

---

## RX210, RX21A, and RX220 Groups

R01AN1423EJ0101

Rev. 1.01

July 1, 2014

### Asynchronous Communication Using the SCI

---

#### Abstract

This document describes the method to perform asynchronous serial transmission and reception using the serial communication interface (SCI) in the RX210, RX21A, and RX220 Groups.

#### Products

- RX210, RX21A, and RX220 Groups

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 .....  | 3  |
| 2. Operation Confirmation Conditions .....                         | 4  |
| 3. Reference Application Note.....                                 | 4  |
| 4. Hardware .....  | 5  |
| 4.1 Pins Used.....   | 5  |
| 5. Software .....  | 6  |
| 5.1 Operation Overview .....                                       | 7  |
| 5.1.1 Serial Transmission .....                                    | 7  |
| 5.1.2 Serial Reception .....                                       | 8  |
| 5.2 File Composition .....   | 9  |
| 5.3 Option-Setting Memory.....                                     | 9  |
| 5.4 Constants.....   | 10 |
| 5.5 Structure/Union List .....                                     | 16 |
| 5.6 Variables .....  | 16 |
| 5.7 Functions.....   | 17 |
| 5.8 Function Specifications.....                                   | 17 |
| 5.9 Flowcharts.....  | 22 |
| 5.9.1 Main Processing .....  | 22 |
| 5.9.2 Port Initialization .....                                    | 23 |
| 5.9.3 Peripheral Function Initialization.....                      | 23 |
| 5.9.4 Callback Function (SCI Transmit End) .....                   | 23 |
| 5.9.5 Callback Function (SCI Receive End) .....                    | 24 |
| 5.9.6 Callback Function (SCI Receive Error).....                   | 24 |
| 5.9.7 User Interface Function (SCI Initialization).....            | 25 |
| 5.9.8 User Interface Function (SCI Receive Start) .....            | 27 |
| 5.9.9 User Interface Function (SCI Transmit Start).....            | 28 |
| 5.9.10 User Interface Function (SCI State Obtain).....             | 29 |
| 5.9.11 Transmit Data Empty Interrupt .....                         | 29 |
| 5.9.12 Transmit End Interrupt.....                                 | 30 |
| 5.9.13 Receive Data Full Interrupt.....                            | 31 |
| 5.9.14 Receive Error Interrupt .....                               | 32 |
| 5.9.15 SCI.ERI Interrupt Handling.....                             | 33 |
| 5.9.16 SCI.RXI Interrupt Handling .....                            | 33 |
| 5.9.17 SCI.TXI Interrupt Handling .....                            | 34 |
| 5.9.18 SCI.TEI Interrupt Handling .....                            | 34 |
| 6. Applying This Application Note to the RX21A or RX220 Group..... | 35 |
| 7. Sample Code.....  | 36 |
| 8. Reference Documents.....  | 36 |

**1. Specifications**

Asynchronous serial transmission and reception are performed using the SCI.

After a reset, transmission and reception are performed only 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.

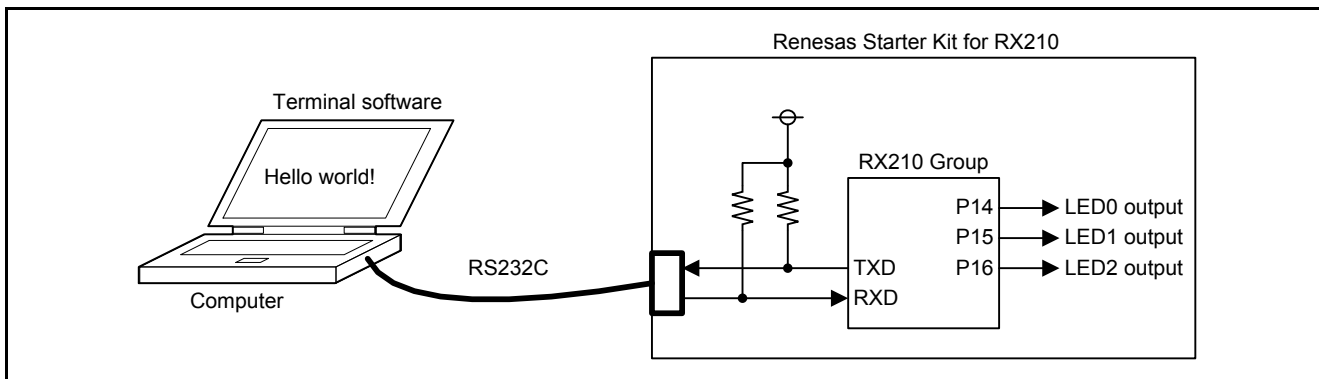
The SCI channel used is selected in the configuration file. Channel 0 (SCI0) is selected in the sample code.

- 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 0, 1, 5, 6, 8, 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                           | R5F52108ADFP (RX210 Group)  |
| Operating frequencies              | - Main clock: 20 MHz<br>- PLL: 100 MHz (main clock divided by 2 and multiplied by 10)<br>- System clock (ICLK): 50 MHz (PLL divided by 2)<br>- Peripheral module clock B (PCLKB): 25 MHz (PLL divided by 4)   |
| Operating voltage                  | 5.0 V   |
| Integrated development environment | Renesas Electronics Corporation<br>High-performance Embedded Workshop Version 4.09.01   |
| C compiler                         | Renesas Electronics Corporation<br>C/C++ Compiler Package for RX Family V.1.02 Release 01<br>Compile options<br>-cpu=rx200 -output=obj="\$ (CONFIGDIR)\\$(FILELEAF).obj" -debug -nologo<br>(The default setting is used in the integrated development environment.) |
| iodefine.h version                 | Version 1.2A  |
| Endian                             | Little endian   |
| Operating mode                     | Single-chip mode  |
| Processor mode                     | Supervisor mode   |
| Sample code version                | Version 1.00  |
| Board used                         | Renesas Starter Kit for RX210 (product part no.: R0K505210C000BE)   |
| Tool used                          | Terminal software   |

**3. Reference Application Notes**

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

- RX210 Group Initial Setting Rev. 2.00 (R01AN1002EJ)
- RX21A Group Initial Setting Rev. 1.10 (R01AN1486EJ)
- RX220 Group Initial Setting Rev. 1.10 (R01AN1494EJ)

The initial setting functions in the reference application notes are used in the sample code in this application note. The revision numbers of the reference application notes are current as of 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)                 |
| P21/RXD0  | Input  | Input pin for SCI0 receive data <sup>(1)</sup>    |
| P20/TXD0  | Output | Output pin for SCI0 transmit data <sup>(1)</sup>  |
| P15/RXD1  | Input  | Input pin for SCI1 receive data <sup>(1)</sup>    |
| P16/TXD1  | Output | Output pin for SCI1 transmit data <sup>(1)</sup>  |
| PA3/RXD5  | Input  | Input pin for SCI5 receive data <sup>(1)</sup>    |
| PA4/TXD5  | Output | Output pin for SCI5 transmit data <sup>(1)</sup>  |
| PB0/RXD6  | Input  | Input pin for SCI6 receive data <sup>(1)</sup>    |
| PB1/TXD6  | Output | Output pin for SCI6 transmit data <sup>(1)</sup>  |
| PC6/RXD8  | Input  | Input pin for SCI8 receive data <sup>(1)</sup>    |
| PC7/TXD8  | Output | Output pin for SCI8 transmit data <sup>(1)</sup>  |
| PB6/RXD9  | Input  | Input pin for SCI9 receive data <sup>(1)</sup>    |
| PB7/TXD9  | Output | Output pin for SCI9 transmit data <sup>(1)</sup>  |
| PE2/RXD12 | Input  | Input pin for SCI12 receive data <sup>(1)</sup>   |
| PE1/TXD12 | Output | Output pin for SCI12 transmit data <sup>(1)</sup> |

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 (25 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

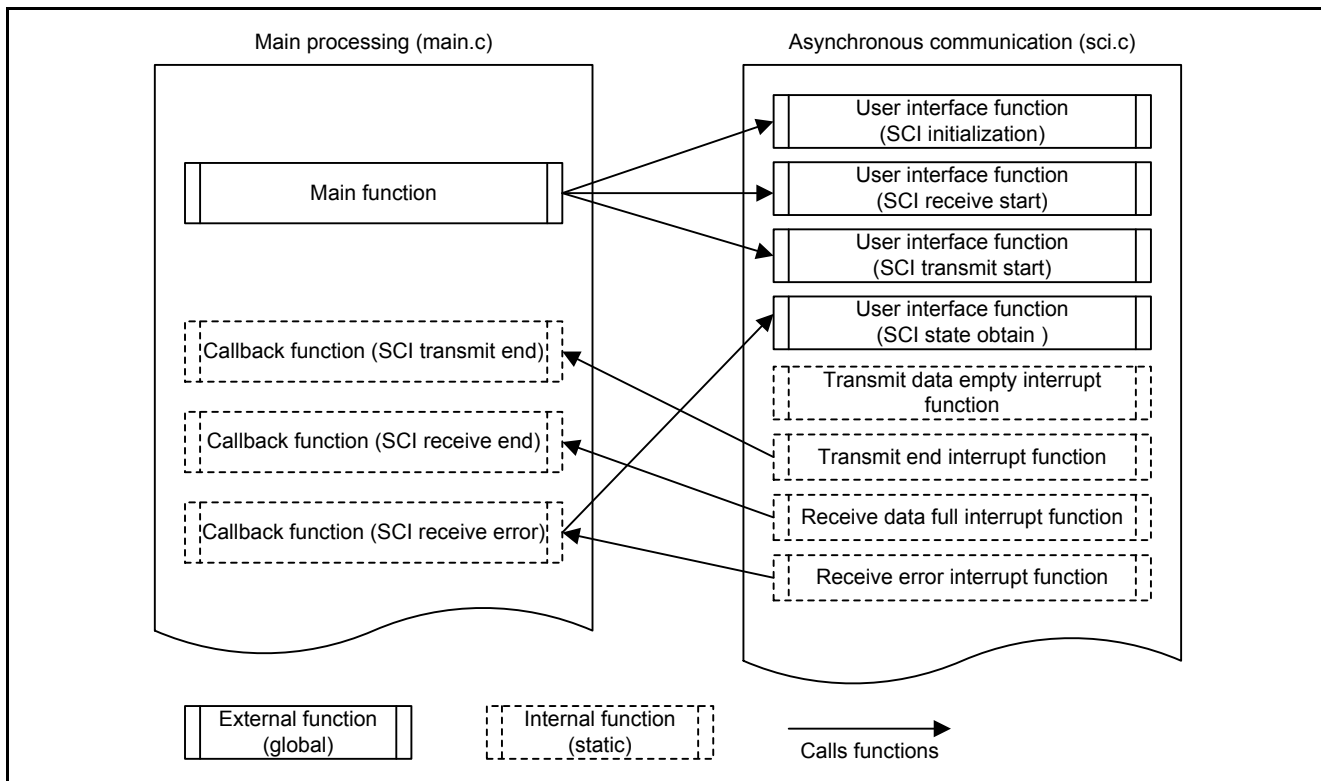


Figure 5.1 Software Configuration

## 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).

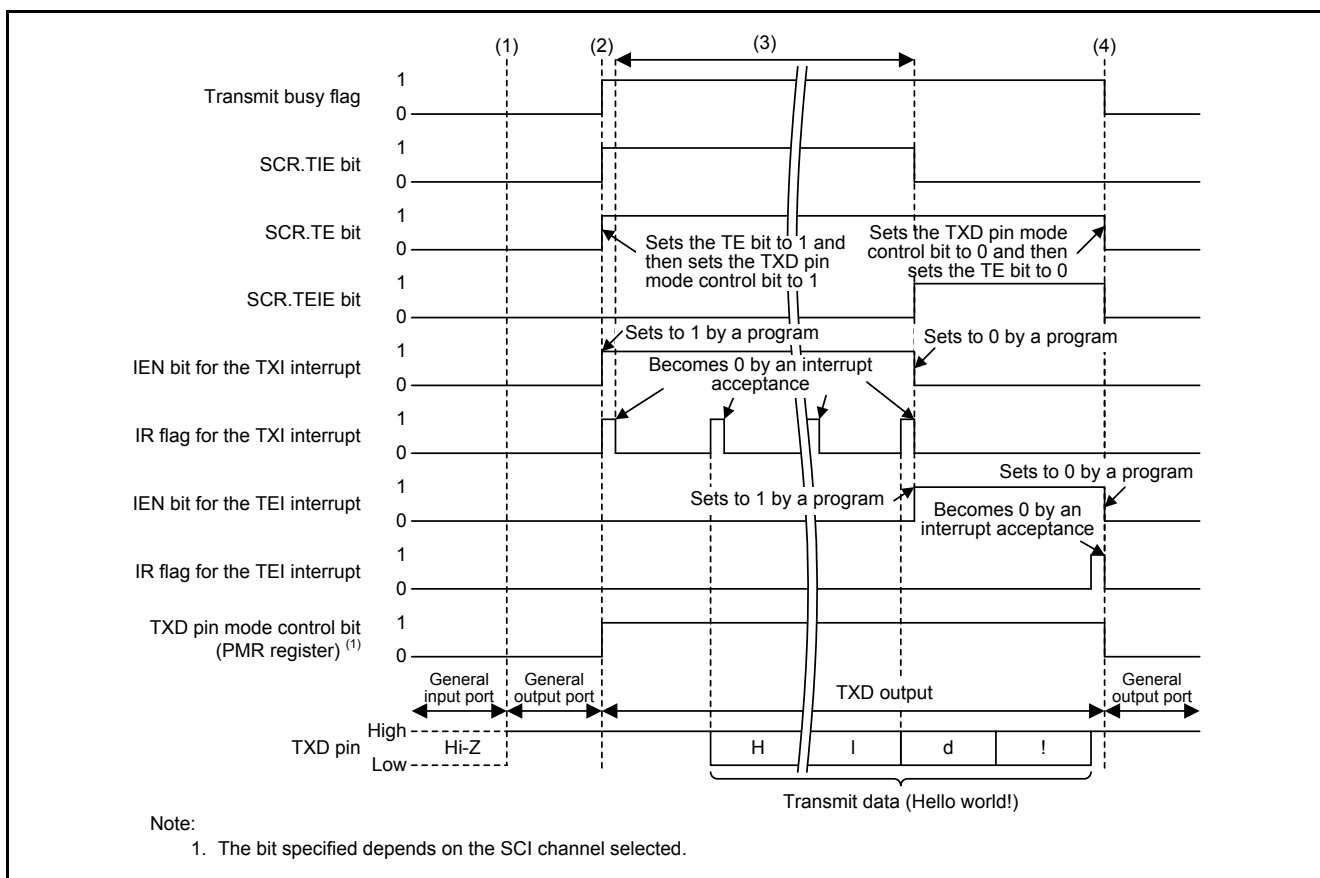


Figure 5.2 Timing of Serial Transmission

### 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).

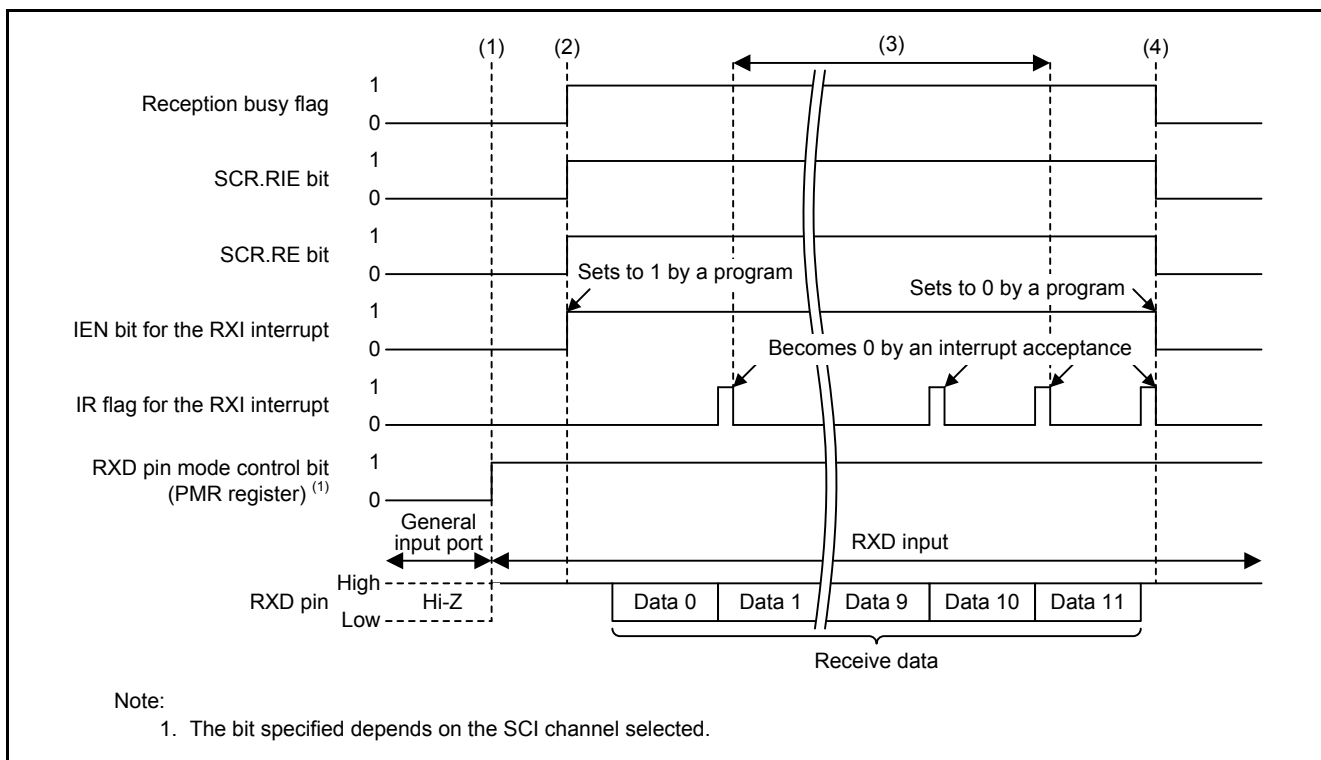


Figure 5.3 Timing of Serial Reception



## 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.<br>The WDT is stopped after a reset.                               |
| OFS1   | FFFF FF8Bh to FFFF FF88h | FFFF FFFFh    | The voltage monitor 0 reset is disabled after a reset.<br>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 5.12 list the Constants Used in the Sample Code.

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

| <b>Constant Name</b> | <b>Setting Value</b>    | <b>Contents</b>  |
|----------------------|-------------------------|--|
| 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<br>0: Transmission ready<br>1: Transmission busy              |
| SCI_B_RX_BUSY        | sci_state.bit.b_rx_busy | Receive busy flag<br>0: Reception ready<br>1: Reception busy                     |
| SCI_B_RX_ORER        | sci_state.bit.b_rx_orer | Overrun error flag<br>0: Overrun error not occurred<br>1: Overrun error occurred |
| SCI_B_RX_FER         | sci_state.bit.b_rx_fer  | Framing error flag<br>0: Framing error not occurred<br>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<br>0: Transmission ready<br>1: Transmission busy              |
| B_RX_BUSY       | state.bit.b_rx_busy | Receive busy flag<br>0: Reception ready<br>1: Reception busy                     |
| B_RX_ORER       | state.bit.b_rx_orer | Overrun error flag<br>0: Overrun error not occurred<br>1: Overrun error occurred |
| B_RX_FER        | state.bit.b_rx_fer  | Framing error flag<br>0: Framing error not occurred<br>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:<br>SCI transmit/receive start                                       |
| SCI_BUSY      | 01h           | Return value of the SCI_StartTransmit and SCI_StartReceive functions:<br>SCI transmission or reception being processed                    |
| SCI_NG        | 02h           | Return value of the SCI_StartTransmit and SCI_StartReceive functions:<br>Argument error (number of bytes to be transmitted/received is 0) |

**Table 5.6 Constants Used in the Sample Code (when SELECT\_SCI0 is Selected in sci\_cfg.h) <sup>(1)</sup>**

| Constant Name | Setting Value     | Contents   |
|---------------|-------------------|--|
| SCIn          | SCI0              | SCI channel: SCI0  |
| MSTP_SCIn     | MSTP(SCI0)        | SCI0 module stop setting bit                             |
| IPR_SCIn      | IPR(SCI0, )       | SCI0 interrupt priority level setting bit                |
| IR_SCIn_ERIn  | IR(SCI0,ERI0)     | SCI0.ERI0 interrupt status flag                          |
| IR_SCIn_RXIn  | IR(SCI0,RXI0)     | SCI0.RXI0 interrupt status flag                          |
| IR_SCIn_TXIn  | IR(SCI0,TXI0)     | SCI0.TXI0 interrupt status flag                          |
| IR_SCIn_TEIn  | IR(SCI0,TEI0)     | SCI0.TEI0 interrupt status flag                          |
| IEN_SCIn_ERIn | IEN(SCI0,ERI0)    | SCI0.ERI0 interrupt request enable bit                   |
| IEN_SCIn_RXIn | IEN(SCI0,RXI0)    | SCI0.RXI0 interrupt request enable bit                   |
| IEN_SCIn_TXIn | IEN(SCI0,TXI0)    | SCI0.TXI0 interrupt request enable bit                   |
| IEN_SCIn_TEIn | IEN(SCI0,TEI0)    | SCI0.TEI0 interrupt request enable bit                   |
| RXDn_PDR      | PORT2.PDR.BIT.B1  | P21 I/O select bit                                       |
| RXDn_PMR      | PORT2.PMR.BIT.B1  | P21 pin mode control bit                                 |
| RXDnPFS       | P21PFS            | P21 pin function control register                        |
| TXDn_PODR     | PORT2.PODR.BIT.B0 | P20 output data store bit                                |
| TXDn_PDR      | PORT2.PDR.BIT.B0  | P20 I/O select bit                                       |
| TXDn_PMR      | PORT2.PMR.BIT.B0  | P20 pin mode control bit                                 |
| TXDnPFS       | P20PFS            | P20 pin function control register                        |
| PSEL_SETTING  | 0Ah               | Setting value of the pin function select bit: RXD0, TXD0 |

Note:

1. SCI0 is not available in 48-pin and 64-pin products.

**Table 5.7 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 5.8 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 5.9 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 5.10 Constants Used in the Sample Code (when SELECT\_SCI8 is Selected in sci\_cfg.h)**

| Constant Name | Setting Value     | Contents   |
|---------------|-------------------|--|
| SCIn          | SCI8              | SCI channel: SCI8  |
| MSTP_SCIn     | MSTP(SCI8)        | SCI8 module stop setting bit                             |
| IPR_SCIn      | IPR(SCI8, )       | SCI8 interrupt priority level setting bit                |
| IR_SCIn_ERIn  | IR(SCI8,ERI8)     | SCI8.ERI8 interrupt status flag                          |
| IR_SCIn_RXIn  | IR(SCI8,RXI8)     | SCI8.RXI8 interrupt status flag                          |
| IR_SCIn_TXIn  | IR(SCI8,TXI8)     | SCI8.TXI8 interrupt status flag                          |
| IR_SCIn_TEIn  | IR(SCI8,TEI8)     | SCI8.TEI8 interrupt status flag                          |
| IEN_SCIn_ERIn | IEN(SCI8,ERI8)    | SCI8.ERI8 interrupt request enable bit                   |
| IEN_SCIn_RXIn | IEN(SCI8,RXI8)    | SCI8.RXI8 interrupt request enable bit                   |
| IEN_SCIn_TXIn | IEN(SCI8,TXI8)    | SCI8.TXI8 interrupt request enable bit                   |
| IEN_SCIn_TEIn | IEN(SCI8,TEI8)    | SCI8.TEI8 interrupt request enable bit                   |
| RXDn_PDR      | PORTC.PDR.BIT.B6  | PC6 I/O select bit                                       |
| RXDn_PMR      | PORTC.PMR.BIT.B6  | PC6 pin mode control bit                                 |
| RXDnPFS       | PC6PFS            | PC6 pin function control register                        |
| TXDn_PODR     | PORTC.PODR.BIT.B7 | PC7 output data store bit                                |
| TXDn_PDR      | PORTC.PDR.BIT.B7  | PC7 I/O select bit                                       |
| TXDn_PMR      | PORTC.PMR.BIT.B7  | PC7 pin mode control bit                                 |
| TXDnPFS       | PC7PFS            | PC7 pin function control register                        |
| PSEL_SETTING  | 0Ah               | Setting value of the pin function select bit: RXD8, TXD8 |

**Table 5.11 Constants Used in the Sample Code (when SELECT\_SCI9 is Selected in sci\_cfg.h) <sup>(1)</sup>**

| 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 5.12 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 5.13 lists the static Variables.

Table 5.13 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  |
| static uint8_t         | tx_cnt           | Transmit counter                | sci_txi_isr  |
| static uint8_t *       | pbuf_rx          | Pointer to the reception buffer | SCI_StartReceive   |
| static uint8_t         | rx_cnt           | Receive counter                 | sci_rxi_isr  |
| static sci_state_t     | state            | SCI state                       | SCI_StartReceive<br>SCI_StartTransmit<br>SCI_GetState<br>sci_tei_isr<br>sci_rxi_isr<br>sci_eri_isr |



## 5.7 Functions

Table 5.14 lists the Functions Used in the Sample Code.

**Table 5.14 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                |  |
| <b>Outline</b>      | Main processing  |
| <b>Header</b>       | None   |
| <b>Declaration</b>  | void main(void)  |
| <b>Description</b>  | After initialization, starts SCI reception and then starts transmission. |
| <b>Arguments</b>    | None   |
| <b>Return Value</b> | None   |
| port_init           |  |
| <b>Outline</b>      | Port initialization  |
| <b>Header</b>       | None   |
| <b>Declaration</b>  | static void port_init(void)  |
| <b>Description</b>  | Initializes the ports.   |
| <b>Arguments</b>    | None   |
| <b>Return Value</b> | None   |

|                               |  |
|-------------------------------|--|
| <b>R_INIT_StopModule</b>      |  |
| <b>Outline</b>                | Stop processing for active peripheral functions after a reset  |
| <b>Header</b>                 | r_init_stop_module.h   |
| <b>Declaration</b>            | void R_INIT_StopModule(void)   |
| <b>Description</b>            | Configures the setting to enter the module stop state.   |
| <b>Arguments</b>              | None   |
| <b>Return Value</b>           | None   |
| <b>Remarks</b>                | Transition to the module stop state is not performed in the sample code. For details on this function, refer to the Initial Setting application note for the product used.   |
| <b>R_INIT_NonExistentPort</b> |  |
| <b>Outline</b>                | Nonexistent port initialization  |
| <b>Header</b>                 | r_init_non_existent_port.h   |
| <b>Declaration</b>            | void R_INIT_NonExistentPort(void)  |
| <b>Description</b>            | Initializes port direction registers for ports that do not exist in products with less than 100 pins.  |
| <b>Arguments</b>              | None   |
| <b>Return Value</b>           | None   |
| <b>Remarks</b>                | 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.<br>For details on this function, refer to the Initial Setting application note for the product used. |
| <b>R_INIT_Clock</b>           |  |
| <b>Outline</b>                | Clock initialization   |
| <b>Header</b>                 | r_init_clock.h   |
| <b>Declaration</b>            | void R_INIT_Clock(void)  |
| <b>Description</b>            | Initializes the clock.   |
| <b>Arguments</b>              | None   |
| <b>Return Value</b>           | None   |
| <b>Remarks</b>                | The sample code selects processing which uses PLL as the system clock without using the sub-clock.<br>For details on this function, refer to the Initial Setting application note for the product used.  |
| <b>peripheral_init</b>        |  |
| <b>Outline</b>                | Peripheral function initialization   |
| <b>Header</b>                 | None   |
| <b>Declaration</b>            | static void peripheral_init (void)   |
| <b>Description</b>            | Initializes peripheral functions used.   |
| <b>Arguments</b>              | None   |
| <b>Return Value</b>           | None   |

|                         |  |
|-------------------------|--|
| <b>cb_sci_tx_end</b>    |  |
| <b>Outline</b>          | Callback function (SCI transmit end)   |
| <b>Header</b>           | None   |
| <b>Declaration</b>      | static void cb_sci_tx_end(void)  |
| <b>Description</b>      | This function is called when the SCI transmission has been completed.  |
| <b>Arguments</b>        | None   |
| <b>Return Value</b>     | None   |
| <b>cb_sci_rx_end</b>    |  |
| <b>Outline</b>          | Callback function (SCI receive end)  |
| <b>Header</b>           | None   |
| <b>Declaration</b>      | static void cb_sci_rx_end(void)  |
| <b>Description</b>      | This function is called when the SCI reception has been completed.   |
| <b>Arguments</b>        | None   |
| <b>Return Value</b>     | None   |
| <b>cb_sci_rx_error</b>  |  |
| <b>Outline</b>          | Callback function (SCI receive error)  |
| <b>Header</b>           | None   |
| <b>Declaration</b>      | static void cb_sci_rx_error(void)  |
| <b>Description</b>      | This function is called when the SCI receive error occurs.   |
| <b>Arguments</b>        | None   |
| <b>Return Value</b>     | None   |
| <b>Remarks</b>          | Error processing is not performed in the sample code. Add a program as required.   |
| <b>SCI_Init</b>         |  |
| <b>Outline</b>          | User interface function (SCI initialization)   |
| <b>Header</b>           | sci.h  |
| <b>Declaration</b>      | void SCI_Init(void)  |
| <b>Description</b>      | Initializes the SCI.   |
| <b>Arguments</b>        | None   |
| <b>Return Value</b>     | None   |
| <b>SCI_StartReceive</b> |  |
| <b>Outline</b>          | User interface function (SCI receive start)  |
| <b>Header</b>           | sci.h  |
| <b>Declaration</b>      | uint8_t SCI_StartReceive(uint8_t * pbuf, uint8_t num, CallbackFunc pcb_rx_end, CallbackFunc pcb_rx_error)  |
| <b>Description</b>      | Starts SCI reception.  |
| <b>Arguments</b>        | uint8_t * pbuf: Pointer to the receive data storage<br>uint8_t num: Number of bytes to be received<br>CallbackFunc pcb_rx_end: Pointer to the callback function (SCI receive end)<br>CallbackFunc pcb_rx_error: Pointer to the callback function (SCI receive error) |
| <b>Return Value</b>     | SCI_NG: Argument error (number of bytes to be received is 0)<br>SCI_BUSY: SCI reception being processed<br>SCI_OK: SCI reception started   |

---

## SCI\_StartTransmit

---

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

---

## SCI\_GetState

---

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

---

## sci\_txi\_isr

---

|                     |  |
|---------------------|--|
| <b>Outline</b>      | Transmit data empty interrupt  |
| <b>Header</b>       | None   |
| <b>Declaration</b>  | static void sci_txi_isr(void)  |
| <b>Description</b>  | 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. |
| <b>Arguments</b>    | None   |
| <b>Return Value</b> | None   |

---

## sci\_tej\_isr

---

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

|                     |   |
|---------------------|---|
| sci_rxi_isr         |   |
| <b>Outline</b>      | Receive data full interrupt   |
| <b>Header</b>       | None  |
| <b>Declaration</b>  | static void sci_rxi_isr(void)   |
| <b>Description</b>  | 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). |
| <b>Arguments</b>    | None  |
| <b>Return Value</b> | None  |
| sci_eri_isr         |   |
| <b>Outline</b>      | Receive error interrupt   |
| <b>Header</b>       | None  |
| <b>Declaration</b>  | static void sci_eri_isr(void)   |
| <b>Description</b>  | This function is called in the SCI.ERI interrupt handling. Disables the serial reception and calls the callback function (SCI receive error).   |
| <b>Arguments</b>    | None  |
| <b>Return Value</b> | None  |
| Excep_SCIn_ERIn     |   |
| <b>Outline</b>      | SCI.ERI interrupt handling  |
| <b>Header</b>       | None  |
| <b>Declaration</b>  | static void Excep_SCIn_ERIn(void)   |
| <b>Description</b>  | Performs processing for the receive error interrupt.  |
| <b>Arguments</b>    | None  |
| <b>Return Value</b> | None  |
| Excep_SCIn_RXIn     |   |
| <b>Outline</b>      | SCI.RXI interrupt handling  |
| <b>Header</b>       | None  |
| <b>Declaration</b>  | static void Excep_SCIn_RXIn(void)   |
| <b>Description</b>  | Performs processing for the reception data full interrupt.  |
| <b>Arguments</b>    | None  |
| <b>Return Value</b> | None  |
| Excep_SCIn_TXIn     |   |
| <b>Outline</b>      | SCI.TXI interrupt handling  |
| <b>Header</b>       | None  |
| <b>Declaration</b>  | static void Excep_SCIn_TXIn(void)   |
| <b>Description</b>  | Performs processing for the transmit data empty interrupt.  |
| <b>Arguments</b>    | None  |
| <b>Return Value</b> | None  |
| Excep_SCIn_TEIn     |   |
| <b>Outline</b>      | SCI.TEI interrupt handling  |
| <b>Header</b>       | None  |
| <b>Declaration</b>  | static void Excep_SCIn_TEIn(void)   |
| <b>Description</b>  | Performs processing for the transmit end interrupt.   |
| <b>Arguments</b>    | None  |
| <b>Return Value</b> | None  |

## 5.9 Flowcharts

### 5.9.1 Main Processing

Figure 5.5 shows the Main Processing.

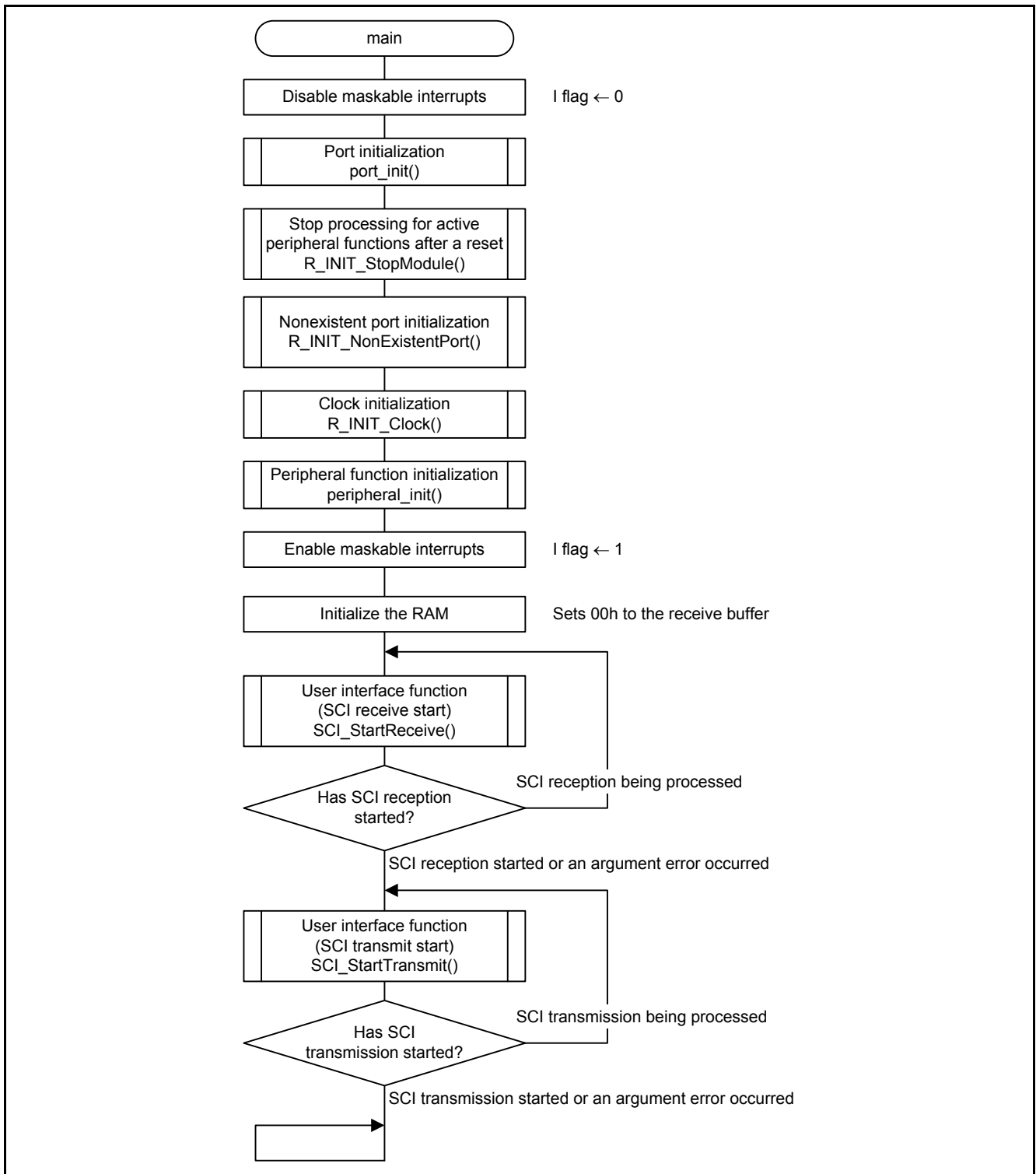


Figure 5.5 Main Processing

### 5.9.2 Port Initialization

Figure 5.6 shows the Port Initialization.

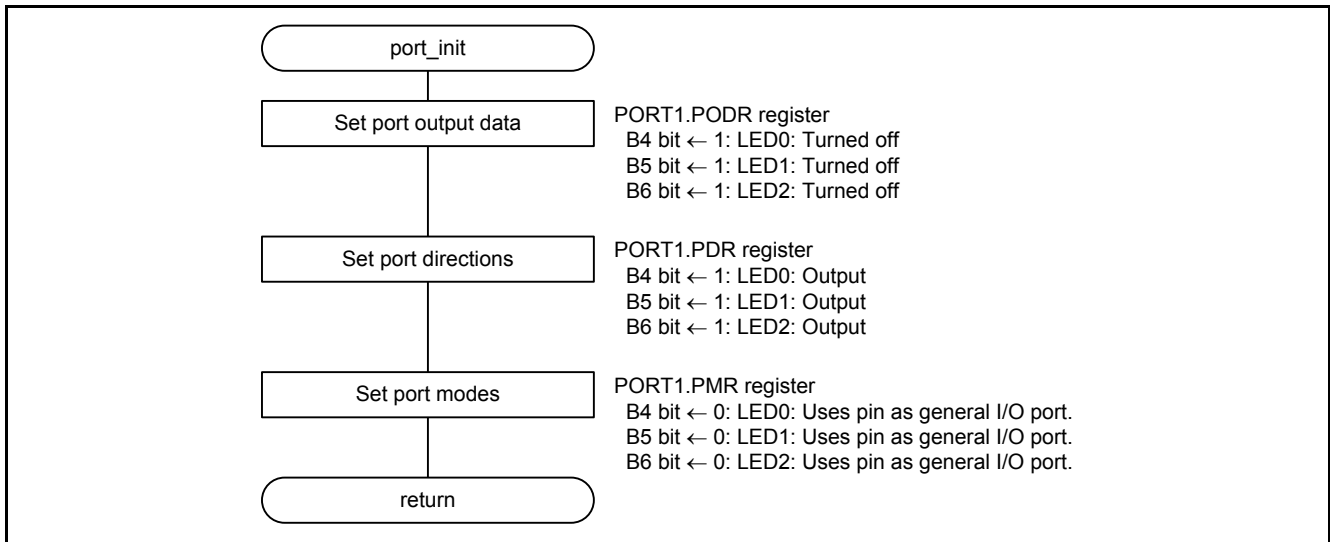


Figure 5.6 Port Initialization

### 5.9.3 Peripheral Function Initialization

Figure 5.7 shows the Peripheral Function Initialization.

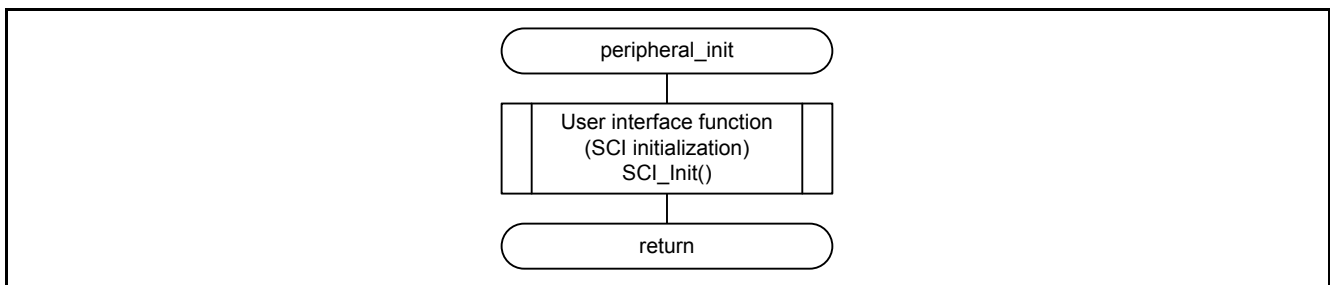


Figure 5.7 Peripheral Function Initialization

### 5.9.4 Callback Function (SCI Transmit End)

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

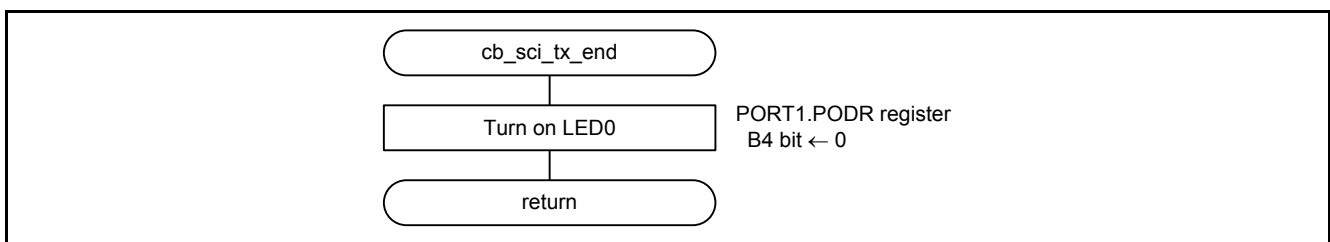
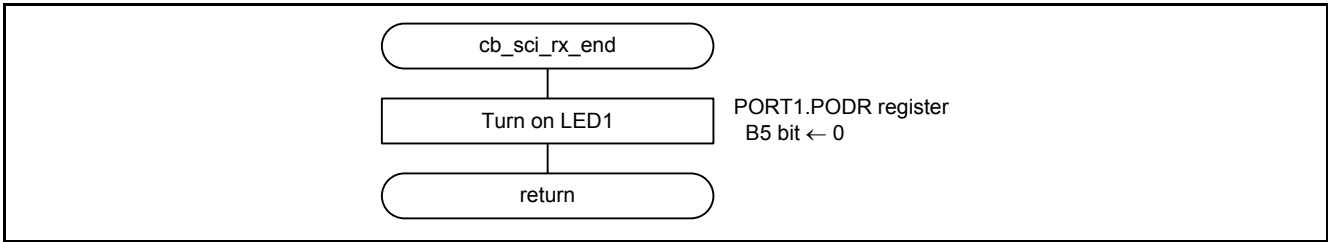


Figure 5.8 Callback Function (SCI Transmit End)

**5.9.5 Callback Function (SCI Receive End)**

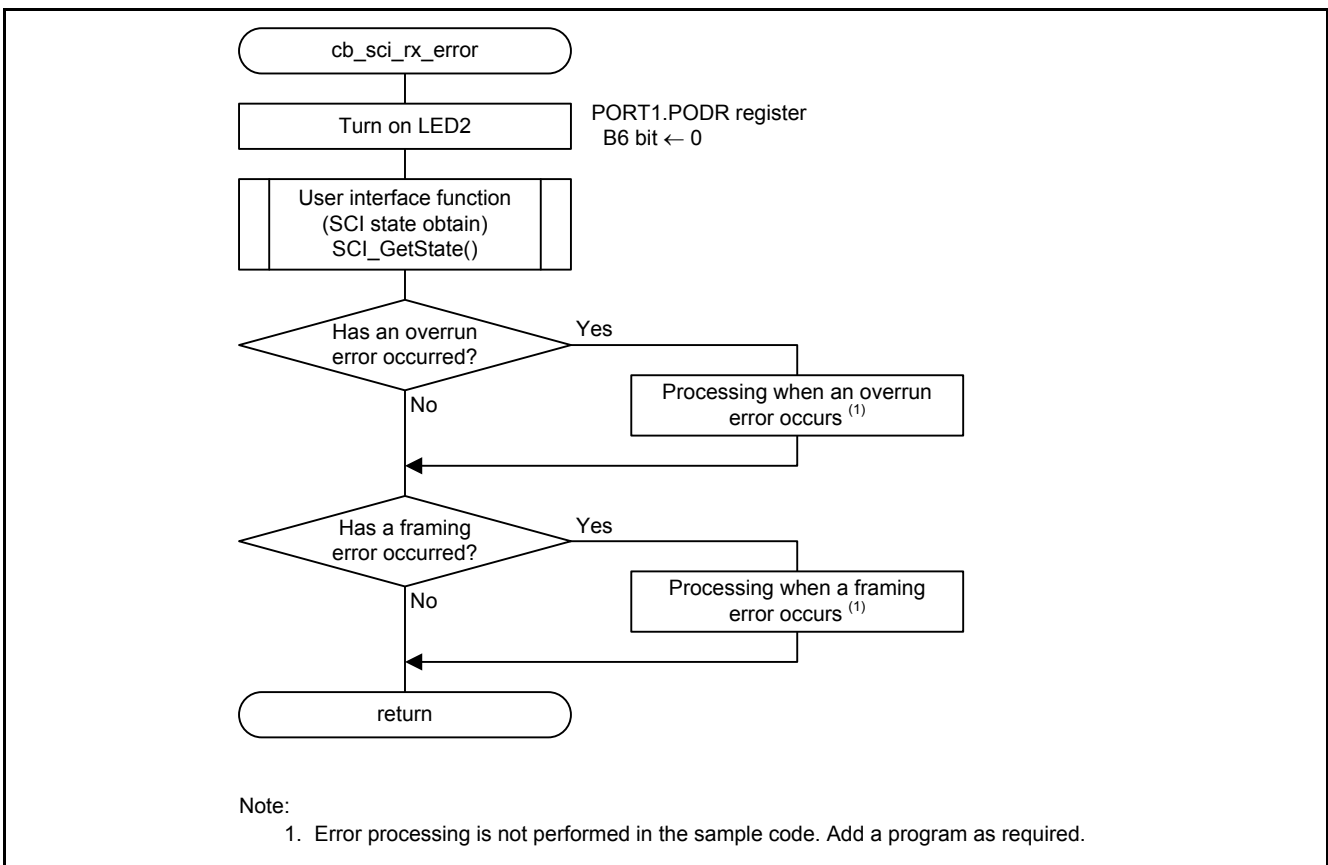
Figure 5.9 shows the Callback Function (SCI Receive End).



**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).

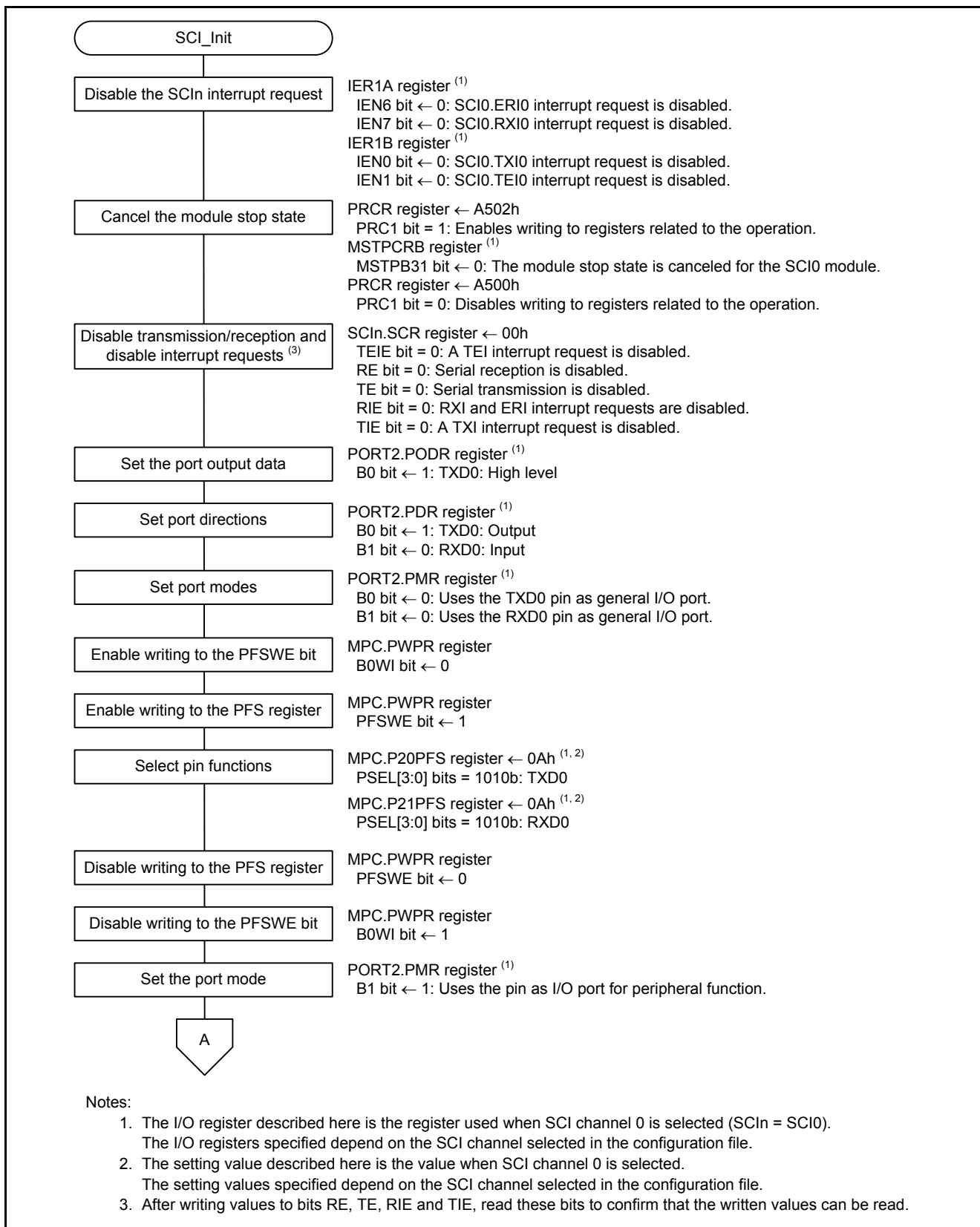


**Figure 5.10 Callback Function (SCI Receive Error)**

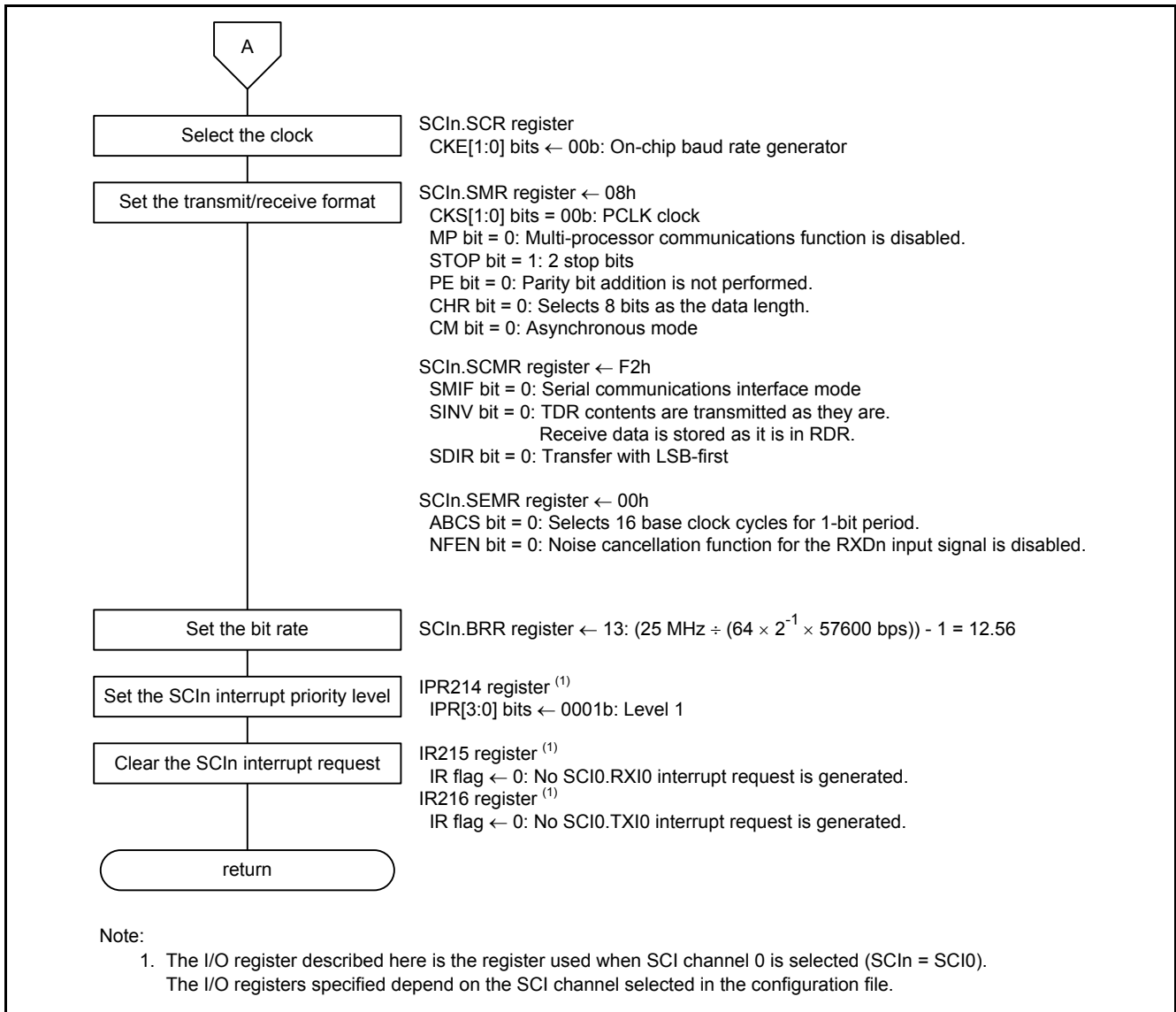


**5.9.7 User Interface Function (SCI Initialization)**

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



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



**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).

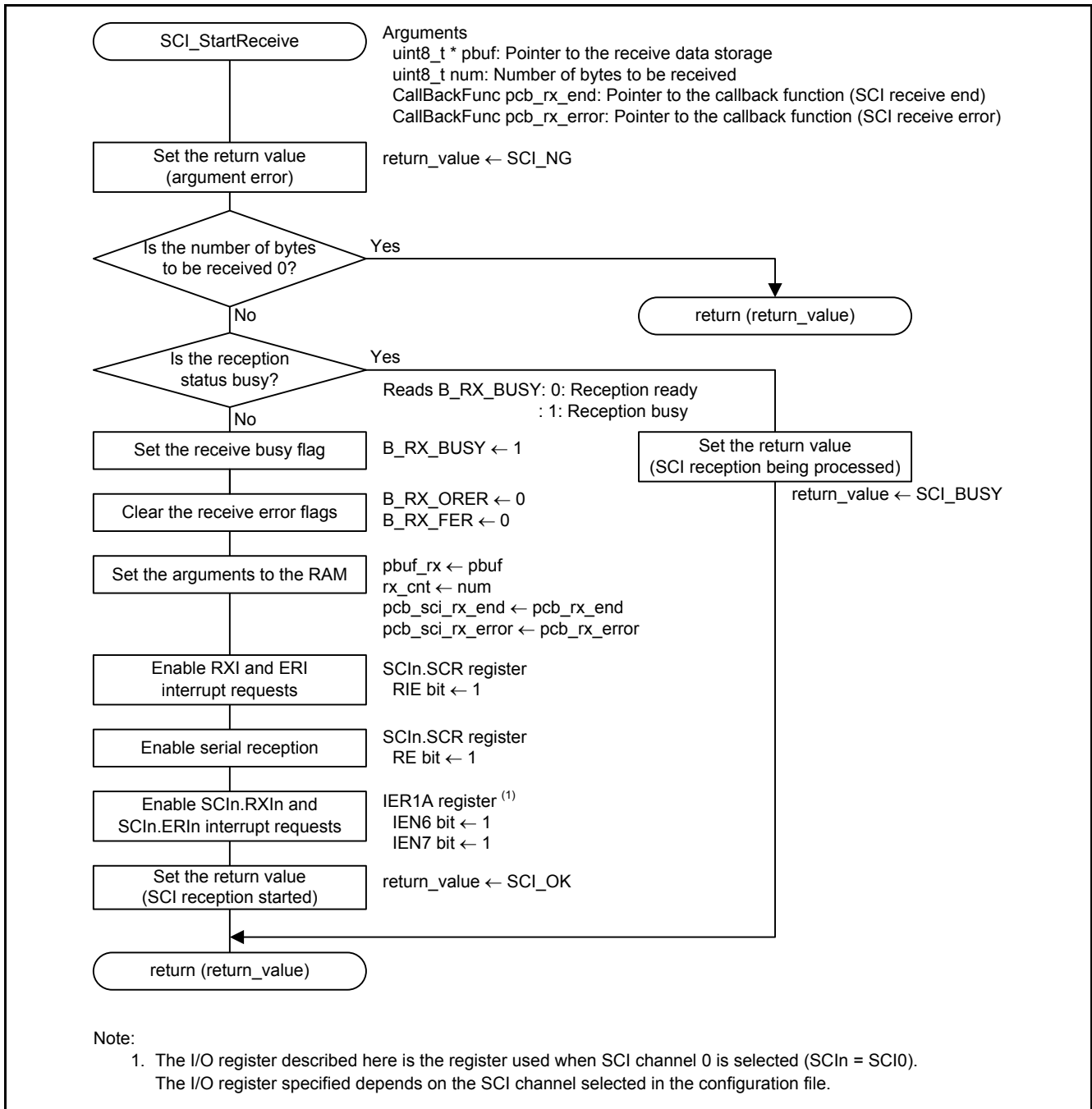


Figure 5.13 User Interface Function (SCI Receive Start)

5.9.9 User Interface Function (SCI Transmit Start)

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

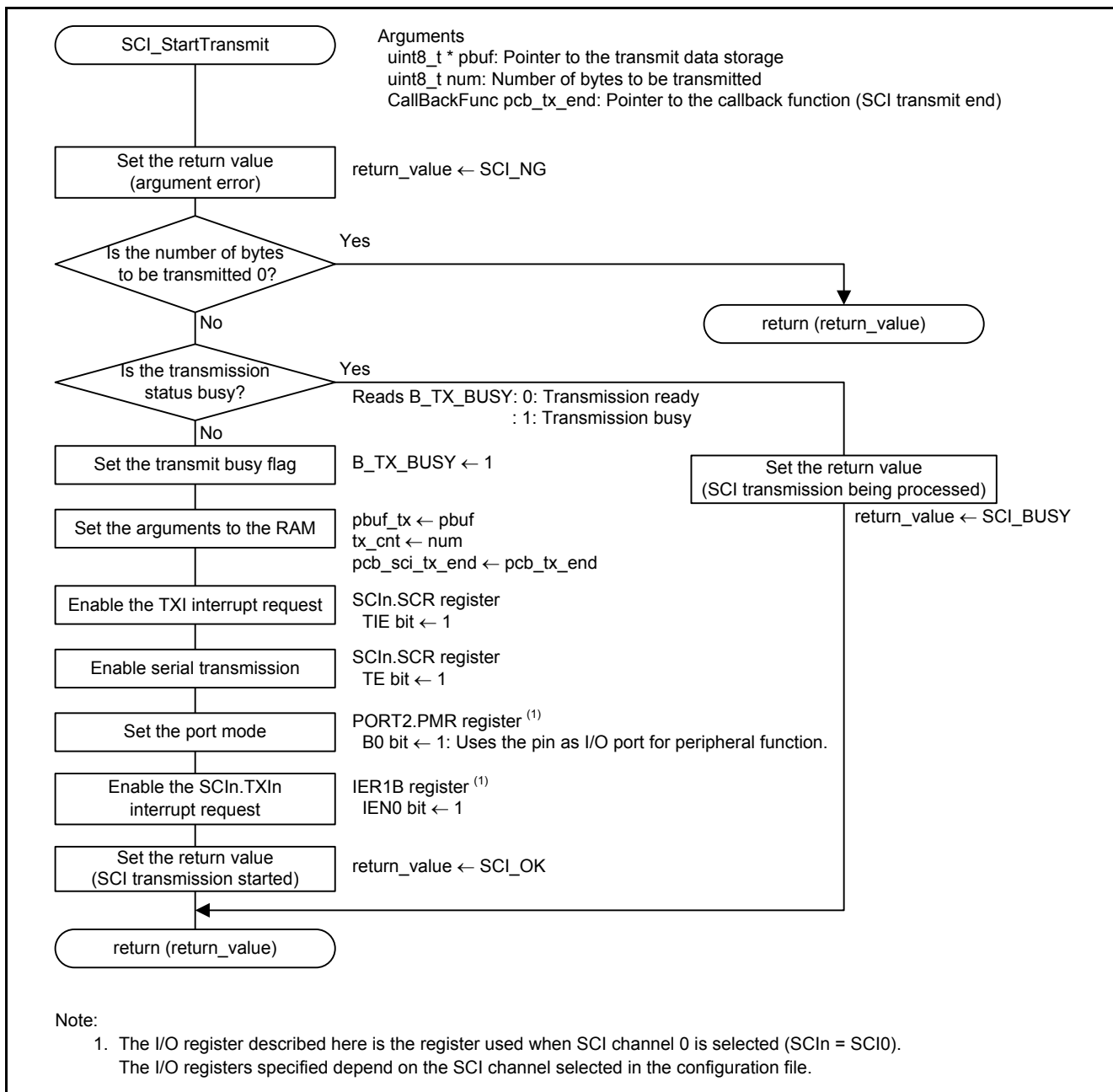
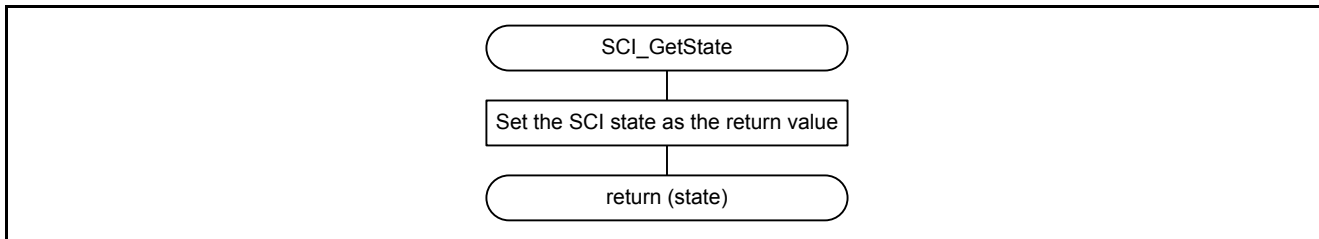


Figure 5.14 User Interface Function (SCI Transmit Start)

### 5.9.10 User Interface Function (SCI State Obtain)

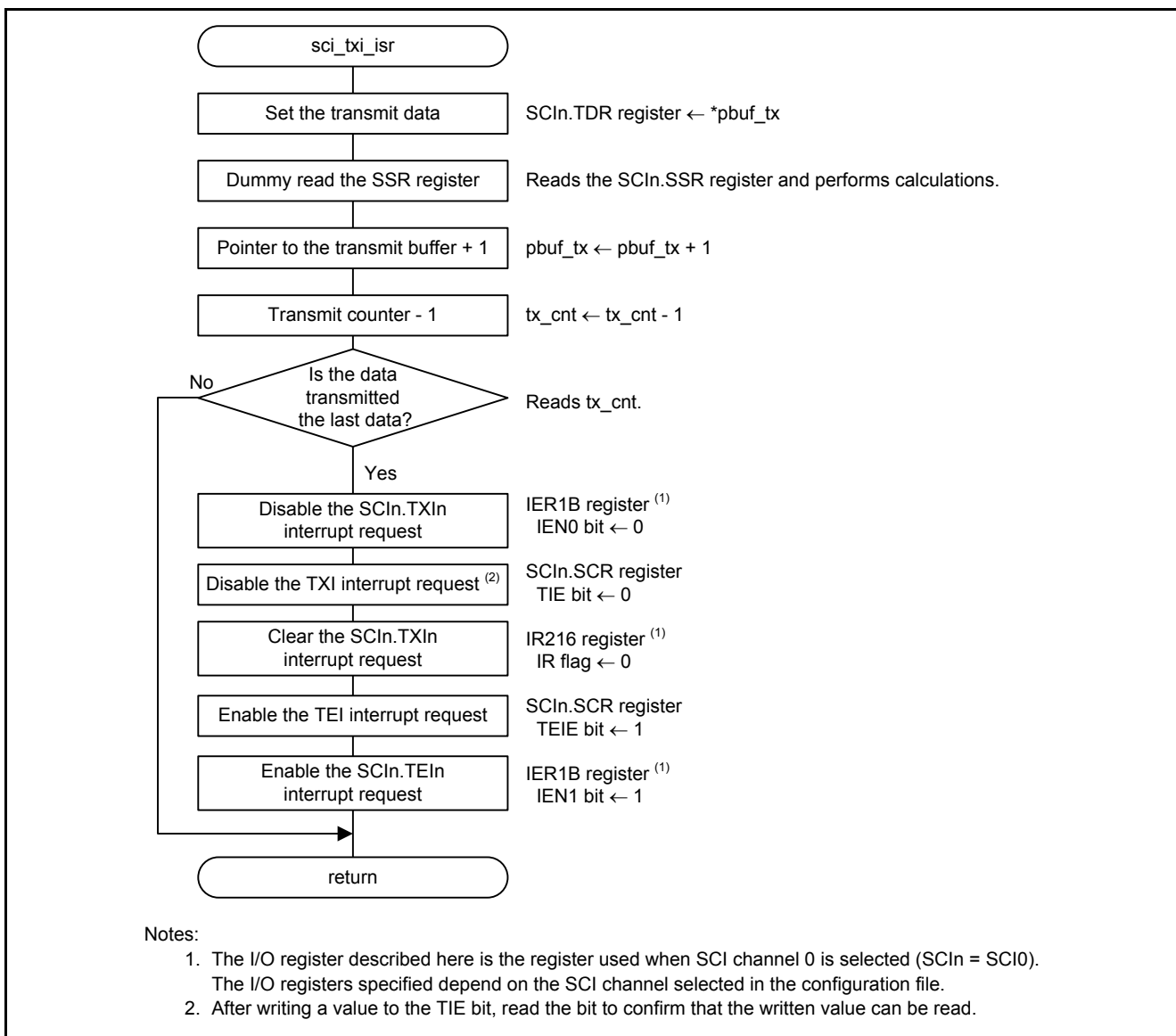
Figure 5.15 shows the User Interface Function (SCI State Obtain).



**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.



**Figure 5.16 Transmit Data Empty Interrupt**

### 5.9.12 Transmit End Interrupt

Figure 5.17 shows the Transmit End Interrupt.

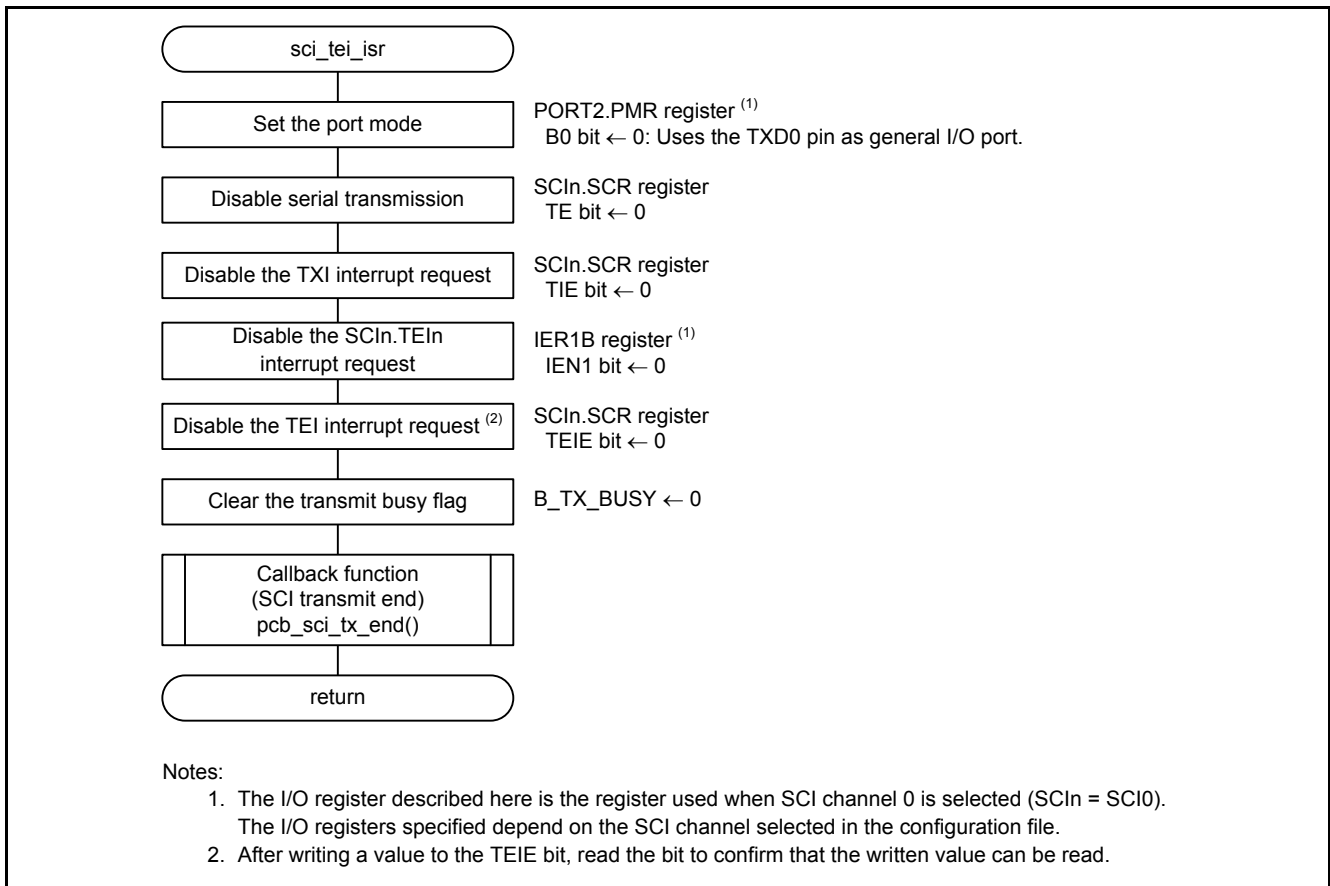


Figure 5.17 Transmit End Interrupt

5.9.13 Receive Data Full Interrupt

Figure 5.18 shows the Receive Data Full Interrupt.

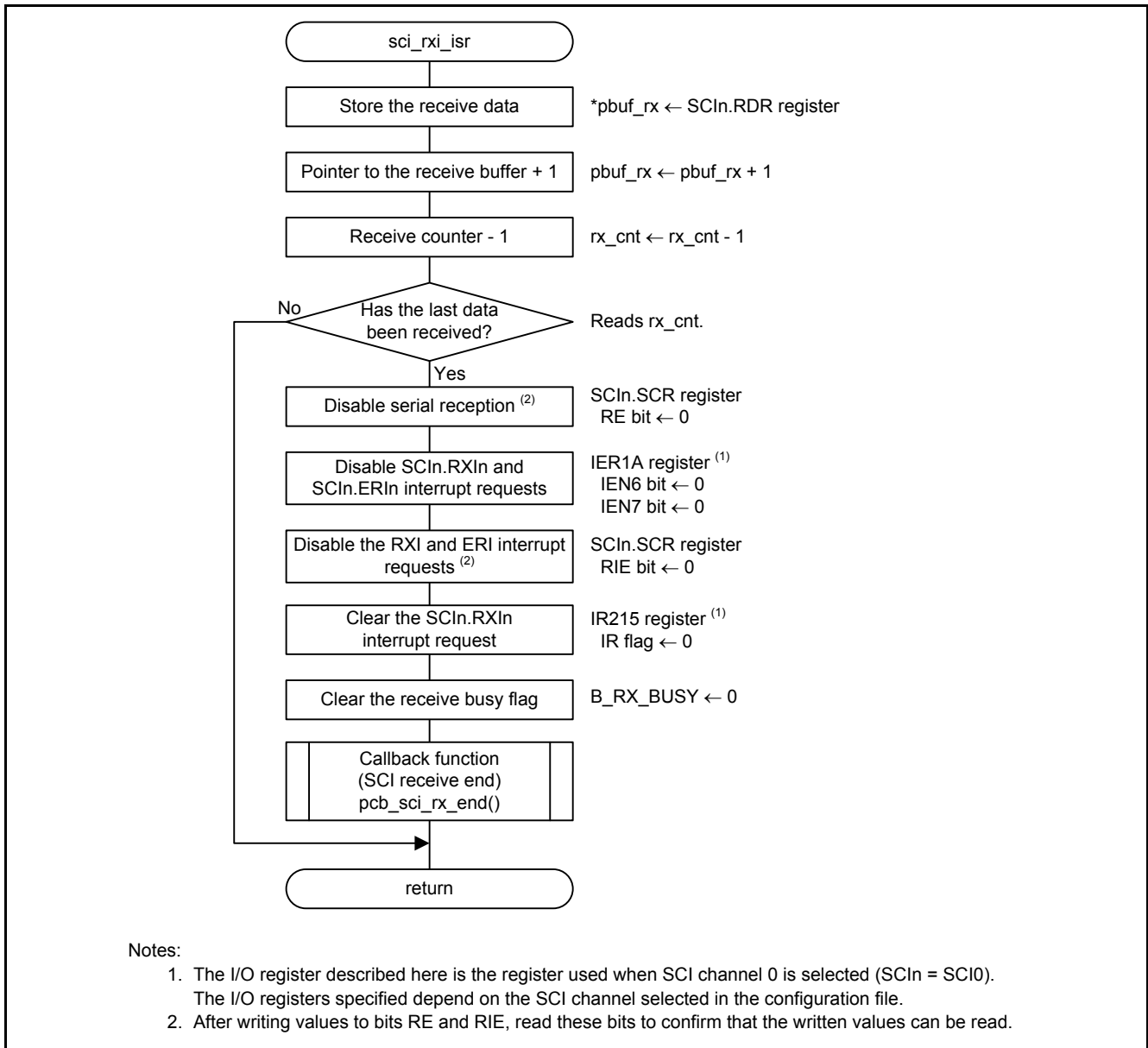


Figure 5.18 Receive Data Full Interrupt

5.9.14 Receive Error Interrupt

Figure 5.19 shows the Receive Error Interrupt.

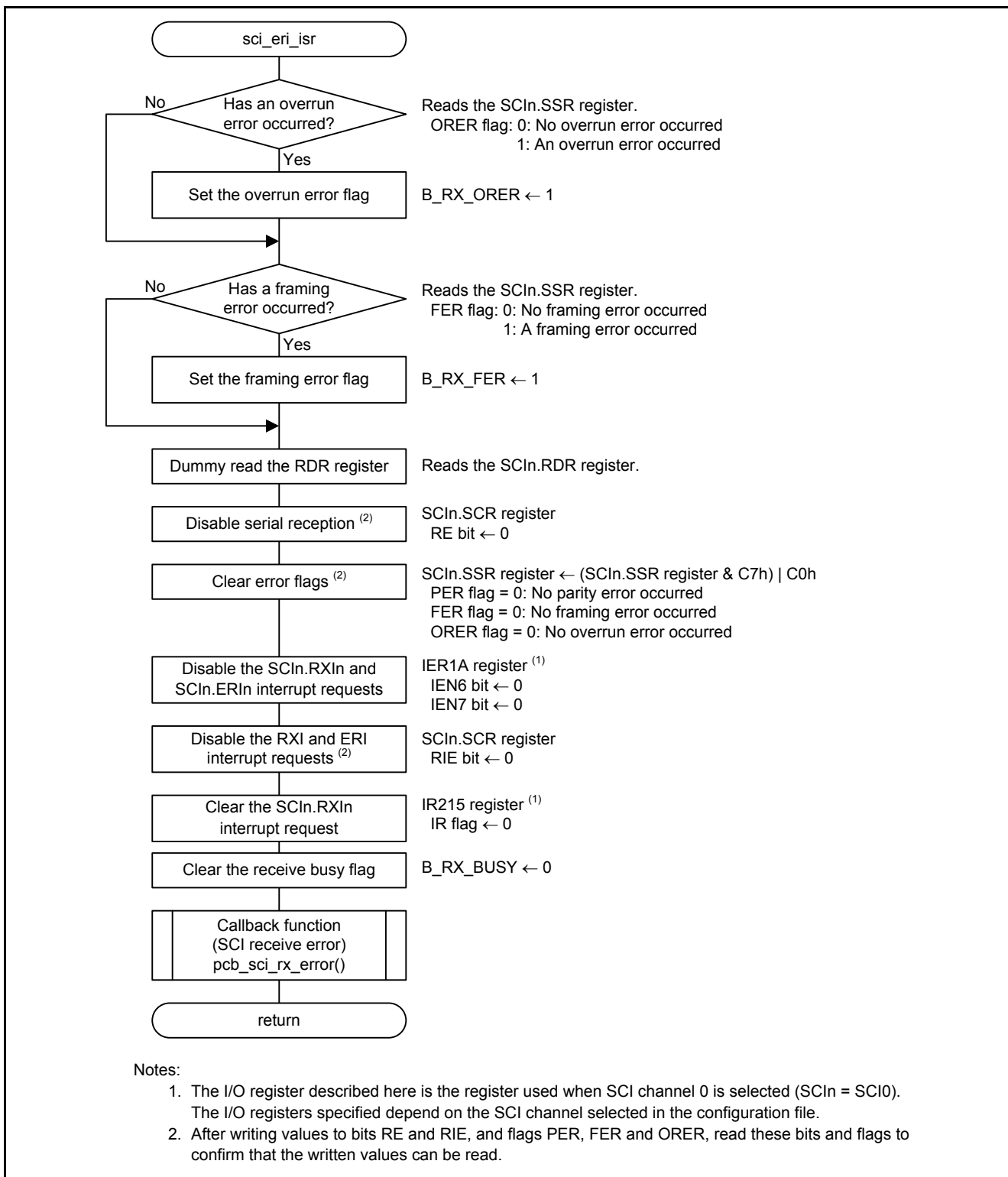


Figure 5.19 Receive Error Interrupt



5.9.15 SCI.ERI Interrupt Handling

Figure 5.20 shows the SCI.ERI Interrupt Handling.

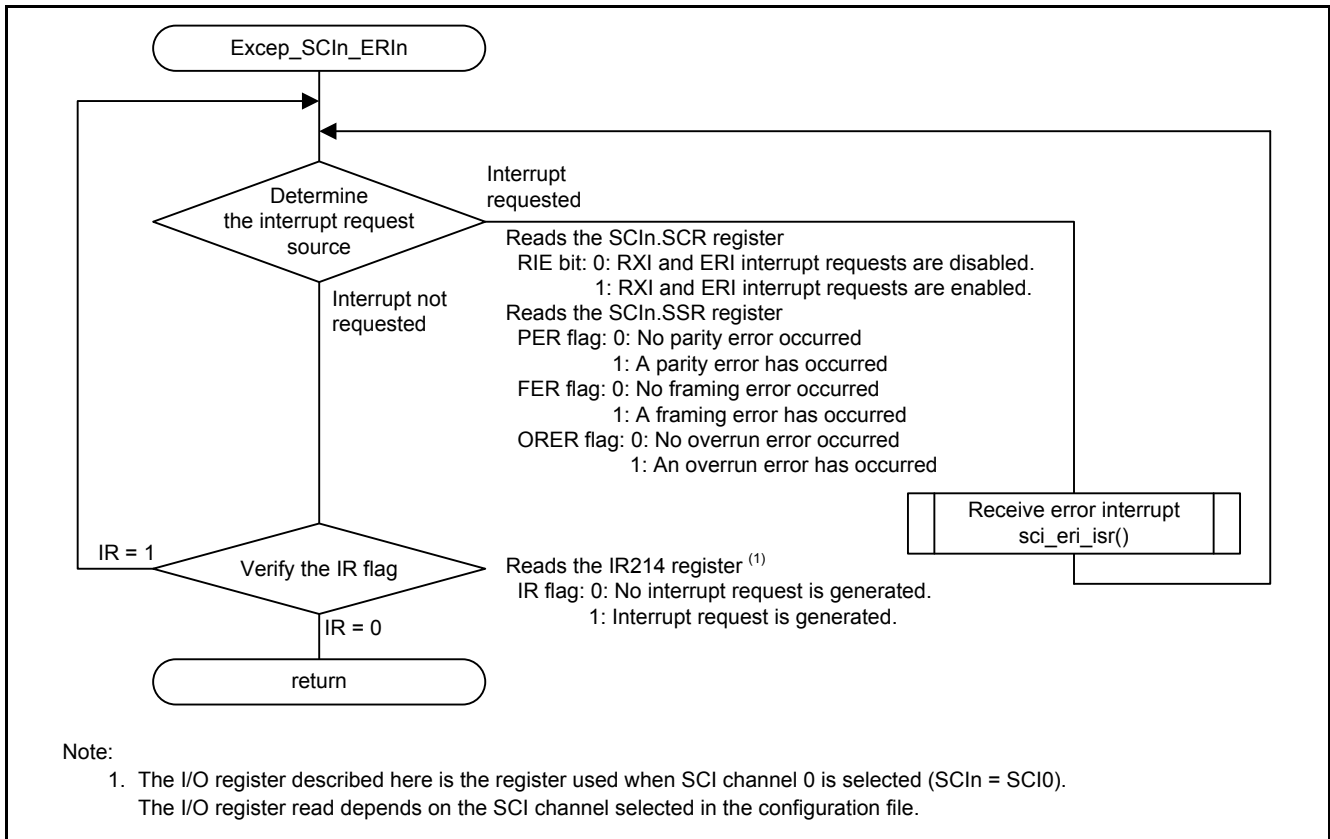


Figure 5.20 SCI.ERI Interrupt Handling

5.9.16 SCI.RXI Interrupt Handling

Figure 5.21 shows the SCI.RXI Interrupt Handling.

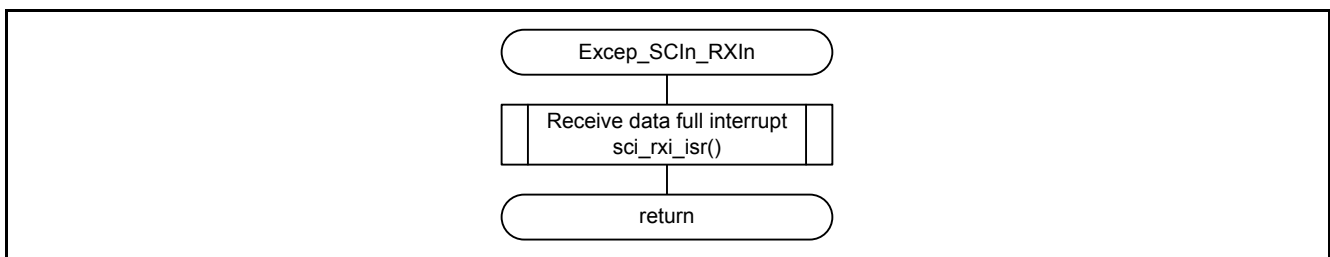
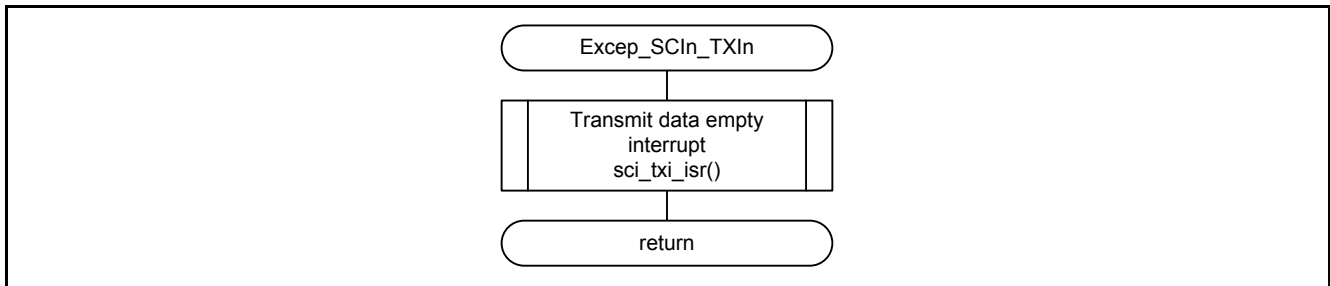


Figure 5.21 SCI.RXI Interrupt Handling

**5.9.17 SCI.TXI Interrupt Handling**

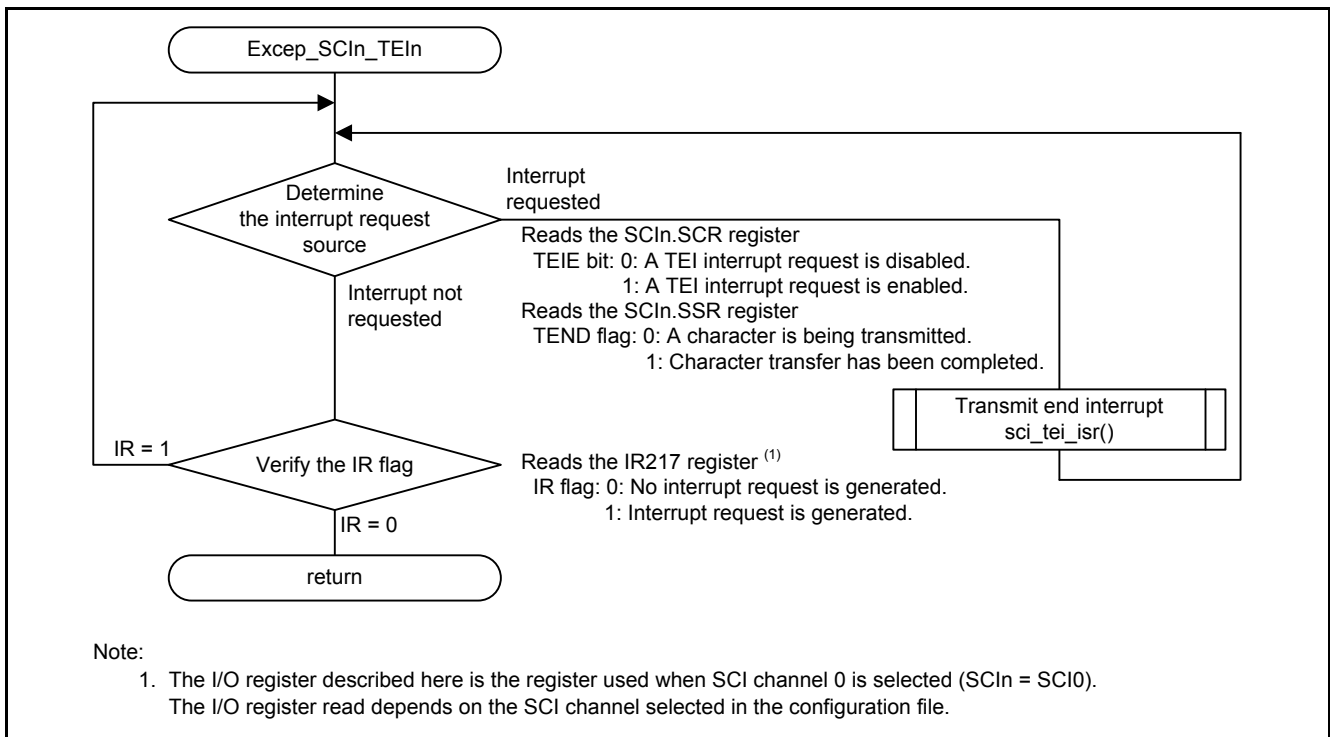
Figure 5.22 shows the SCI.TXI Interrupt Handling.



**Figure 5.22 SCI.TXI Interrupt Handling**

**5.9.18 SCI.TEI Interrupt Handling**

Figure 5.23 shows the SCI.TEI Interrupt Handling.



**Figure 5.23 SCI.TEI Interrupt Handling**

### **6. Applying This Application Note to the RX21A or RX220 Group**

The sample code accompanying this application note has been confirmed to operate with the RX210 Group. To make the sample code operate with the RX21A or RX220 Group, use this application note in conjunction with the Initial Setting application note for each group.

For details on using this application note with the RX21A and RX220 Groups, refer to “5. Applying the RX210 Group Application Note to the RX21A Group” in the RX21A Group Initial Setting application note, and “4. Applying the RX210 Group Application Note to the RX220 Group” in the RX220 Group Initial Setting application note.

Note: • When using the RX21A Group, SCI0 and SCI12 are not available. Use SCI1, SCI5, SCI6, SCI8, or SCI9.  
When using the RX220 Group, SCI0, SCI8, and SCI12 are not available. Use SCI1, SCI5, SCI6, or SCI9.

## **7. Sample Code**

Sample code can be downloaded from the Renesas Electronics website.

## **8. Reference Documents**

User's Manual: Hardware

RX210 Group User's Manual: Hardware Rev.1.50 (R01UH0037EJ)

RX21A Group User's Manual: Hardware Rev.1.00 (R01UH0251EJ)

RX220 Group User's Manual: Hardware Rev.1.10 (R01UH0292EJ)

The latest versions 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/>

|                         |   |
|-------------------------|---|
| <b>REVISION HISTORY</b> | RX210, RX21A, and RX220 Groups Application Note<br>Asynchronous Communication Using the SCI |
|-------------------------|---|

| Rev. | Date         | Description |   |
|------|--------------|-------------|---|
|      |              | Page        | Summary   |
| 1.00 | July 1, 2013 | —           | First edition issued  |
| 1.01 | July 1, 2014 | 1           | Products: Added the RX21A and RX220 Groups.   |
|      |              | 4           | 3. Reference Application Notes: Added the Initial Setting application notes for the RX21A and RX220 Groups.                                     |
|      |              | 18          | Modified the description of reference application note in the following functions: R_INIT_StopModule, R_INIT_NonExistentPort, and R_INIT_Clock. |
|      |              | 35          | 6. Applying This Application Note to the RX21A or RX220 Group: Added.   |
|      |              | 36          | 8. Reference Documents: Added the User's Manual: Hardware for the RX21A and RX220 Groups.   |
|      |              |             |   |

|   |
|---|
| All trademarks and registered trademarks are the property of their respective owners. |
|---|

## 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 products 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.



### 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.**

2801 Scott Boulevard Santa Clara, CA 95050-2549, 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: +44-1628-585-100, Fax: +44-1628-585-900

#### **Renesas Electronics Europe GmbH**

Arcadiastrasse 10, 40472 Düsseldorf, Germany  
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

#### **Renesas Electronics (China) Co., Ltd.**

Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, 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 Road, Putuo District, Shanghai, P. R. China 200333  
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: +852-2265-6688, Fax: +852 2886-9022/9044

#### **Renesas Electronics Taiwan Co., Ltd.**

13F, No. 363, Fu Shing North Road, Taipei 10543, 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-Ku, Seoul, 135-920, Korea  
Tel: +82-2-558-3737, Fax: +82-2-558-5141