RX64M, RX71M Group

Using the SCIg Bit Rate Modulation Function

R01AN2085EJ0110
Rev. 1.10
Dec. 21, 2020

Abstract

This document describes using the bit rate modulation function in the RX64M, RX71M Group to perform serial transmission and reception in asynchronous mode.

The bit rate modulation function reduces the bit rate error by correcting evenly the bit rate generated by the on-chip baud rate generator.

Products

RX64M Group

- RX64M Group 177- and 176-pin versions, ROM capacity: 2 MB to 4 MB
- RX64M Group 145- and 144-pin versions, ROM capacity: 2 MB to 4 MB
- RX64M Group 100-pin version, ROM capacity: 2 MB to 4 MB

RX71M Group

- RX71M Group 177- and 176-pin versions, ROM capacity: 2 MB to 4 MB
- RX71M Group 145- and 144-pin versions, ROM capacity: 2 MB to 4 MB
- RX71M Group 100-pin version, ROM capacity: 2 MB to 4 MB

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

This document describes using the serial communications interface (SCI) to perform serial transmission and reception in asynchronous mode. The bit rate modulation function is used for serial transmission and reception.

After a reset, transmission and reception are performed only once. The 12-byte character code "Hello world!" (without quotation marks) set in the transmit buffer is transmitted. LED0 turns on after the 12 bytes have been transmitted.

12 bytes of data are received. The received data is stored in the receive buffer, and LED1 turns on after all 12 bytes are received. If an error occurs during reception, reception is canceled, and LED2 turns on.

Bit rate: 57,600 bps
Data length: 8 bits
Stop bits: 2 bits
Parity: None
Hardware flow control: None

The peripheral functions are listed in Table 1.1 and Figure 1.1 shows a Usage Example.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
SCI (channel 7)	Perform serial transmission and reception in asynchronous mode
I/O ports	Turn on the LEDs

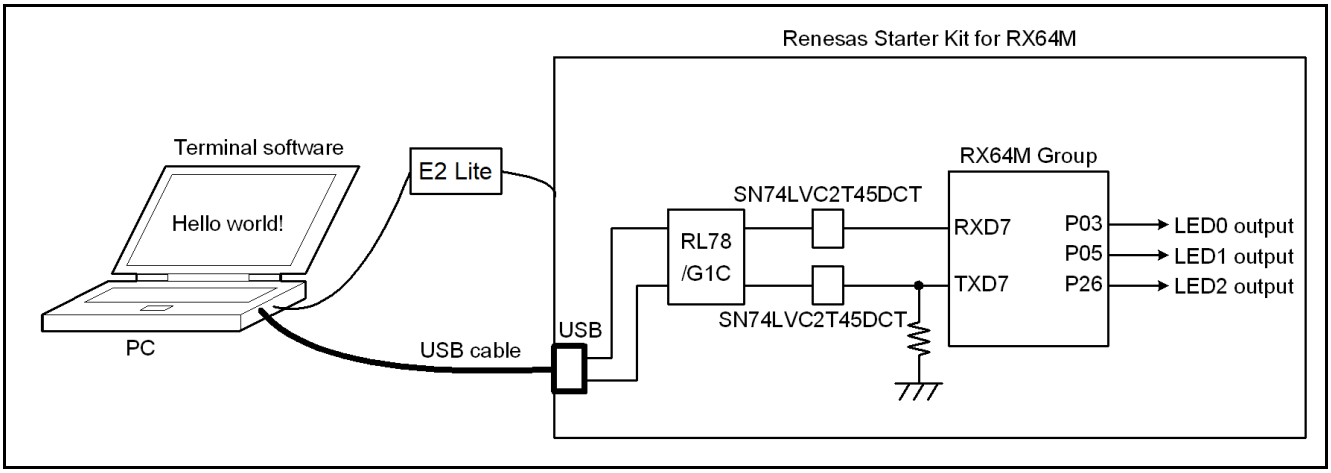


Figure 1.1 Usage Example

1.1 USB Serial Conversion

When the Renesas Starter Kit+ for RX64M is shipped, serial port SCI7 on the RX64M MCU is connected to the serial port on the RL78/G1C MCU and can be used as a virtual COM port. This document describes using this virtual COM port to communicate with the PC.

2. Confirmed Operating Conditions

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

Table 2.1 Confirmed Operating Conditions

Item	Contents
MCU used	R5F564MLCDFC (RX64M Group)
Operating frequencies	• Main clock: 24 MHz • PLL clock: 240 MHz (main clock divided by 1 and multiplied by 10) • System clock (ICLK): 120 MHz[Note1] (PLL clock divided by 2) • Peripheral module clock B (PCLKB): 60 MHz (PLL clock divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e² studio Version: 2020-10
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V3.02.00[Note2]
	Compile options The integrated development environment default settings are used.
iodefine.h version	0.9
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.10
Board used	Renesas Start Kit+ for RX64M (product part no.: R0K50564MSxxxBE)

Note1: When setting the frequency of ICLK to faster than 120 MHz in RX71M, the value of the MEMWAIT register needs to be changed.

Note2: If the same version of the toolchain (C compiler) specified in the original project is not in the import destination,the toolchain will not be selected and an error will occur.
Check the selected status of the toolchain on the project configuration dialog.

For the setting method, refer to FAQ 3000404.

FAQ 3000404 :Program ""make"" not found in PATH' error when attempting to build an imported project (e² studio)"

3. Reference Application Note

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

- RX64M Group Initial Setting Rev. 1.00 (R01AN1918EJ)

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 current at the time this document was created. However, the latest version is always recommended. The latest version can be downloaded from the Renesas Electronics Corporation website.

4. Hardware

4.1 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P03	Output	LED0 output (SCI transmission complete)
P05	Output	LED1 output (SCI reception complete)
P26	Output	LED2 output (SCI reception error)
P92/RXD7	Input	Receive data input pin for SCI7
P90/TXD7	Output	Transmit data output pin for SCI7

5. Software

After a reset, the user interface function (SCI initialization) is called, the SCI is initialized, and the transmit and receive operations are enabled.

When the user interface function (start SCI transmission) is called, the transmit data empty interrupt request is enabled. When the specified number of bytes has been transmitted, the callback function (SCI transmission complete) is called. The callback function (SCI transmission complete) turns on LED0.

When the user interface function (start SCI reception) is called, the receive data full interrupt request and receive error interrupt request are enabled. When the specified number of bytes has been received, the callback function (SCI reception complete) is called. The callback function (SCI reception complete) turns on LED1.

If a reception error occurs, the SCI transmit and receive operations are disabled, and the callback function (SCI reception error) is called. The callback function (SCI reception error) turns on LED2.

The following are the settings for the peripheral functions.

SCI

- Serial communication method: Asynchronous
- Bit rate: 57,600 bps
- Clock source: PCLKB (60 MHz)
- Data length: 8 bits
- Stop bits: 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 shows the Software Configuration.

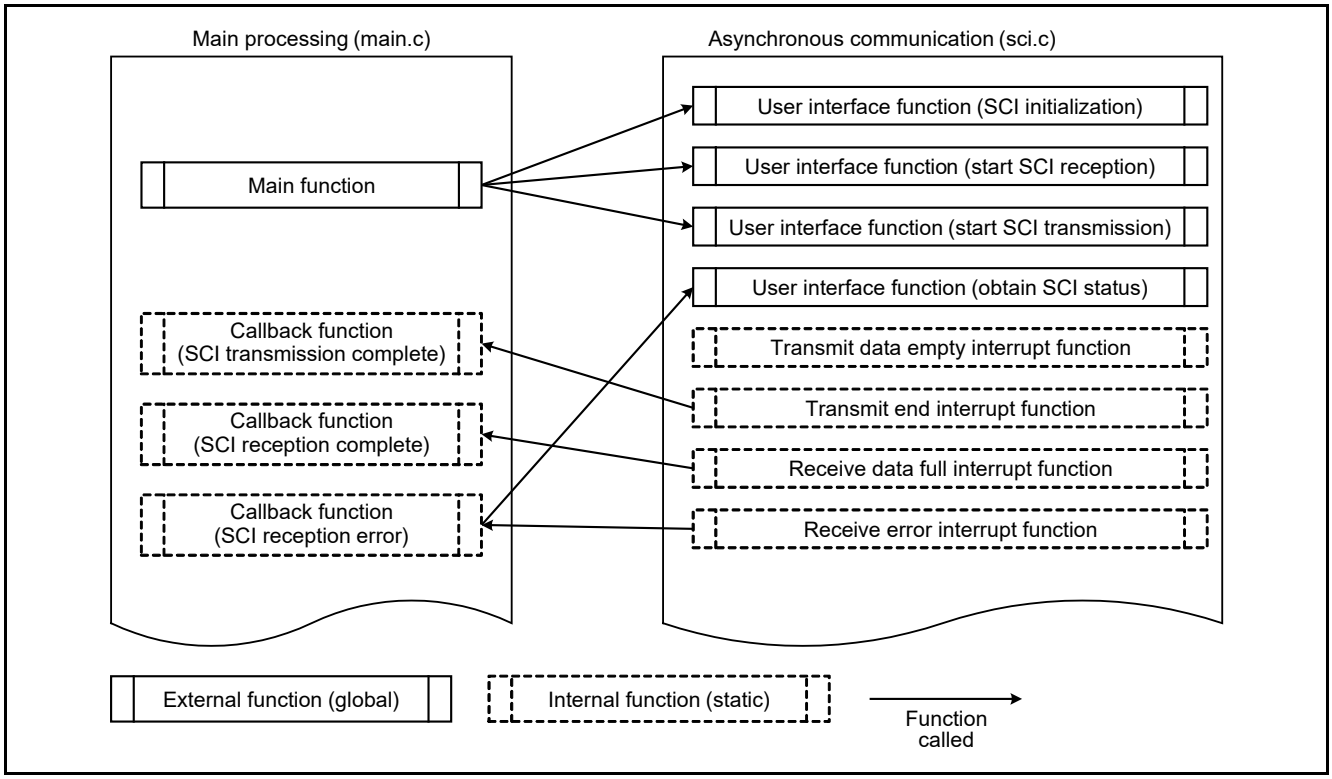


Figure 5.1 Software Configuration

5.1 Operation Overview

5.1.1 Serial Transmission

Figure 5.2 shows the Serial Transmission Timing Diagram. The following numbers correspond to operations and processing shown in the figure.

(1) Initialization
SCI initialization is performed by the user interface function (SCI initialization), the SCR.TIE bit is set to 1 (a TXI interrupt request is enabled), and the SCR.TE bit is set to 1 (serial transmission is enabled). (In the sample code, the SCR.TE and RE bits are set to 1 simultaneously.)

(2) Start transmission
The user interface function (start SCI transmission) is used to check the transmission busy flag (variable). If the transmission busy flag is 1 (transmitting data), SCI_BUSY (SCI transmitting data) is returned. If the transmission busy flag is 0 (ready to transmit data), after the transmission busy flag is set to 1, the IERm.IENj bit for the TXI interrupt is set to 1 (interrupt request is enabled). At that time, a TXI interrupt occurs as the IRi.IR flag for the TXI interrupt has already become 1 (interrupt request is generated) in (1) above.

(3) Transmit data
The value in the transmit buffer is written to the TDR register in the TXI interrupt handling. When the transmit buffer value is transferred from the TDR register to the TSR register, the IRi.IR flag for the TXI interrupt becomes 1 (interrupt request is generated), and an interrupt is generated. This processing is repeated until the last data is written. When the last data is written, the SCR.TEIE bit is set to 1 (TEI interrupt request is enabled).

(4) Transmission complete
When the last data has been transmitted, the TEI interrupt request is generated. The TEIE bit is set to 0 (TEI interrupt request is disabled) in the TEI interrupt handling. Then, the transmission busy flag is set to 0, and the callback function (SCI transmission complete) is called.

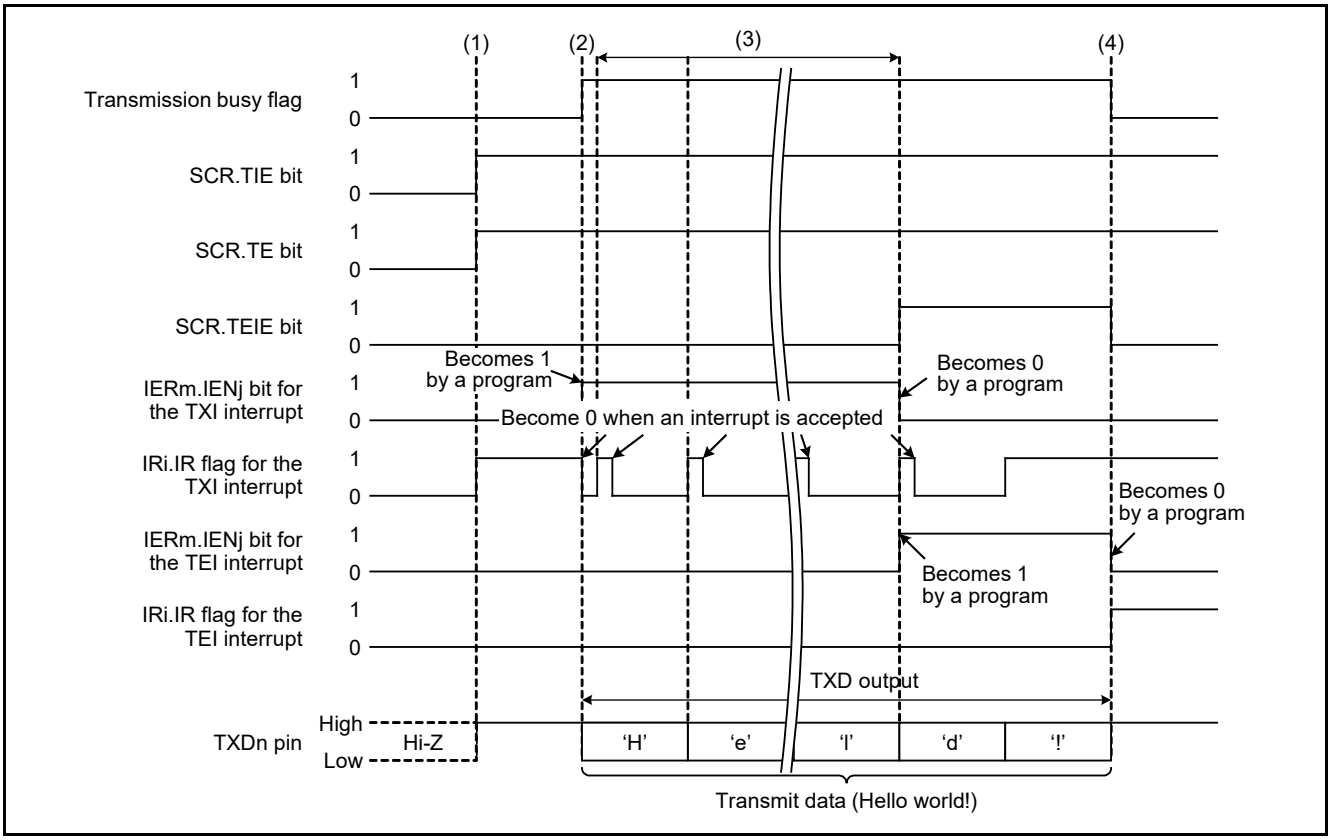


Figure 5.2 Serial Transmission Timing Diagram

5.1.2 Serial Reception

Figure 5.3 shows the Serial Reception Timing Diagram. The following numbers correspond to operations and processing shown in the figure.

(1) Initialization
SCI initialization is performed by the user interface function (SCI initialization), the SCR.RIE bit is set to 1 (RXI and ERI interrupt requests are enabled), and the SCR.RE bit is set to 1 (serial reception is enabled). (In the sample code, the SCR.RE and TE bits are set to 1 simultaneously.)

(2) Start reception *1
The user interface function (start SCI reception) is used to check the reception busy flag (variable). If the reception busy flag is 1 (receiving data), SCI_BUSY (SCI receiving data) is returned. If the reception busy flag is 0 (ready to receive data), the reception busy flag is set to 1, and the error flag is cleared. The IERm.IENj bit for the RXI interrupt or ERI interrupt are set to 1 (interrupt request is enabled).

(3) Receive data
When data is received, the RXI interrupt request is generated. The value in the RDR register is written to the receive buffer in the RXI interrupt handling.
If a reception error *2 occurs, the ERI interrupt request is generated. In the ERI interrupt handling, the error flag (variable) is set, and the RDR register is dummy read. The RE bit and TE bit are set to 0, and the error flags in the SSR register are cleared. Bits RIE, TIE, TEIE and the reception busy flag are all set to 0, and the callback function (SCI reception error) is called.

(4) Reception complete
When the last data has been received, the reception busy flag is set to 0, and the callback function (SCI reception complete) is called.

Note 1. After calling the user interface function (SCI initialization), and before calling the user interface function (start SCI reception), if 2 or more bytes of data are received, an overrun error occurs.

Note 2. After a reception error occurs, if serial transmission and reception are restarted, after calling the user interface function (SCI initialization), call the user interface functions (start SCI reception and start SCI transmission).

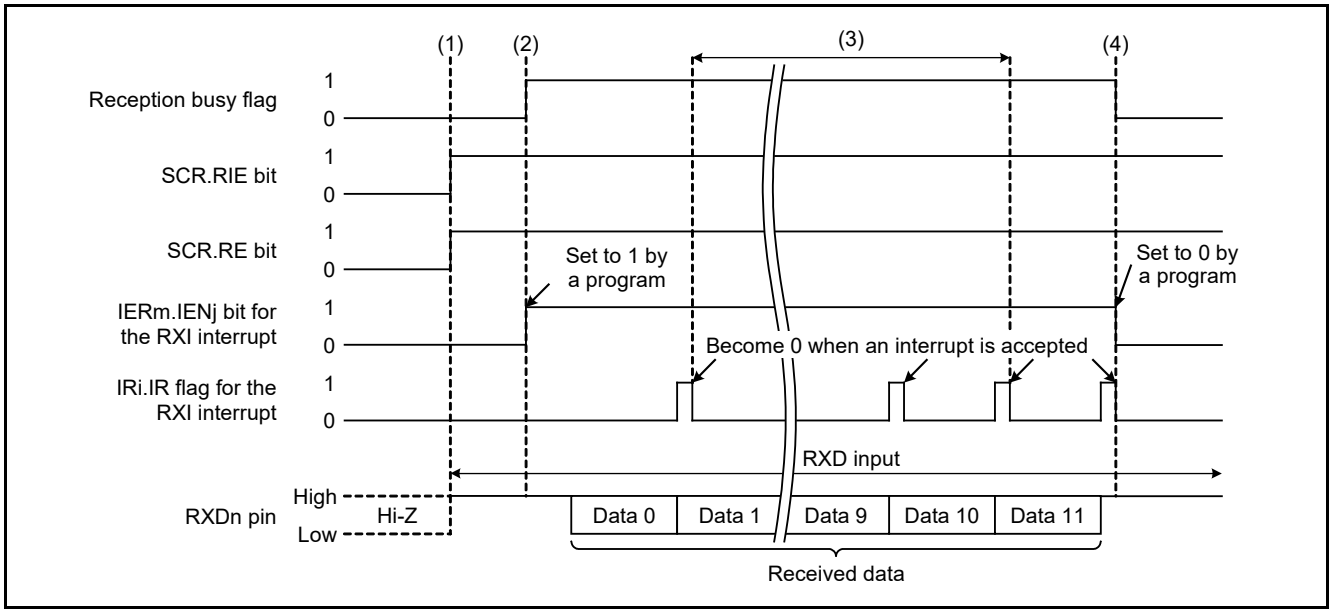


Figure 5.3 Serial Reception Timing Diagram

5.2 Bit Rate Modulation Function

5.2.1 About the Bit Rate Modulation Function

The bit rate modulation function evenly corrects the bit rate by enabling the internal clock, which is selected by the SMR.CKS[1:0] bits, with the value specified by the MDDR register. For example, when the MDDR register is set to 160, 160 clocks out of 256 clocks are validated (96 clocks are not validated).

For more details on the bit rate modulation function, refer to section 40.9 Bit Rate Modulation Function in Rev.1.00 of the RX64M Group User's Manual: Hardware.

5.2.2 Using the Bit Rate Modulation Function

When the SEMR.BRME bit is set to 1, the bit rate modulation function is enabled. Then the BRR and MDDR registers are set.

Set the BRR and MDDR registers with values to make the bit rate error be within the available range of the serial communications. The bit rate error can be calculated using the formulas in Table 5.1. For details on the formulas used to calculate the bit rate error when using the bit rate modulation function, refer to section 40.2.12 Modulation Duty Register (MDDR) in Rev.1.00 of the RX64M Group User's Manual: Hardware.

Table 5.1 Formulas for Calculating the Bit Rate Error When Using the Bit Rate Modulation Function

Mode	SEMR Register Setting		Error
	BGDM bit	ABCS bit	
Asynchronous mode	0	0	Error [%] = $\left\{\frac{PCLKB\times10^6}{B\times64\times2^{2n-1}\times\left(\frac{256}{M}\right)\times(N+1)}-1\right\}\times100$
	0	1	Error [%] = $\left\{\frac{PCLKB\times10^6}{B\times32\times2^{2n-1}\times\left(\frac{256}{M}\right)\times(N+1)}-1\right\}\times100$
	1	0	
	1	1	Error [%] = $\left\{\frac{PCLKB\times10^6}{B\times16\times2^{2n-1}\times\left(\frac{256}{M}\right)\times(N+1)}-1\right\}\times100$

B: Bit rate [bps]
M: MDDR register setting value ($128 \le M \le 255$)
N: BRR register setting value for the baud rate generator ($0 \le N \le 255$)
PCLKB: Operating frequency [MHz]
n: SMR.CKS[1:0] bit setting value ($0 \le n \le 3$)

5.2.3 Comparison of Using and Not Using the Bit Rate Modulation Function

Table 5.2 shows the settings to obtain the minimum error when the bit rate is 57,600 bps.

Table 5.2 Serial Transmission and Reception Settings, and N and M Settings to Obtain the Minimum Error

	Bit Rate Modulation Function	
	Values When Used	Values When Not Used
Transfer speed [bps] (B)	57,600	
Operating frequency [MHz] (PCLKB)	60	
SEMR.BGDM bit setting	0	0
SEMR.ABCS bit setting	0	0
SMR.CKS[1:0] bit setting (n)	0	0
BRR register setting (N)	21	32
MDDR register setting (M)	173	

(a) Calculating the bit rate error when using the bit rate modulation function
This formula assumes the SEMR.BGDM and ABCS bits are 0.

$$\text{Error [\%]} = \left\{ \frac{PCLKB \times 10^6}{B \times 64 \times 2^{2n-1} \times \left(\frac{256}{M}\right) \times (N+1)} - 1 \right\} \times 100$$

When you replace PCLKB, M, N, and n in the above formula, you get

$$\text{Error [\%]} = \left\{ \frac{60 \times 10^6}{57{,}600 \times 64 \times 2^{2\times 0-1} \times \left(\frac{256}{173}\right) \times (21+1)} - 1 \right\} \times 100$$

So, the error is

$$\left\{ \frac{60 \times 10^6}{57{,}600 \times 64 \times 2^{-1} \times \left(\frac{256}{173}\right) \times 22} - 1 \right\} \times 100 = -0.008692886\ [\%]$$

(b) Calculating the bit rate error when not using the bit rate modulation function
This formula [*1] assumes the SEMR.BGDM and ABCS bits are 0.

$$\text{Error [\%]} = \left\{ \frac{PCLKB \times 10^6}{B \times 64 \times 2^{2n-1} \times (N+1)} - 1 \right\} \times 100$$

When you replace PCLKB, N, and n in the above formula, you get

$$\text{Error [\%]} = \left\{ \frac{60 \times 10^6}{57{,}600 \times 64 \times 2^{2\times 0-1} \times (32+1)} \right\} \times 100$$

So, the error is

$$\left\{ \frac{60 \times 10^6}{57{,}600 \times 64 \times 2^{-1} \times 33} - 1 \right\} \times 100 = -1.357323232 \text{ [\%]}$$

The results from (a) and (b) show that the bit rate error can be reduced by using the bit rate modulation function.

Note 1. For details on calculating the bit rate error when not using the bit rate modulation function, refer to section 40.2.11 Bit Rate Register (BRR) in Rev.1.00 of the RX64M Group User's Manual: Hardware.

Table 5.3 and Table 5.4 list the results when the bit rate error is calculated using values not included in this application note.

Table 5.3 Bit Rate Error When Using the Bit Rate Modulation Function

Bit Rate [bps]	PCLKB Operating Frequencies [MHz]											
	10				30				60			
	n	N	M	Error [%]	n	N	M	Error [%]	n	N	M	Error [%]
1,200	0	176	174	0.0011	1	176	232	0.0011	1	205	135	–0.0031
2,400	0	117	232	0.0011	0	205	135	–0.0031	1	176	232	0.0011
4,800	0	58	232	0.0011	0	176	232	0.0011	0	205	135	–0.0031
9,600	0	21	173	–0.0087	0	73	194	0.0069	0	176	232	0.0011
14,400	0	14	177	0.0298	0	58	232	0.0011	0	117	232	0.0011
19,200	0	10	173	–0.0087	0	36	194	0.0069	0	73	194	0.0069
38,400	0	6	220	–0.0913	0	22	241	–0.0715	0	36	194	0.0069
57,600	0	4	236	0.0298	0	10	173	–0.0087	0	21	173	–0.0087

Table 5.4 Bit Rate Error When Not Using the Bit Rate Modulation Function

Bit Rate [bps]	PCLKB Operating Frequencies [MHz]								
	10			30			60		
	n	N	Error [%]	n	N	Error [%]	n	N	Error [%]
1,200	1	64	0.1603	1	194	0.1603	2	97	–0.3508
2,400	0	129	0.1603	1	97	–0.3508	1	194	0.1603
4,800	0	64	0.1603	0	194	0.1603	1	97	–0.3508
9,600	0	32	–1.3573	0	97	–0.3508	0	194	0.1603
14,400	0	21	–1.3573	0	64	0.1603	0	129	0.1603
19,200	0	15	1.7252	0	48	–0.3508	0	97	–0.3508
38,400	0	7	1.7252	0	23	1.7252	0	48	–0.3508
57,600	0	4	8.5069	0	15	1.7252	0	32	–1.357

5.3 File Composition

Table 5.5 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 5.5 Files Used in the Sample Code

File Name	Outline	Remarks
main.c	Main processing	
sci.c	Asynchronous communication	
sci.h	Header file for sci.c	
sci_cfg.h	Configuration header file for sci.c	

5.4 Option-Setting Memory

Table 5.6 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.6 Option-Setting Memory Configured in the Sample Code

Symbol	Addresses	Setting Value	Contents
OFS0	0012 0068h to 0012 006Bh	FFFF FFFFh	The IWDT is stopped after a reset. The WDT is stopped after a reset.
OFS1	0012 006Ch to 0012 006Fh	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDE	0012 0064h to 0012 0067h	FFFF FFFFh	Little endian

5.5 Constants

Table 5.7 to Table 5.10 list the constants used in the sample code.

Table 5.7 Constants Used in the Sample Code (main.c)

Constant Name	Setting Value	Contents
LED0_REG_PODR	PORT0.PODR.BIT.B3	LED0 output data storage bit
LED0_REG_PDR	PORT0.PDR.BIT.B3	LED0 direction control bit
LED0_REG_PMR	PORT0.PMR.BIT.B3	LED0 pin mode control bit
LED1_REG_PODR	PORT0.PODR.BIT.B5	LED1 output data storage bit
LED1_REG_PDR	PORT0.PDR.BIT.B5	LED1 direction control bit
LED1_REG_PMR	PORT0.PMR.BIT.B5	LED1 pin mode control bit
LED2_REG_PODR	PORT2.PODR.BIT.B6	LED2 output data storage bit
LED2_REG_PDR	PORT2.PDR.BIT.B6	LED2 direction control bit
LED2_REG_PMR	PORT2.PMR.BIT.B6	LED2 pin mode control bit
LED_ON	0	LED output data: On
LED_OFF	1	LED output data: Off
BUF_SIZE	12	Buffer size
NULL_SIZE	1	NULL code size
SCI_B_TX_BUSY	sci_state.bit.b_tx_busy	Transmission busy flag 0: Ready for transmission 1: Transmitting data
SCI_B_RX_BUSY	sci_state.bit.b_rx_busy	Reception busy flag 0: Ready to receive data 1: Receiving data
SCI_B_RX_ORER	sci_state.bit.b_rx_orer	Overrun error flag 0: Overrun error did not occur 1: Overrun error occurred
SCI_B_RX_FER	sci_state.bit.b_rx_fer	Framing error flag 0: Framing error did not occur 1: Framing error occurred

Table 5.8 Constants Used in the Sample Code (sci.c)

Constant Name	Setting Value	Contents
SSR_ERROR_FLAGS	38h	Bit pattern for the SCI.SSR register error flags
B_TX_BUSY	state.bit_b_tx_busy	Transmission busy flag 0: Ready to transmit data 1: Transmitting data
B_RX_BUSY	state_bit_b_rx_busy	Reception busy flag 0: Ready to receive data 1: Receiving data
B_RX_ORER	state.bit_b_rx_orer	Overrun error flag 0: Overrun error did not occur 1: Overrun error occurred
B_RX_FER	state.bit_b_rx_fer	Framing error flag 0: Framing error did not occur 1: Framing error occurred

Table 5.9 Constants Used in the Sample Code (sci.h)

Constant Name	Setting Value	Contents
SCI_OK	00h	Return value for the SCI_StartTransmit function and SCI_StartReceive function: SCI transmit/receive start
SCI_BUSY	01h	Return value for the SCI_StartTransmit function and SCI_StartReceive function: SCI transmitting/receiving
SCI_NG	02h	Return value for the SCI_StartTransmit function and SCI_StartReceive function: Argument error (number of bytes transmitted/received is 0)
SCIn	SCI7	SCI channel: SCI7
GROUPBLn	GROUPBL0	Group BL0 interrupt
MSTP_SCIn	MSTP(SCI7)	SCI7 module stop setting bit
IPR_SCIn_RXIn	IPR(SCI7,RXI7)	SCI7.RXI7 interrupt priority level setting bit
IPR_SCIn_TXIn	IPR(SCI7,TXI7)	SCI7.TXI7 interrupt priority level setting bit
IPR_SCIn_GROUPBLn	IPR(ICU,GROUPBL0)	SCI7.GROUPBL0 interrupt priority level setting bit
IR_SCIn_RXIn	IR(SCI7,RXI7)	SCI7.RXI7 interrupt status flag
IR_SCIn_TXIn	IR(SCI7,TXI7)	SCI7.TXI7 interrupt status flag
IR_SCIn_GROUPBLn	IR(ICU,GROUPBL0)	SCI7.GROUPBL0 interrupt status flag
IS_SCIn_ERIn	ICU.GRPBL0.BIT.IS15	SCI7.ERI7 interrupt status flag
IS_SCIn_TEIn	ICU.GRPBL0.BIT.IS14	SCI7.TEI7 interrupt status flag
IEN_SCIn_RXIn	IEN(SCI7,RXI7)	SCI7.RXI7 interrupt request enable bit
IEN_SCIn_TXIn	IEN(SCI7,TXI7)	SCI7.TXI7 interrupt request enable bit
IEN_SCIn_GROUPBLn	IEN(ICU,GROUPBL0)	SCI7.GROUPBL0 interrupt request enable bit
EN_SCIn_ERIn	ICU.GENBL0.BIT.EN15	SCI7.ERI7 interrupt request enable bit
EN_SCIn_TEIn	ICU.GENBL0.BIT.EN14	SCI7.TEI7 interrupt request enable bit
RXDn_PDR	PORT9.PDR.BIT.B2	P92 direction control bit
RXDn_PMR	PORT9.PMR.BIT.B2	P92 pin mode control bit
RXDnPFS	P92PFS	P92 pin function control bit
TXDn_PODR	PORT9.PODR.BIT.B0	P90 output data storage bit
TXDn_PDR	PORT9.PDR.BIT.B0	P90 direction control bit
TXDn_PMR	PORT9.PMR.BIT.B0	P90 pin mode control bit
TXDnPFS	P90PFS	P90 pin function control bit
PSEL_SETTING	0x0Ah	Setting value for the pin function select bit: RXD7, TXD7

Table 5.10 Constants Used in the Sample Code (sci_cfg.h)

Constant Name	Setting Value	Contents
ENABLE_BIT_RATE_MODULATION	—	Bit rate modulation function enabled
DISABLE_BIT_RATE_MODULATION	—	Bit rate modulation function disabled
SELECT_SCI7	—	Channel SCI7 selected

5.6 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                        /* Boundary alignment value for structure members: Alignment by member type */
typedef   union
{
    uint8_t    byte;
    struct
    {
        uint8_t    b_tx_busy   :1;    /* Transmission busy flag    0: Ready to transmit data    1: Transmitting data */
        uint8_t    b_rx_busy   :1;    /* Reception busy flag    0: Ready to receive data    1: Receiving data */
        uint8_t    b_rx_orer   :1;    /* Overrun error flag    0: Overrun error did not occur    1: Overrun error occurred */
        uint8_t    b_rx_fer    :1;    /* Framing error flag    0: Framing error did not occur    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

5.7 Variables

Table 5.11 lists the static Variables.

Table 5.11 static Variables

Type	Variable Name	Contents	Function
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 status	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 receive buffer	SCI_StartReceive sci_rxi_isr
static uint8_t	rx_cnt	Receive counter	
static sci_state_t	state	SCI status	SCI_StartReceive SCI_StartTransmit SCI_GetState sci_tei_isr sci_rxi_isr sci_eri_isr

5.8 Functions

Table 5.12 lists the Functions.

Table 5.12 Functions

Function Name	Outline
main	Main processing
port_init	Port initialization
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
peripheral_init	Peripheral function initialization
cb_sci_tx_end	Callback function (SCI transmission complete)
cb_sci_rx_end	Callback function (SCI reception complete)
cb_sci_rx_error	Callback function (SCI reception error)
SCI_Init	User interface function (SCI initialization)
SCI_StartReceive	User interface function (start SCI reception)
SCI_StartTransmit	User interface function (start SCI transmission)
SCI_GetState	User interface function (obtain SCI status)
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
Excep_ICU_GROUPBLn	Group BL0 interrupt handling

5.9 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	After initialization, this function starts SCI reception and transmission.
Arguments	None
Return values	None

port_init	
Outline	Port initialization
Header	None
Declaration	static void port_init(void)
Description	This function initializes the ports.
Arguments	None
Return values	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	This function configures settings to enter the module-stop state.
Arguments	None
Return values	None
Remarks	Transition to the module-stop state is not performed in the sample code. For more information on this function, refer to the RX64M Group Initial Setting Rev. 1.00 application note.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	This function initializes port direction registers for ports that do not exist.
Arguments	None
Return values	None
Remarks	The number of pins in the sample code is set for the 176-pin package (PIN_SIZE=176). After this function is called, when writing in byte units to the PDR and 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. For more information on this function, refer to the RX64M Group Initial Setting Rev. 1.00 application note.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	This function initializes the clocks.
Arguments	None
Return values	None
Remarks	In the sample code, the PLL clock is selected as the system clock, and the sub-clock is not used. For more information on this function, refer to the RX64M Group Initial Setting Rev. 1.00 application note.

peripheral_init	
Outline	Peripheral function initialization
Header	None
Declaration	static void peripheral_init(void)
Description	This function initializes the peripheral functions used.
Arguments	None
Return values	None

cb_sci_tx_end	
Outline	Callback function (SCI transmission complete)
Header	None
Declaration	static void cb_sci_tx_end(void)
Description	This function is called when SCI transmission is complete.
Arguments	None
Return values	None

cb_sci_rx_end	
Outline	Callback function (SCI reception complete)
Header	None
Declaration	static void cb_sci_rx_end(void)
Description	This function is called when SCI reception is complete.
Arguments	None
Return values	None

cb_sci_rx_error	
Outline	Callback function (SCI reception error)
Header	None
Declaration	static void cb_sci_rx_error(void)
Description	This function is called when an SCI reception error occurs.
Arguments	None
Return values	None
Remarks	Error processing is not performed in the sample code. Add error processing when necessary.

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

SCI_StartReceive	
Outline	User interface function (start SCI reception)
Header	sci.h
Declaration	uint8_t SCI_StartReceive(uint8_t * pbuf, uint8_t num, CallBackFunc pcb_rx_end, CallBackFunc pcb_rx_error)
Description	This function starts SCI reception.
Arguments	• uint8_t * pbuf: Pointer for the receive data storage • uint8_t num: Number of bytes to be received • CallBackFunc pcb_rx_end: Pointer for the callback function (reception complete) • CallBackFunc pcb_rx_error: Pointer for the callback function (reception error)
Return values	• SCI_NG: Argument error (number of bytes to be received is 0) • SCI_BUSY: SCI receiving data • SCI_OK: Start SCI reception

SCI_StartTransmit	
Outline	User interface function (start SCI transmission)
Header	sci.h
Declaration	uint8_t SCI_StartTransmit(const uint8_t * pbuf, uint8_t num, CallBackFunc pcb_tx_end)
Description	This function starts SCI transmission.
Arguments	• const uint8_t * pbuf: Pointer for the transmit data storage • uint8_t num: Number of bytes to be transmitted • CallBackFunc pcb_tx_end: Pointer for the callback function (transmission complete)
Return values	• SCI_NG: Argument error (number of bytes to be transmitted is 0) • SCI_BUSY: SCI transmitting data • SCI_OK: Start SCI transmission

SCI_GetState	
Outline	User interface function (obtain SCI status)
Header	sci.h
Declaration	sci_state_t SCI_GetState(void)
Description	This function returns the SCI status.
Arguments	None
Return values	• sci_state_t.bit.b_tx_busy: Transmission busy flag 0: Ready to transmit data 1: Transmitting data • sci_state_t.bit.b_rx_busy: Reception busy flag 0: Ready to receive data 1: Receiving data • sci_state_t.bit.b_rx_orer: Overrun error flag 0: Overrun error did not occur 1: Overrun error occurred • sci_state_t.bit.b_rx_fer: Framing error flag 0: Framing error did not occur 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 function. Transmit data is written. When the last data is transmitted, the TXI interrupt request is disabled, and the TEI interrupt request is enabled.
Arguments	None
Return values	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 function. The TEI interrupt request is disabled, and the callback function (SCI transmission complete) is called.
Arguments	None
Return values	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 function. Received data is stored. When the last data is received, the callback function (SCI reception complete) is called.
Arguments	None
Return values	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 function. Serial reception and serial transmission are disabled, and the callback function (SCI reception error) is called.
Arguments	None
Return values	None

Excep_SCIn_ERIn	
Outline	SCI.ERI interrupt handling
Header	None
Declaration	static void Excep_SCIn_ERIn(void)
Description	This function performs receive error interrupt handling.
Arguments	None
Return values	None

Excep_SCIn_RXIn	
Outline	SCI.RXI interrupt handling
Header	None
Declaration	static void Excep_SCIn_RXIn(void)
Description	This function performs receive data full interrupt handling.
Arguments	None
Return values	None

Excep_SCIn_TXIn	
Outline	SCI.TXI interrupt handling
Header	None
Declaration	static void Excep_SCIn_TXIn(void)
Description	This function performs transmit data empty interrupt handling.
Arguments	None
Return values	None

Excep_SCIn_TEIn	
Outline	SCI.TEI interrupt handling
Header	None
Declaration	static void Excep_SCIn_TEIn(void)
Description	This function performs transmit end interrupt handling.
Arguments	None
Return values	None

Excep_ICU_GROUPBLn	
Outline	Group BL0 interrupt handling
Header	None
Declaration	static void Excep_ICU_GROUPBL0(void)
Description	This function performs group BL0 interrupt handling.
Arguments	None
Return values	None

5.10 Flowcharts

5.10.1 Main Processing

Figure 5.5 shows the Main Processing.

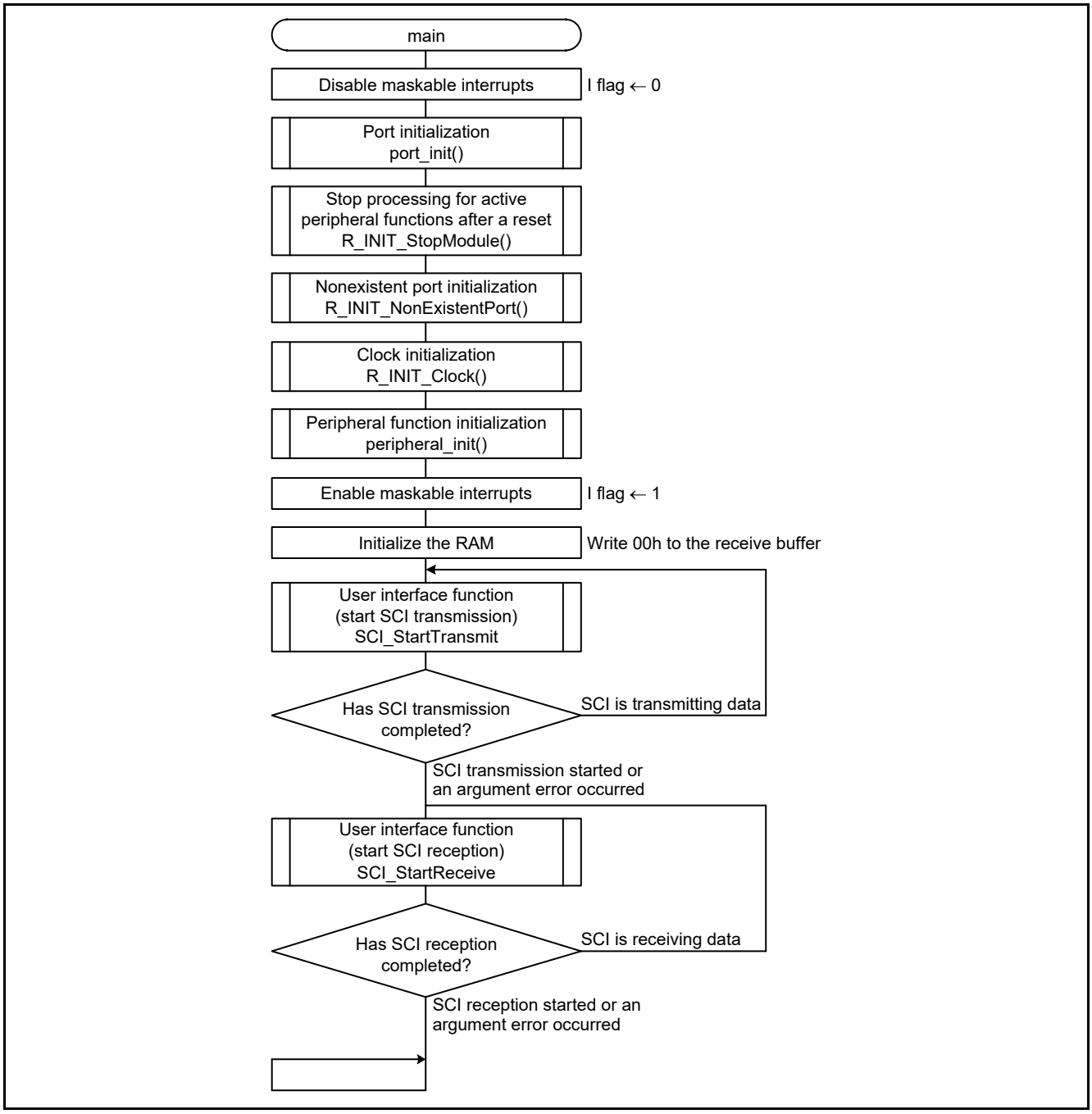


Figure 5.5 Main Processing

5.10.2 Port Initialization

Figure 5.6 shows Port Initialization.

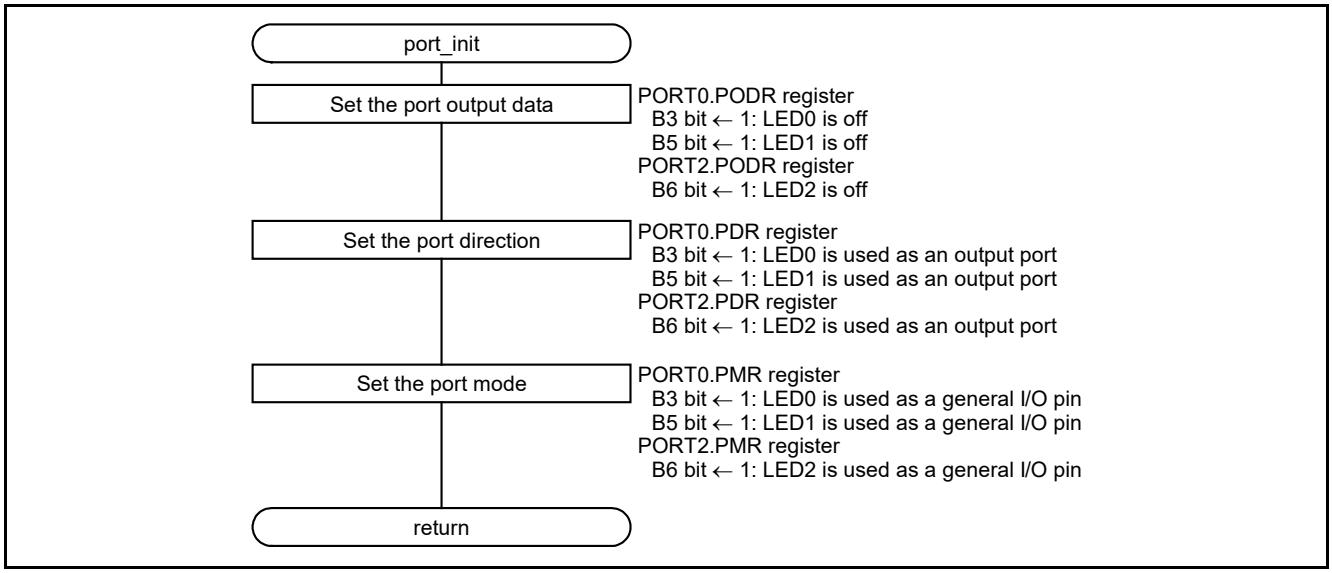


Figure 5.6 Port Initialization

5.10.3 Peripheral Function Initialization

Figure 5.7 shows Peripheral Function Initialization.

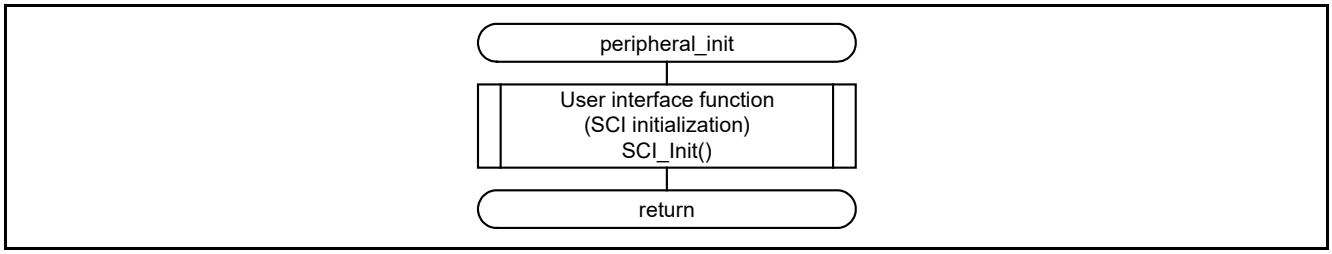


Figure 5.7 Peripheral Function Initialization

5.10.4 Callback Function (SCI Transmission Complete)

Figure 5.8 shows the Callback Function (SCI Transmission Complete).

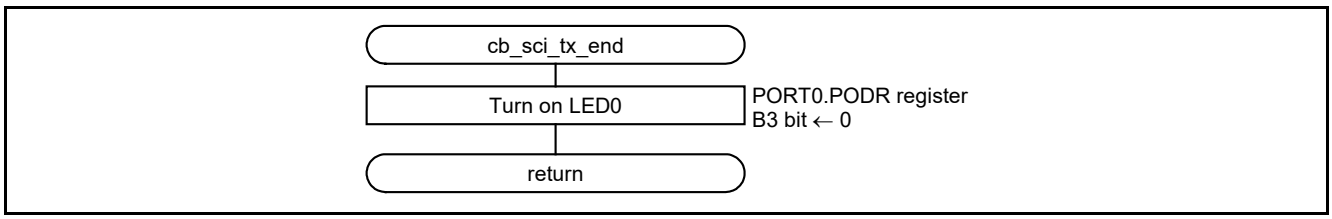


Figure 5.8 Callback Function (SCI Transmission Complete)

5.10.5 Callback Function (SCI Reception Complete)

Figure 5.9 shows the Callback Function (SCI Reception Complete).

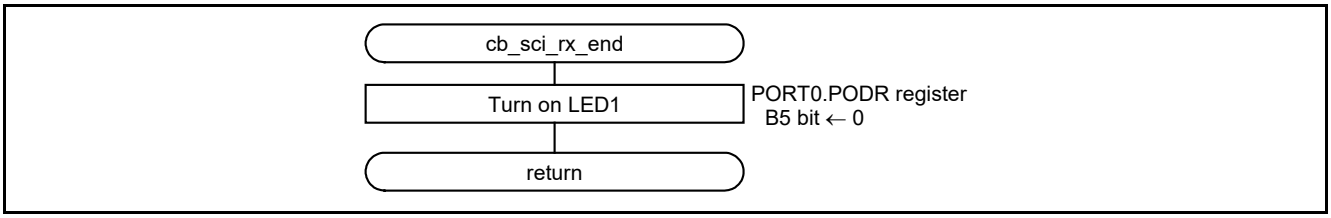


Figure 5.9 Callback Function (SCI Reception Complete)

5.10.6 Callback Function (SCI Reception Error)

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

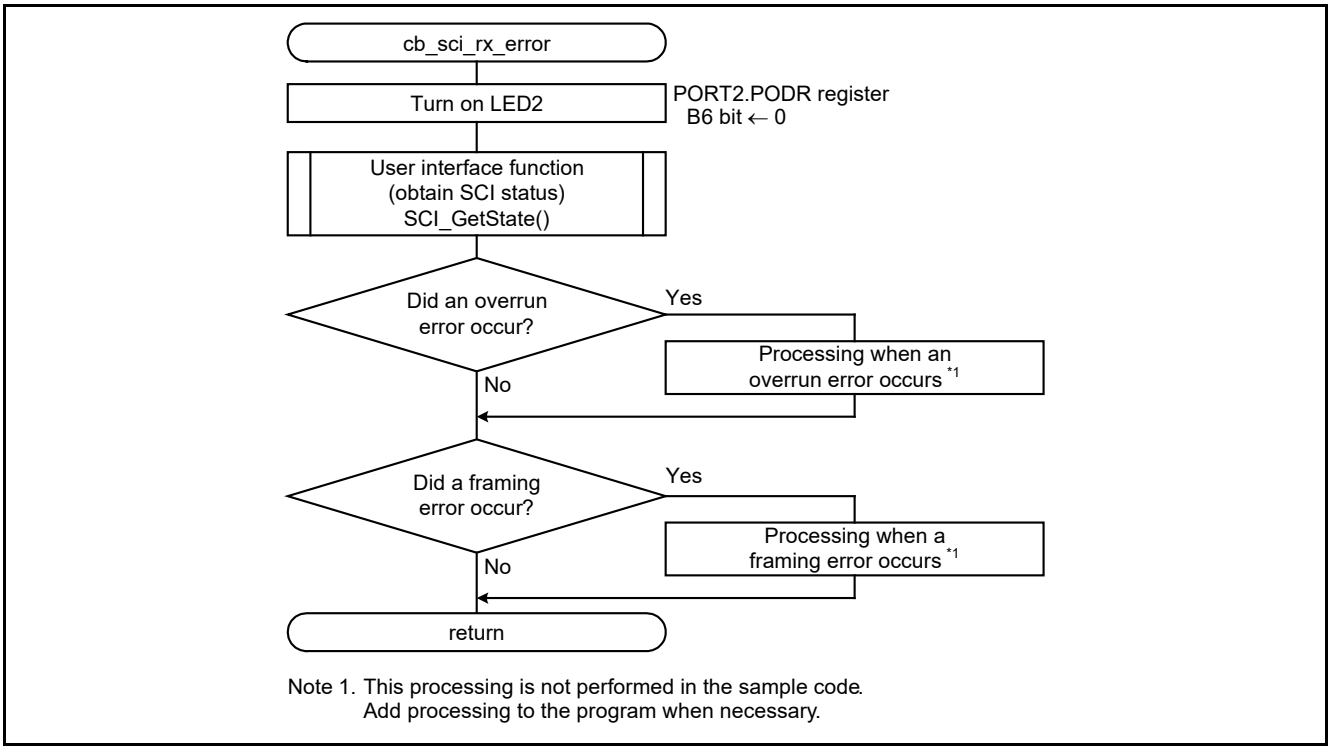


Figure 5.10 Callback Function (SCI Reception Error)

5.10.7 User Interface Function (SCI Initialization)

Figure 5.11 and Figure 5.12 show the user interface function (SCI initialization).

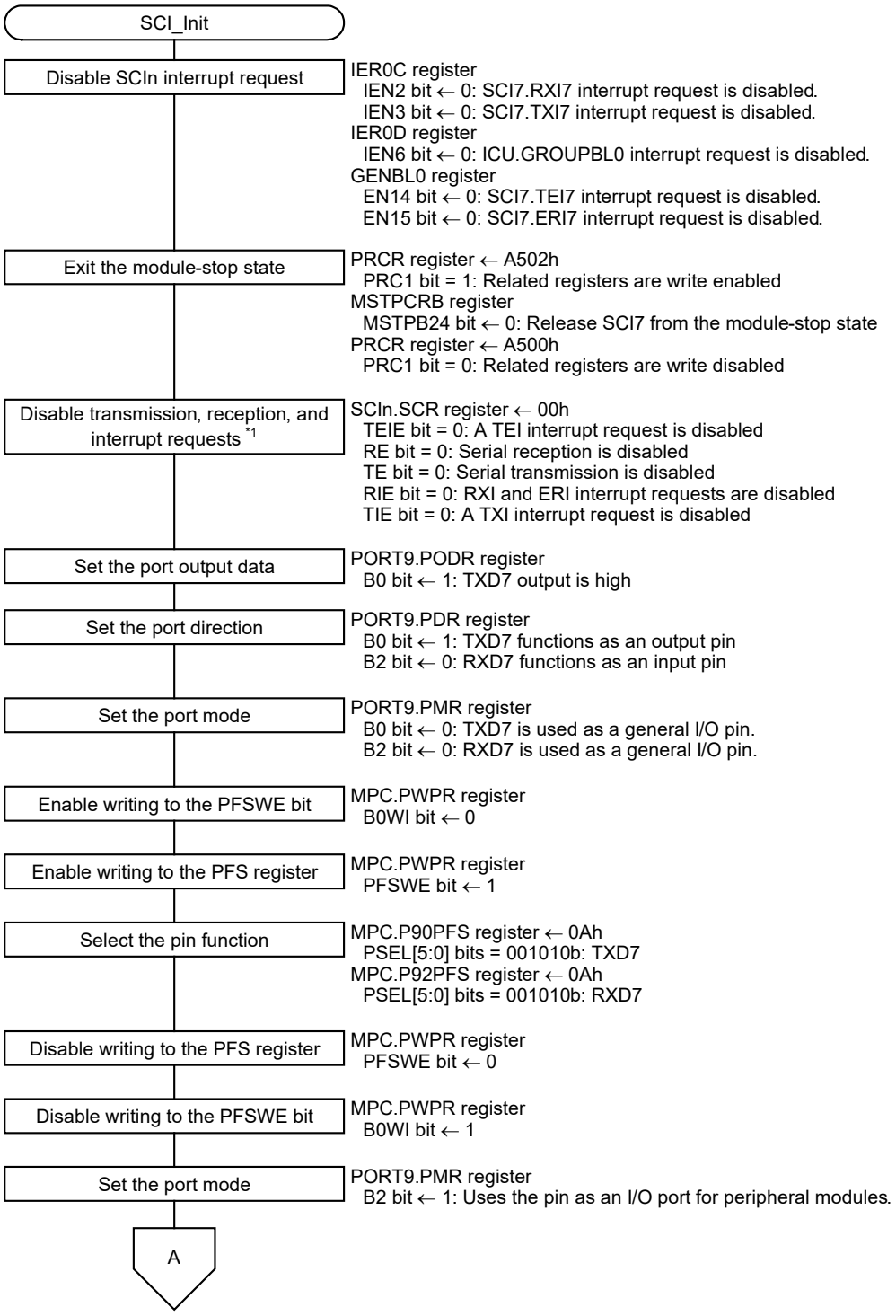
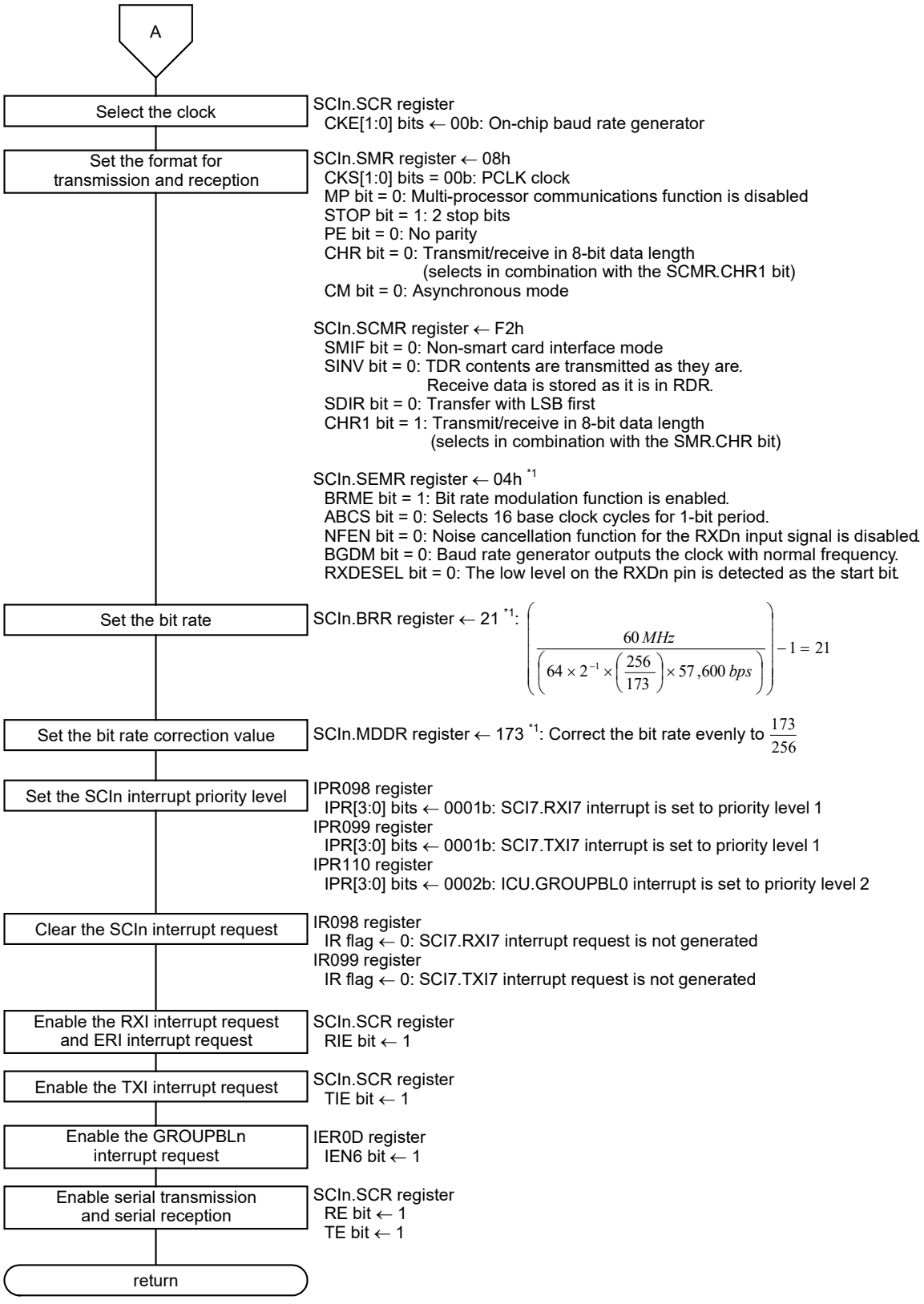


Figure 5.11 User Interface Function (SCI Initialization) (1/2)



Note 1. Setting value when using the bit rate modulation function.

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

5.10.8 User Interface Function (Start SCI Reception)

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

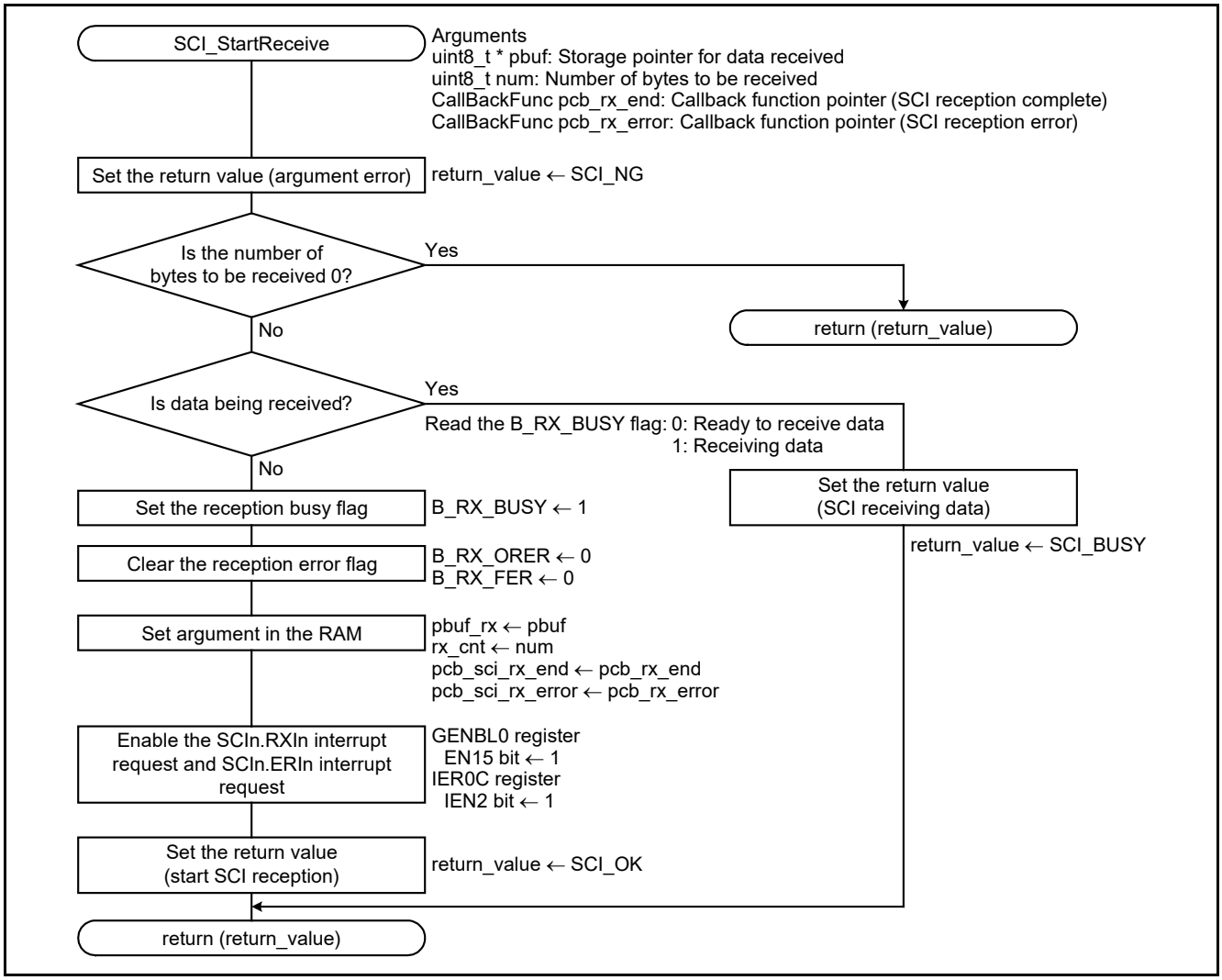


Figure 5.13 User Interface Function (Start SCI Reception)

5.10.9 User Interface Function (Start SCI Transmission)

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

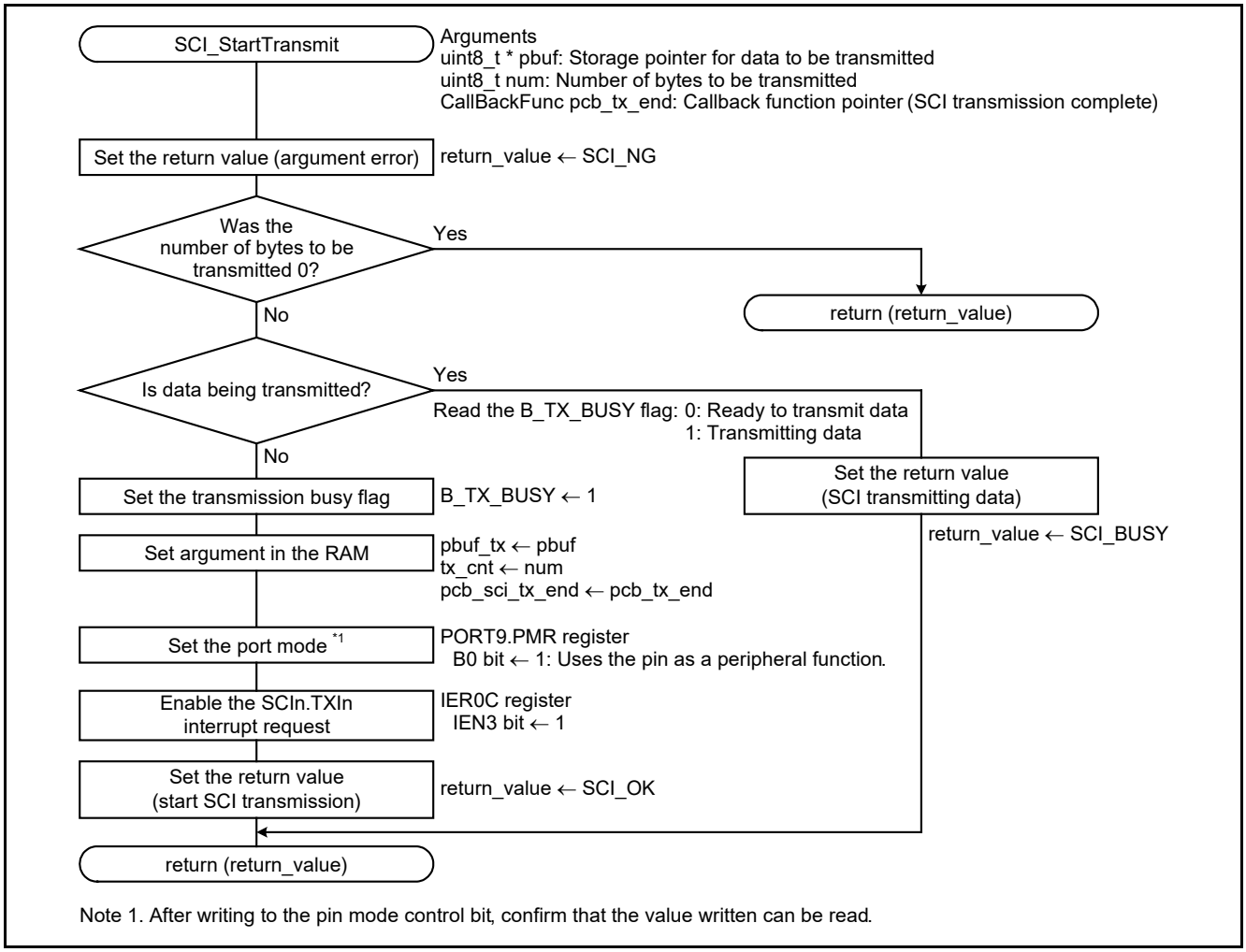


Figure 5.14 User Interface Function (Start SCI Transmission)

5.10.10 User Interface Function (Obtain SCI Status)

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

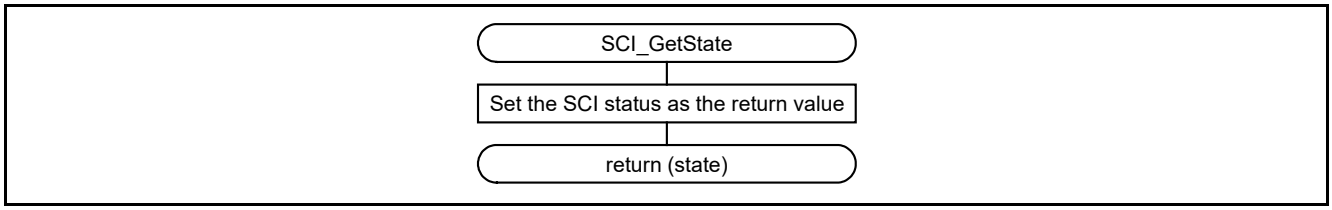


Figure 5.15 User Interface Function (Obtain SCI Status)

5.10.11 Transmit Data Empty Interrupt

Figure 5.16 shows the Transmit Data Empty Interrupt.

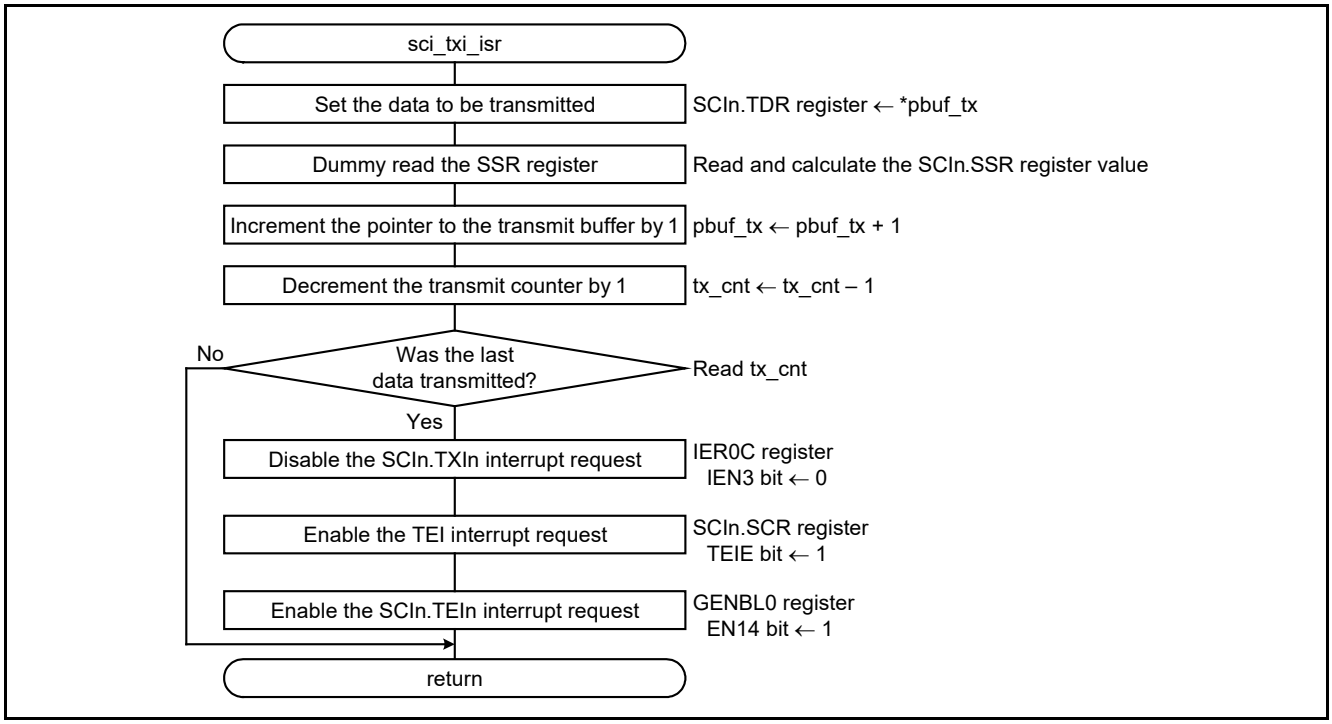


Figure 5.16 Transmit Data Empty Interrupt

5.10.12 Transmit End Interrupt

Figure 5.17 shows the Transmit End Interrupt.

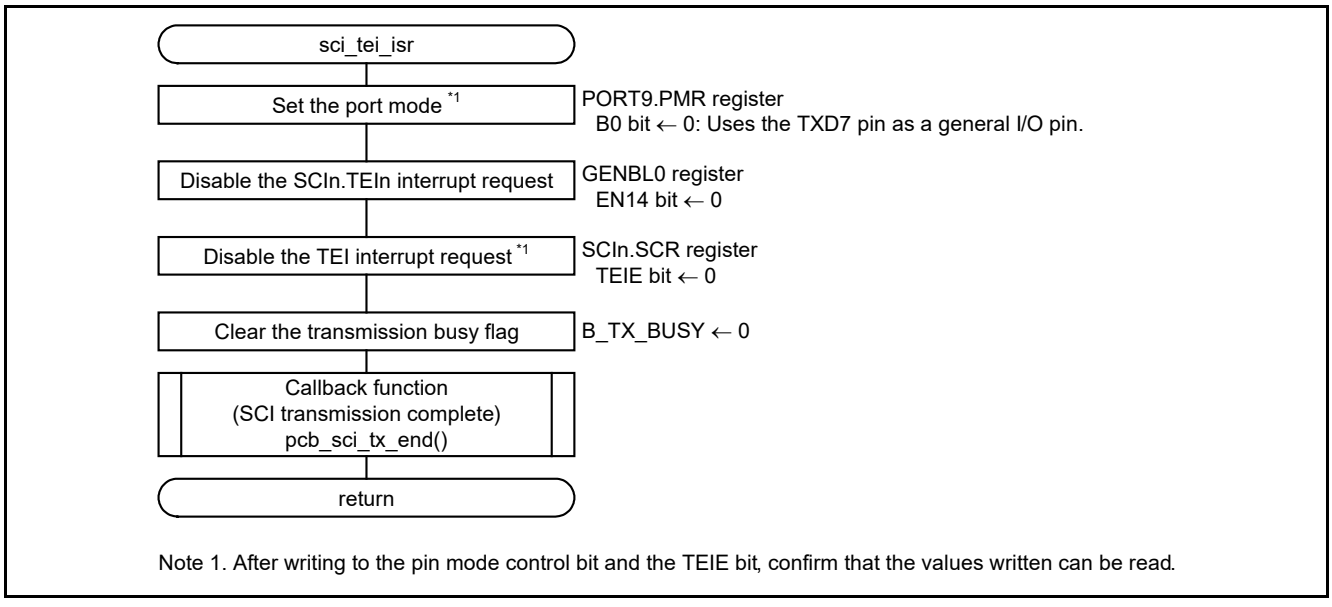


Figure 5.17 Transmit End Interrupt

5.10.13 Receive Data Full Interrupt

Figure 5.18 shows the Receive Data Full Interrupt.

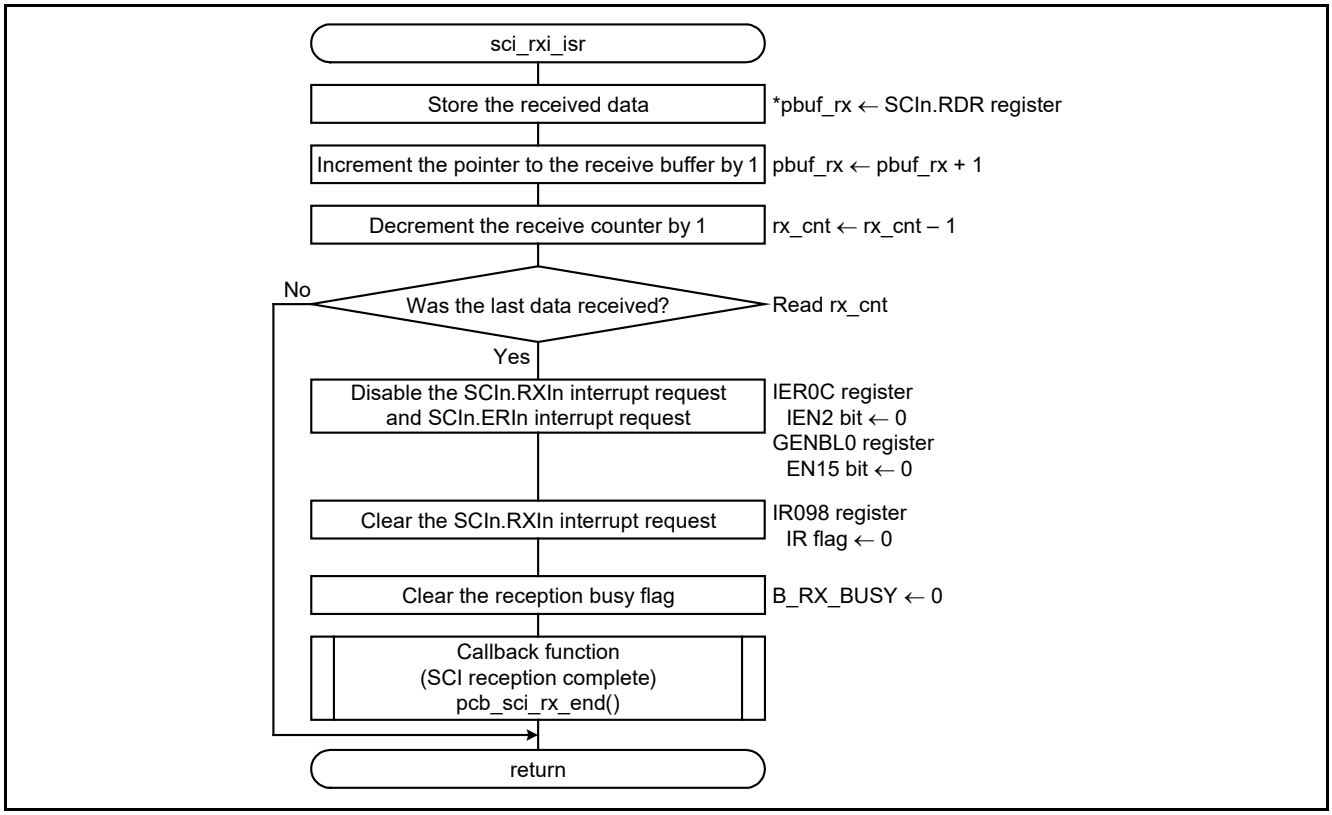
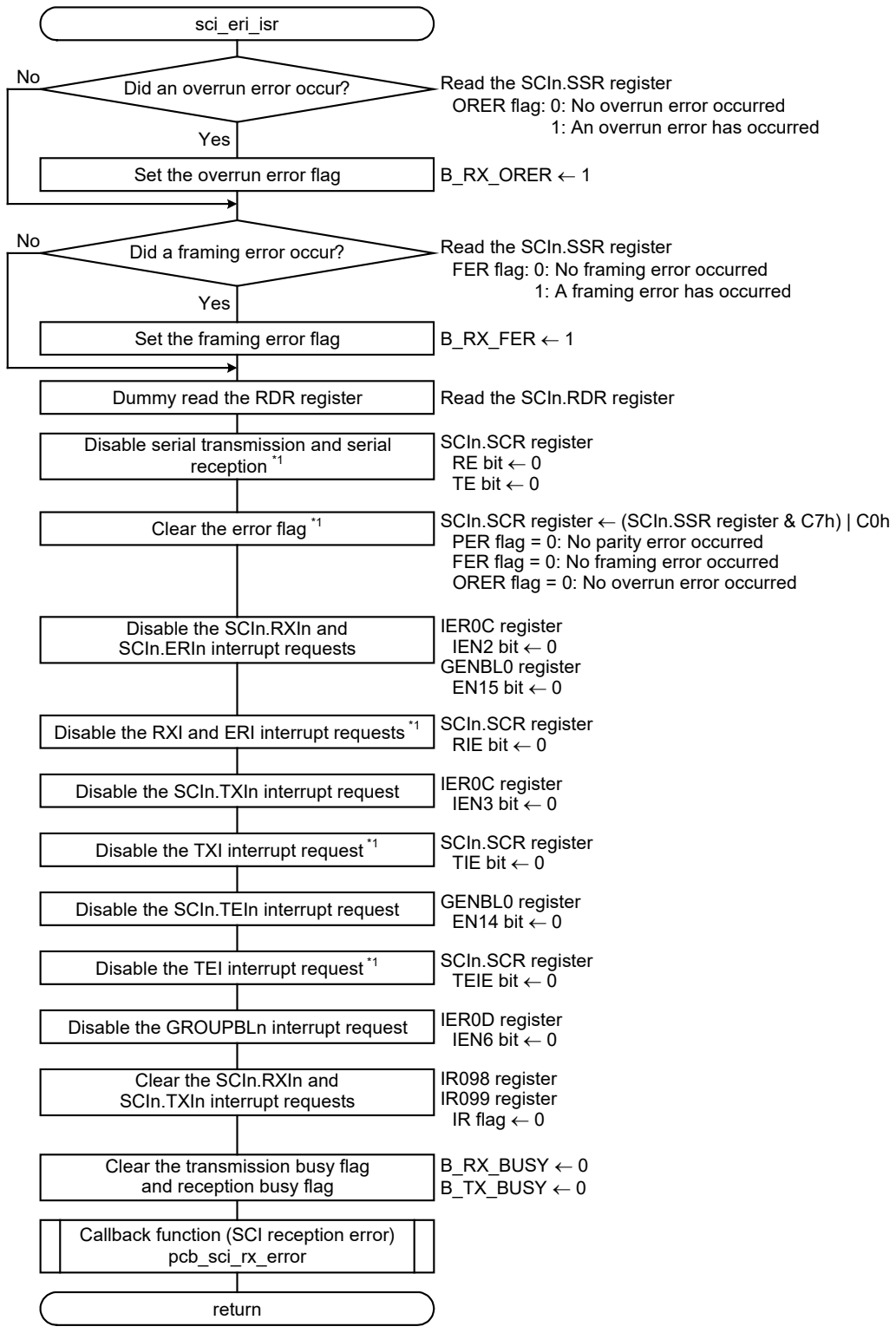


Figure 5.18 Receive Data Full Interrupt

5.10.14 Receive Error Interrupt

Figure 5.19 shows the Receive Error Interrupt.



Note 1. After writing to the RE bit, RIE bit, PER flag, FER flag, and ORER flag, confirm that the values written can be read.
After writing to bits TIE and TEIE, confirm that the values written can be read.

Figure 5.19 Receive Error Interrupt

5.10.15 SCI.ERI Interrupt Handling

Figure 5.20 shows SCI.ERI Interrupt Handling.

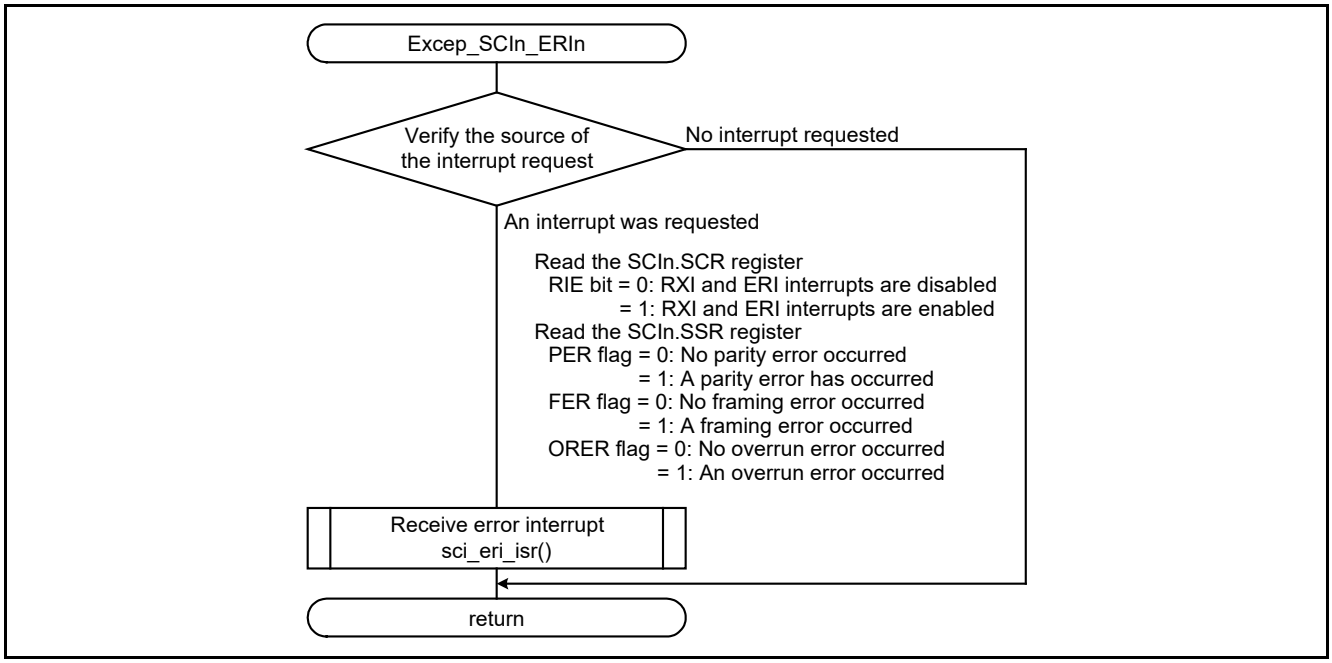


Figure 5.20 SCI.ERI Interrupt Handling

5.10.16 SCI.RXI Interrupt Handling

Figure 5.21 shows SCI.RXI Interrupt Handling.

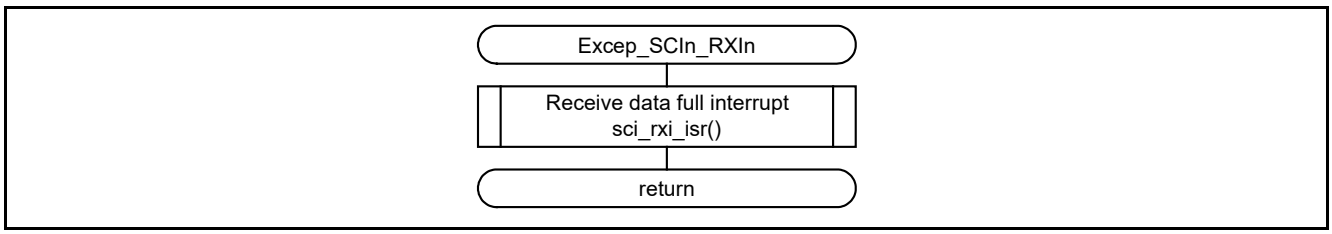


Figure 5.21 SCI.RXI Interrupt Handling

5.10.17 SCI.TXI Interrupt Handling

Figure 5.22 shows SCI.TXI Interrupt Handling.

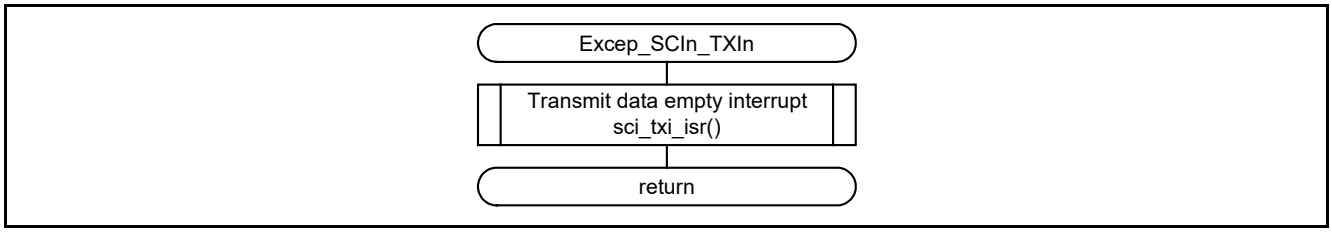


Figure 5.22 SCI.TXI Interrupt Handling

5.10.18 SCI.TEI Interrupt Handling

Figure 5.23 shows SCI.TEI Interrupt Handling.

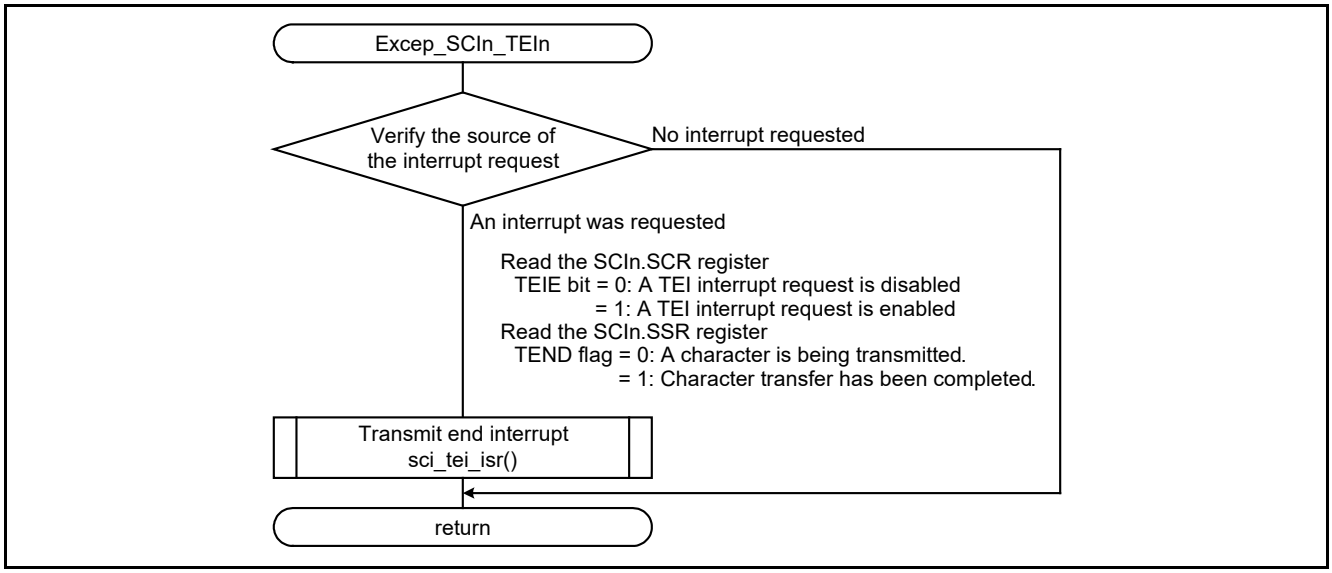


Figure 5.23 SCI.TEI Interrupt Handling

5.10.19 Group BL0 Interrupt Handling

Figure 5.24 shows Group BL0 Interrupt Handling.

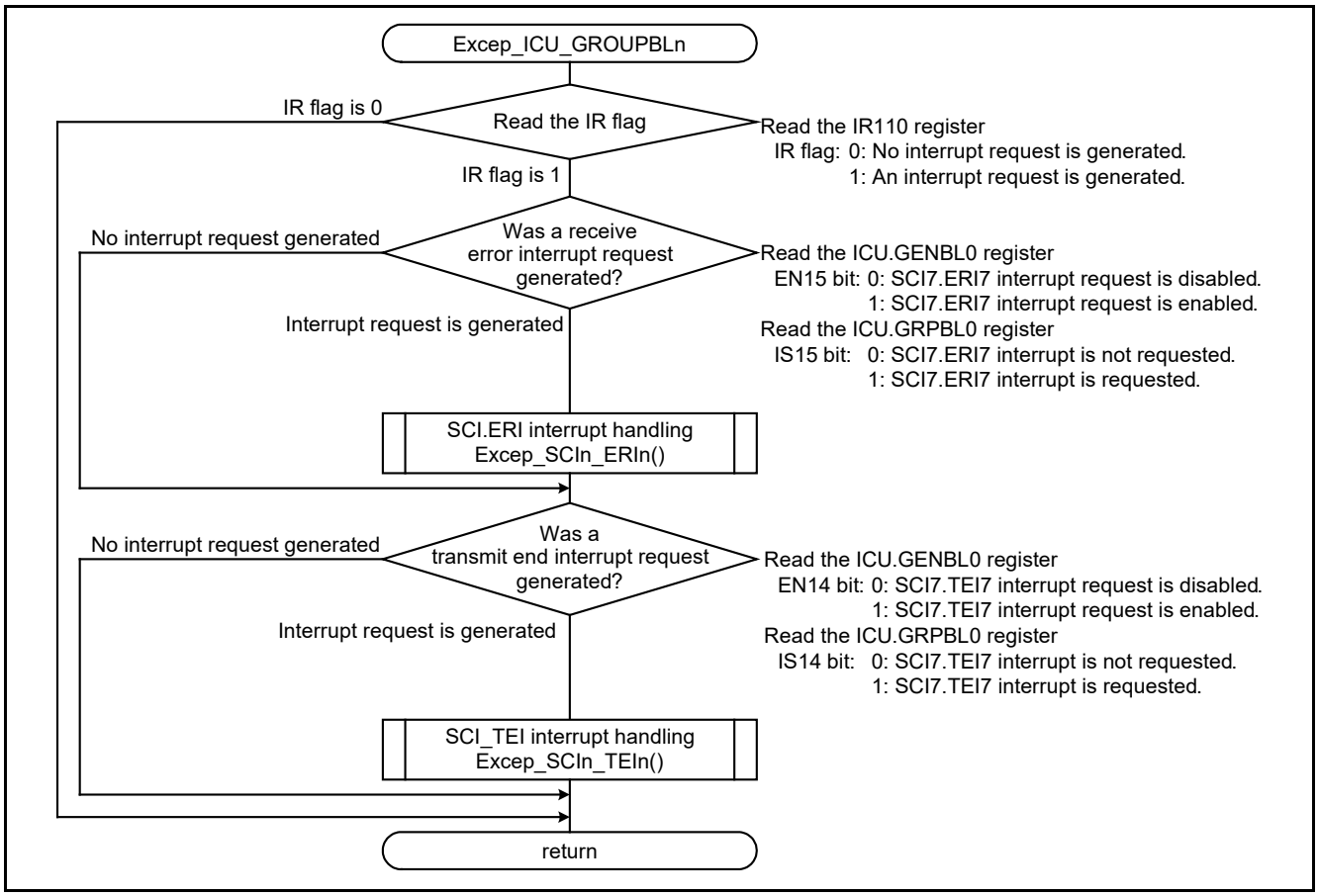


Figure 5.24 Group BL0 Interrupt Handling

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User's Manual: Hardware
RX64M Group User's Manual: Hardware (R01UH0377EJ)
RX71M Group User's Manual: Hardware (R01UH0493EJ)
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
[e^2 studio] RX Family Compiler CC-RX V.2.01.00 User's Manual: RX Coding Rev.1.00 (R20UT2748EJ)
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	RX64M, RX71M Group Application Note Using the SCIg Bit Rate Modulation Function

Rev.	Date	Description	
		Page	Summary
1.00	Oct. 1, 2014	—	First edition issued
1.01	Nov. 2, 2015	—	RX71M Group is added to the target device
		25	“Has SCI transmission started?” is changed to ” Has SCI transmission completed?” in table 5.5
			“Has SCI reception started?” is changed to ” Has SCI reception completed?” in table 5.5
1.10	Dec. 21, 2020	—	Update the toolchain version.

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

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

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

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

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

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

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

Contact information

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