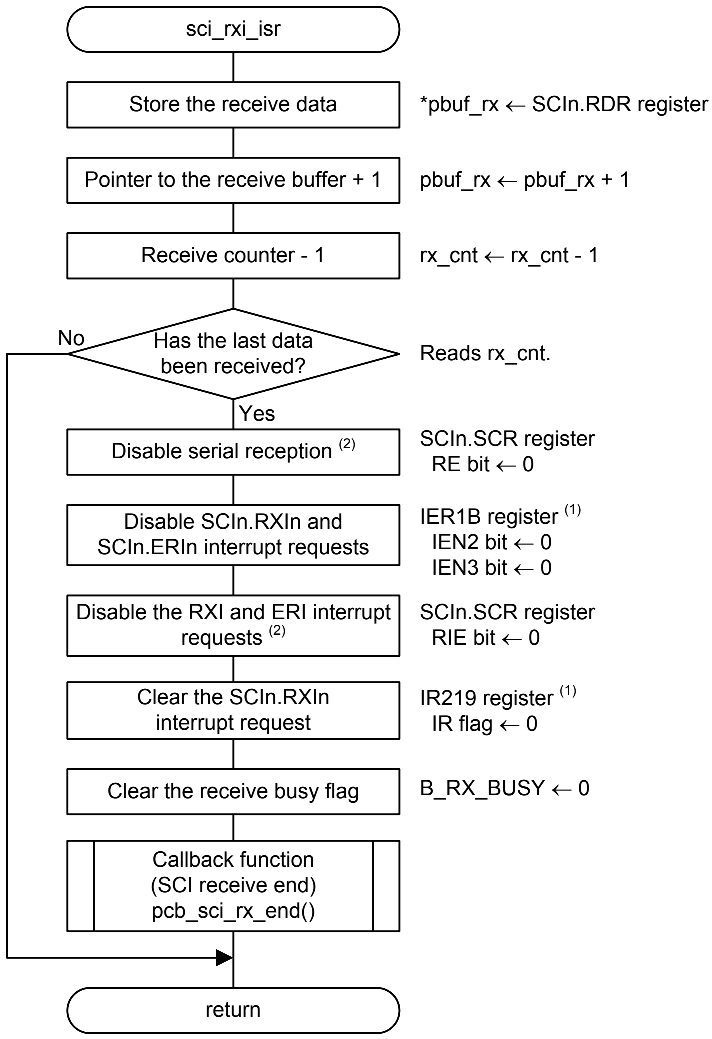
Notes:

1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1). The I/O registers specified depend on the SCI channel selected in the configuration file.
2. After writing a value to the TEIE bit, read the bit to confirm that the written value can be read.

Figure 5.17 Transmit End Interrupt

5.9.13 Receive Data Full Interrupt

Figure 5.18 shows the Receive Data Full Interrupt.



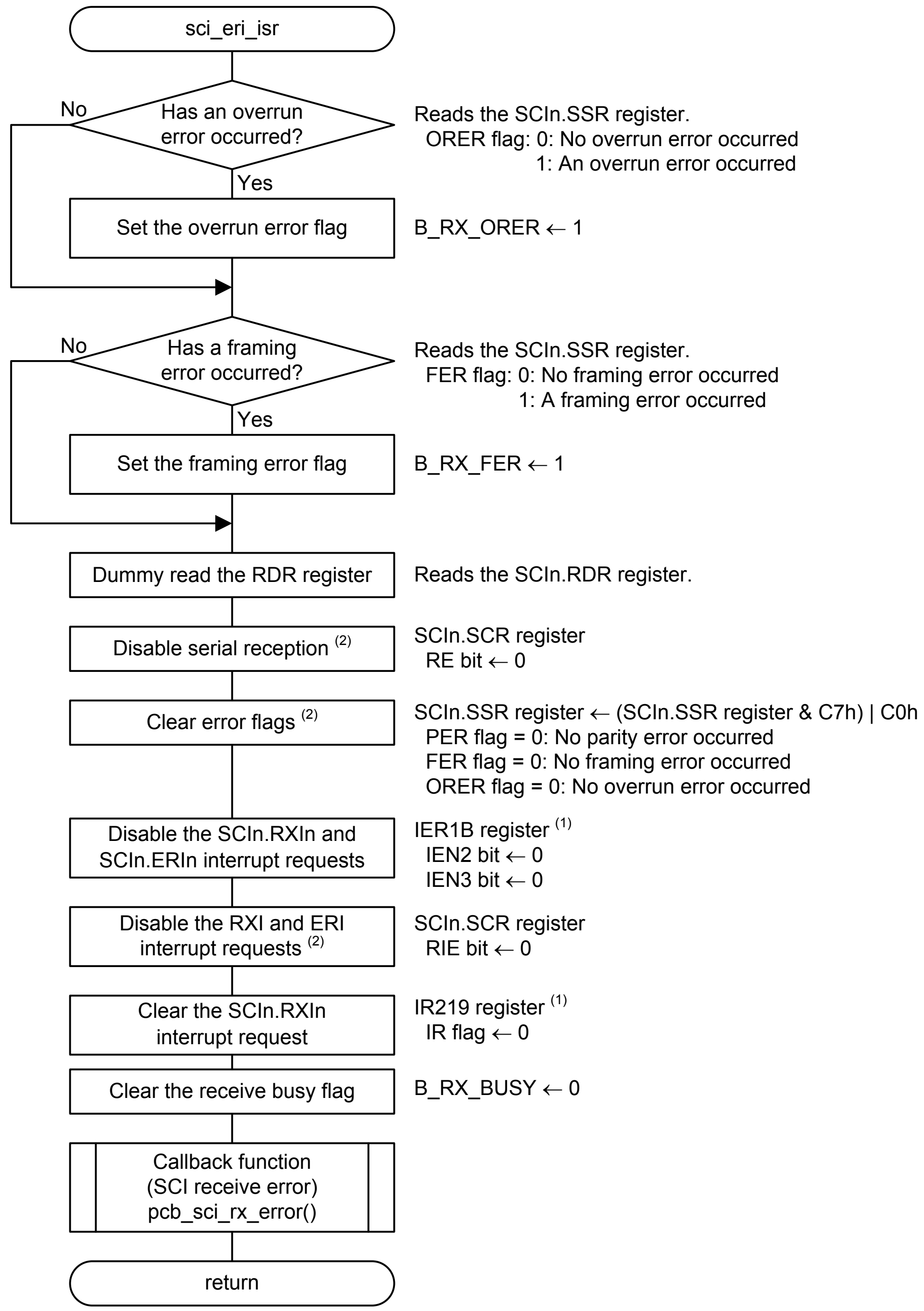
Notes:

1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1). The I/O registers specified depend on the SCI channel selected in the configuration file.
2. After writing values to bits RE and RIE, read these bits to confirm that the written values can be read.

Figure 5.18 Receive Data Full Interrupt

5.9.14 Receive Error Interrupt

Figure 5.19 shows the Receive Error Interrupt.



Notes:

1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1). The I/O registers specified depend on the SCI channel selected in the configuration file.
2. After writing values to bits RE and RIE, and flags PER, FER and ORER, read these bits and flags to confirm that the written values can be read.

Figure 5.19 Receive Error Interrupt

5.9.15 SCI.ERI Interrupt Handling

Figure 5.20 shows the SCI.ERI Interrupt Handling.

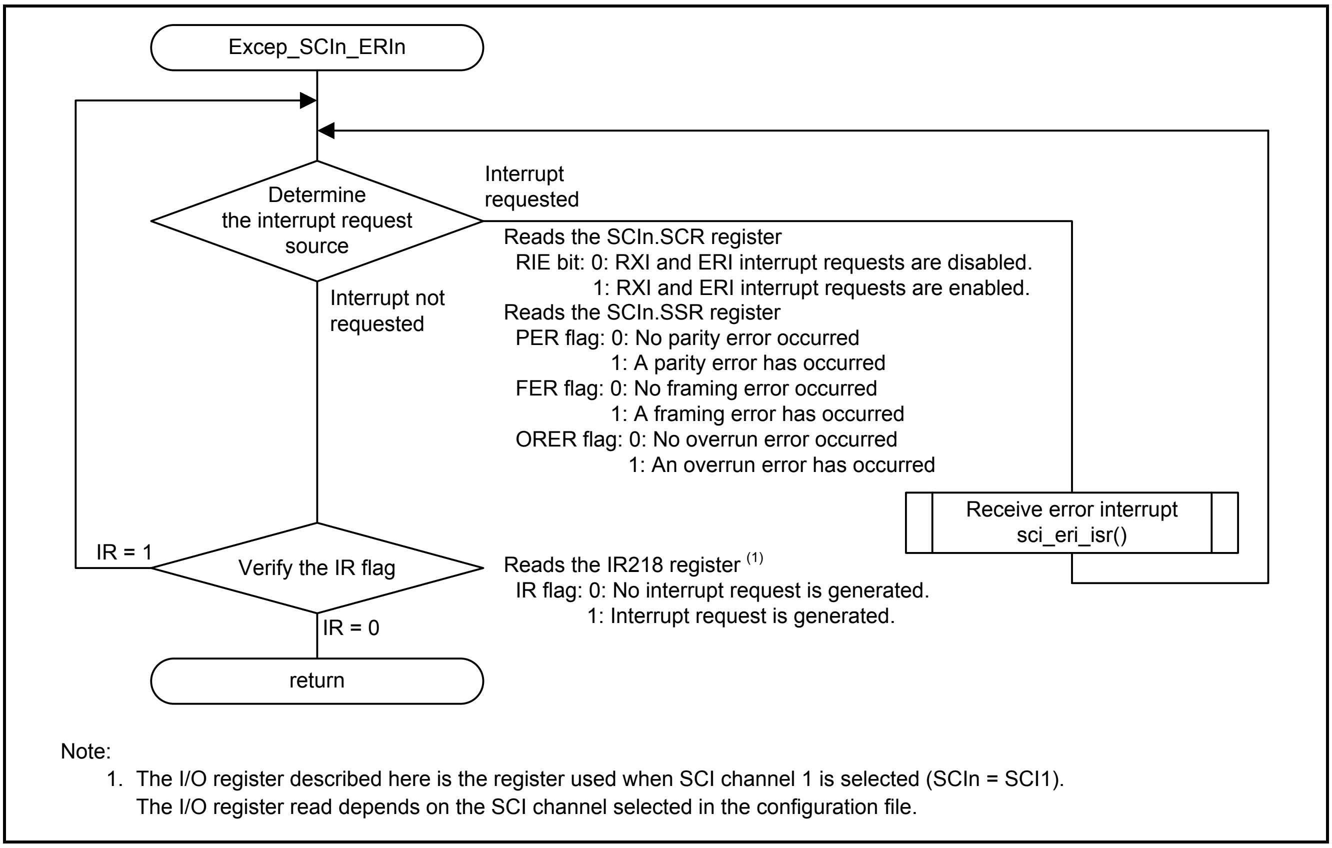


Figure 5.20 SCI.ERI Interrupt Handling

5.9.16 SCI.RXI Interrupt Handling

Figure 5.21 shows the SCI.RXI Interrupt Handling.

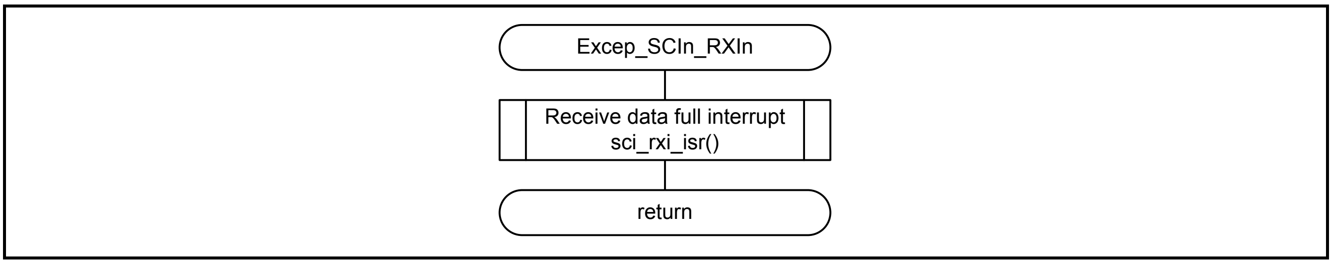


Figure 5.21 SCI.RXI Interrupt Handling

5.9.17 SCI.TXI Interrupt Handling

Figure 5.22 shows the SCI.TXI Interrupt Handling.

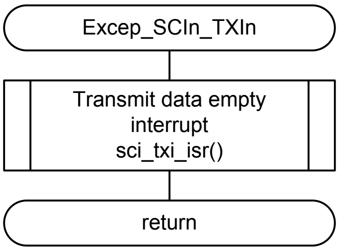
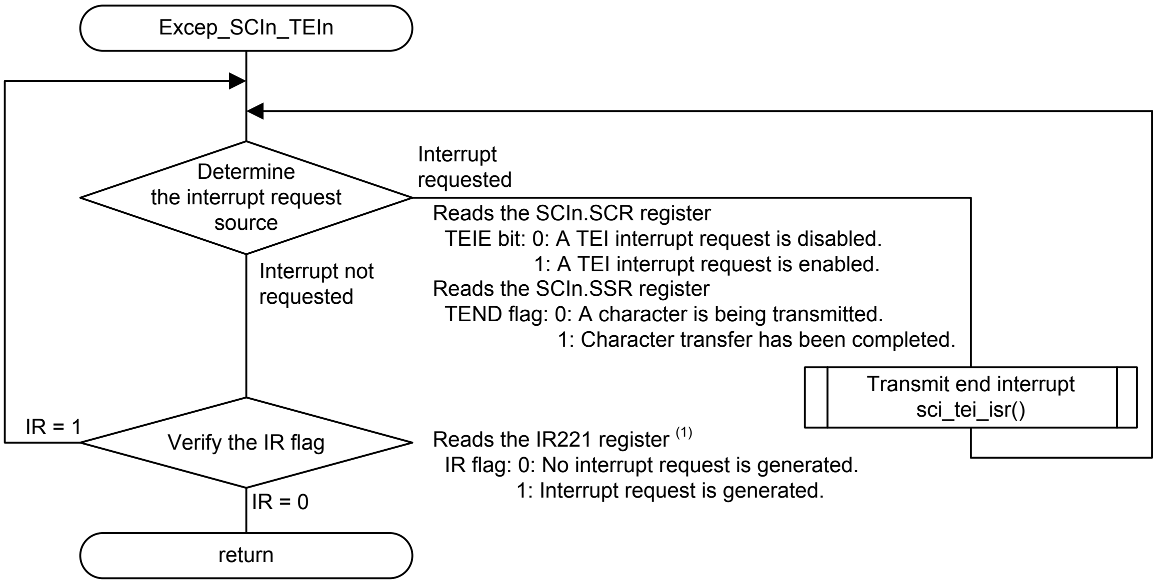


Figure 5.22 SCI.TXI Interrupt Handling

5.9.18 SCI.TEI Interrupt Handling

Figure 5.23 shows the SCI.TEI Interrupt Handling.



Note:
1. The I/O register described here is the register used when SCI channel 1 is selected (SCIn = SCI1).
The I/O register read depends on the SCI channel selected in the configuration file.

Figure 5.23 SCI.TEI Interrupt Handling

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. 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 Asynchronous Communication Using the SCI

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.

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, Jln 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