To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

SH7145F

Synchronous Serial Data Reception

Summary

The SH7144 series is a single-chip microprocessor based on the SH-2 RISC (Reduced Instruction Set Computer) CPU core and integrating a number of peripheral functions.

This application note describes synchronous serial data reception using the SCI (Serial Communication Interface) module of the SH7145F. It is intended to be used as reference by users designing software applications.

The program examples contained in this application note have been tested. However, operation should be confirmed before using them in an actual application.

Device for Which Operation Has Been Confirmed

SH7145F

Contents

1. Specifications

As shown in figure 1, synchronous serial data reception is performed using channel 3 (ch3) of the SCI module of the SH7145F. In this task example 3 bytes of serial data are received by the SH7145. The communication format is 8-bit fixed data length, LSB first, with a logical bit rate of 250,000 bps.

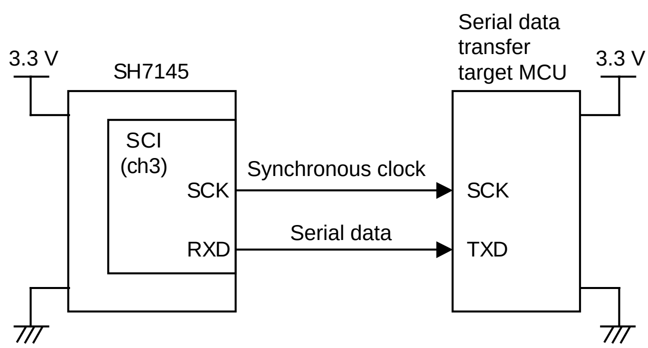


Figure 1 Synchronous Serial Data Reception by SH7145

2. Functions Used

In this task example the SCI (Serial Communication Interface) is used to perform synchronous serial data reception. Figure 2 shows a block diagram of channel 3 (ch3) of the SCI module. The functions of the elements shown in figure 2 are described below.

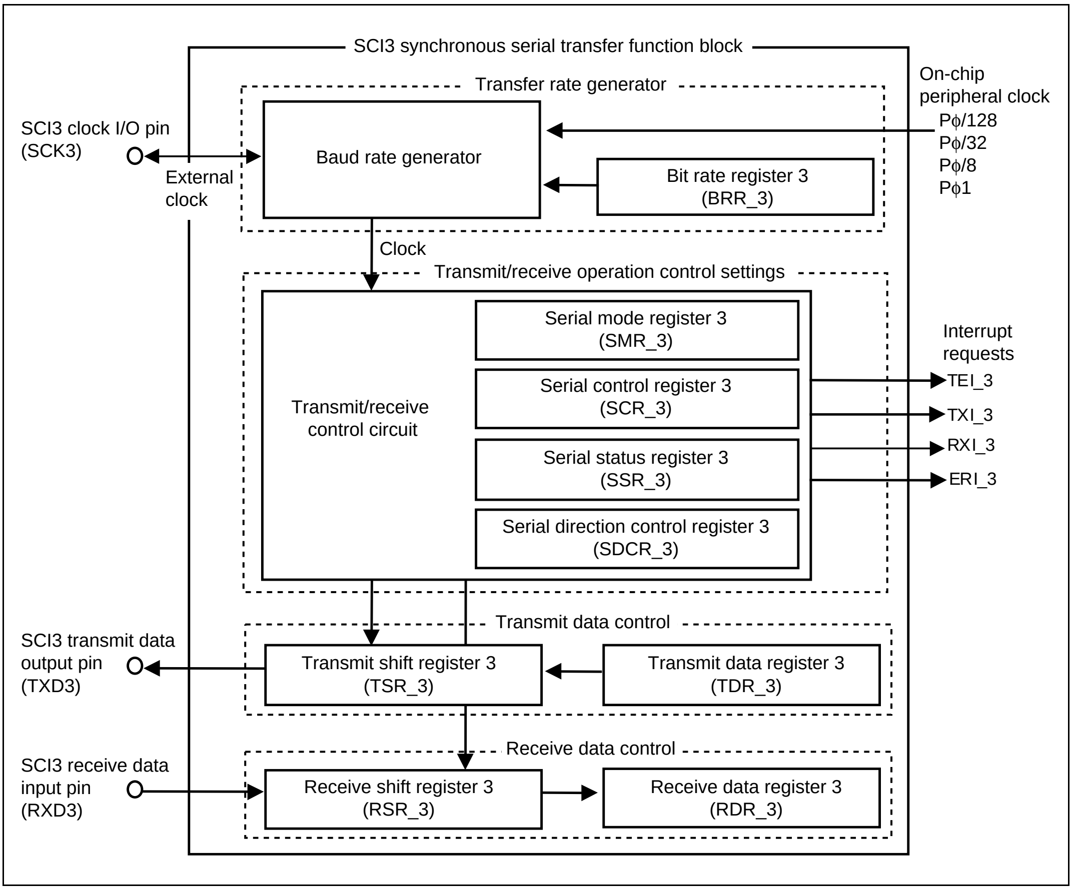


Figure 2 SCI (ch3) Block Diagram

- Synchronous Mode

  Data is transmitted and received in synchronization with clock pulses. The transfer data length is fixed at 8 bits.

- On-Chip Peripheral Clock Pϕ

  This is the reference clock for operation of on-chip peripheral functions. The clock signal is generated by a clock oscillator.

- Receive Shift Register (RSR_3)

  This register is used to receive serial data. Serial data is input to RSR_3 from the RxD_3 pin. When one frame of data has been received, it is automatically transferred to the receive data register (RDR_3). RSR_3 cannot be accessed by the CPU.

- Receive Data Register (RDR_3)

  Received data is stored in this 8-bit register. When one frame of data has been received, it is automatically transferred from RSR_3. RSR_3 and RDR_3 are in a double-buffer configuration, allowing continuous reception of data. RDR_3 is a receive-only register, so it can only be read by the CPU.

- Transmit Shift Register (TSR_3)

  This register is used to transmit serial data. In order to transmit data, the data is first transferred from the transmit data register (TDR_3) to TSR_3. Then the transmit data is output from the TxD_3 pin. TSR_3 cannot be accessed directly by the CPU.

- Transmit Data Register (TDR_3)

  Data to be transmitted is stored in this 8-bit register. When it is detected that TDR_3 is empty, data that has been written to TDR_3 is automatically transferred to TSR_3. TDR_3 and TSR_3 are in a double-buffer configuration. This allows data to be transferred to TSR_3 after one frame of data has been transmitted and the next frame of data is still being written to TDR_3, making possible continuous transmission of data. It is always possible to read or write to the TDR from the CPU, but before writing to the TDR it should be confirmed that the value of the TDRE bit in the serial status register (SSR_3) is 1.

- Serial Mode Register (SMR_3)

  This 8-bit register is used to select the serial data communication format and the clock source for the on-chip baud rate generator.

- Serial Control Register (SCR_3)

  This register is used for transmit and receive control, interrupt control, and to select the transmit and receive clock source.

- Serial Status Register (SSR_3)

  This register comprises the SCI3 status flag and the transmit and receive multiprocessor bits. TDRE, RDRF, ORER, PER, and FER can be cleared only.

- Serial Direction Control Register (SDCR_3)

  This register is used to select whether the LSB or MSB is first. For 8-bit communication either LSB-first or MSB-first may be selected, but LSB-first should be used for 7-bit communication.

- Bit Rate Register (BRR_3)

  This 8-bit register is used to adjust the bit rate. The SCI has independent baud rate generators for the individual channels, allowing different bit rates to be set for each. See the hardware manual for details on setting values, execution rate relationships, etc.

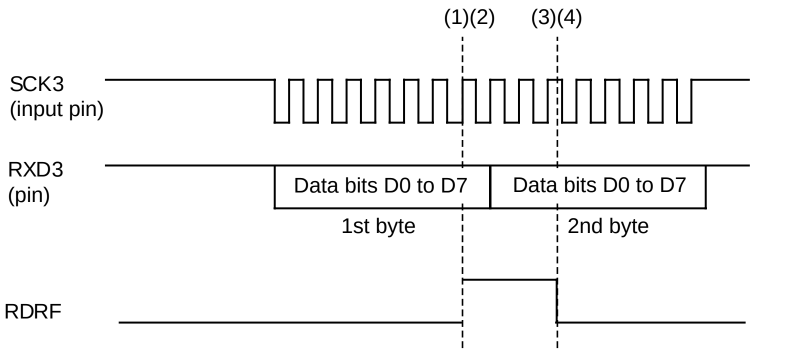
Table 1 shows the function allocations for the task example.

Table 1 Function Allocations

Function	Classification	Function Allocation
SCK3	Pin	Channel 3 synchronous clock I/O
TXD3	Pin	Channel 3 transmit data output pin
RXD3	Pin	Channel 3 receive data input pin
SMR_3	SCI3	Sets communication format to synchronous mode
SCR_3	SCI3	Enables receive operation
SSR_3	SCI3	Status flag showing SCI3 operation status
SDCR_3	SCI3	Specifies LSB-first
BRR_3	SCI3	Sets communication bit rate
TSR_3	SCI3	Register for transmitting serial data
TDR_3	SCI3	Register for storing transmit data
RSR_3	SCI3	Register for receiving serial data
RDR_3	SCI3	Register for storing receive data

3. Operation

Figure 3 shows the operation of synchronous mode data reception in the task example. To help explain figure 3, table 2 lists the software and hardware processing that is performed.



Note: To perform continuous reception, read data transferred to RDR until reception of next data is complete.

Figure 3 Data Reception Operation

Table 2 Processing

	Software Processing	Hardware Processing
(1)	—	RSR_3 receives serial data and transfers it to RDR_3
(2)	—	Set RDRF flag to 1
(3)	Read data from RDR_3	—
(4)	Clear RDRF flag in SSR_3 to 0	—
(5)	Repeat	Repeat

4. Software

(1) Module Descriptions

Table 3 lists the modules used in the task example.

Table 3 Module Descriptions

Module	Label	Function
Main routine	main	Calls modules
SCI routine	init_sci	Initial settings of SCI0
Transfer routine	rcv_sci	Receives serial data

(2) Argument Descriptions

Table 4 lists the arguments used in the task example.

Table 4 Argument Descriptions

Argument	Function	Module
TRANS_DATA_1	Transmits data from SCI_0	Transfer routine
TRANS_DATA_2	Transmits data from SCI_0	Transfer routine
TRANS_DATA_3	Transmits data from SCI_0	Transfer routine

(3) On-Chip Register Descriptions

Table 5 lists the on-chip registers used in the task example. The set values shown are the values used in the task example and differ from the initial settings.

Table 5 On-Chip Register Descriptions

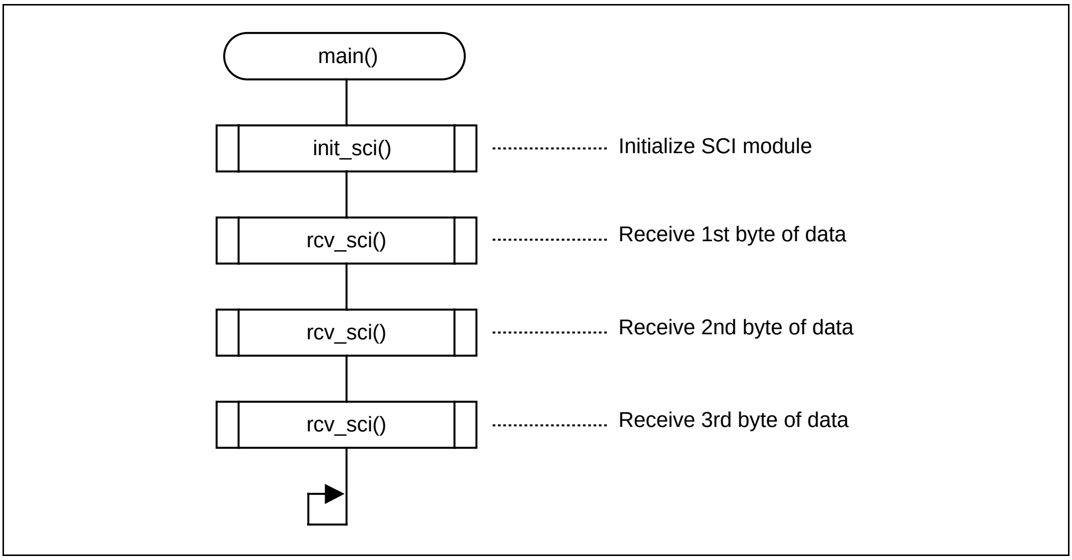
Register	Bit	Set Value	Function
MSTCR1	MSTP16	0	Module standby control register 1 SCI0 standby control bit Standby cancelled when MSTP16 = 0
SCR_0		H'20	Serial control register 0 (SCI_0) Transmit and receive control, interrupt control, transmit and receive clock source control
	TIE	0	Transmit interrupt enable TXI interrupt requests enabled when set to 1
	RIE	0	Receive interrupt enable RXI and ERI interrupt requests enabled when set to 1
	TE	1	Transmit enable Transmit operations enabled when set to 1
	RE	0	Receive enable Receive operations enabled when set to 1
	MPIE	0	Multiprocessor interrupt enable (In asynchronous mode, enabled when MP = 1 in SMR) In the task example, disabled because MP = 0
	TEIE	0	Transmit end interrupt enable TEI interrupt requests enabled when set to 1
	CKE1 CKE2	0 0	Clock enable 1, 0 Selects clock source and SCK pin function In the task example, clock source is on-chip clock and SCK pin is not used
SMR_0		H'00	Serial mode register 0 Selects communication format and the clock source for on-chip baud rate generator
	C/A	0	Communication mode Synchronous mode when cleared to 0
	CHR	0	Character length (enabled in asynchronous mode only) Disabled in task example because synchronous mode is used
	PE	0	Parity enable (enabled in asynchronous mode only) Disabled in task example because synchronous mode is used
	O/E	0	Parity mode (enabled in asynchronous mode when PE = 1) Disabled in task example because synchronous mode is used
	STOP	0	Stop bit length (enabled in asynchronous mode only) Disabled in task example because synchronous mode is used

Register	Bit	Set Value	Function
SMR_0	MP	0	Multiprocessor mode (enabled in asynchronous mode only) Disabled in task example because synchronous mode is used
	CKS1 CKS2	0 0	Clock select 1, 0 When value is 00, Pϕ clock selected using on-chip baud rate generator as clock source
BRR_0		H'40	Bit rate register 0 8-bit register for adjusting bit rate
SDCR_0		H'F2	Serial direction control register 0 DIR bit (bit 3) selects LSB-first or MSB-first In task example, DIR = 0 (LSB-first)
SSR_0		H'xx	Serial status register 0 Comprises SCI0 status flag and transmit and receive multiprocessor bits Only 0 may be written to the status flag, to clear it
	TDRE	*	Transmit data register empty (status flag)
	RDRF	*	Receive data register full (status flag)
	ORER	*	Overrun error (status flag)
	FER	*	Framing error (status flag)
	PER	*	Parity error (status flag)
	TEND	*	Transmit end (status flag)
	MPB	0	Multiprocessor bit
	MPBT	0	Multiprocessor bit transfer
PACRL2	PA2MD1 PA2MD0	0 1	Port A control register L2 Function setting for port A multiplex pin (SCK0)
	PA1MD1 PA1MD0	0 1	Port A control register L2 Function setting for port A multiplex pin (TXD0)

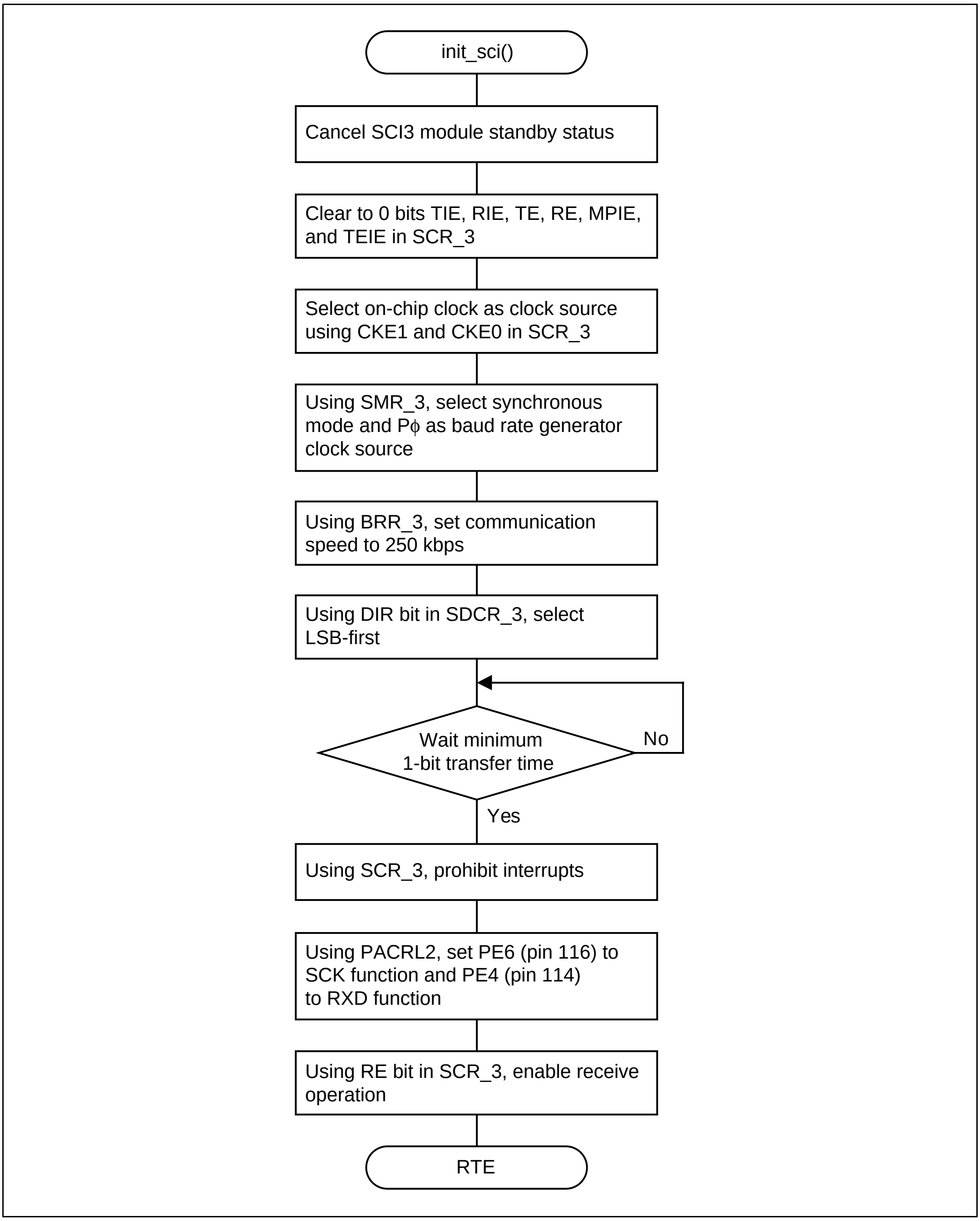
*: Can only be cleared to 0. Setting to 1 is performed by hardware.

5. Flowcharts

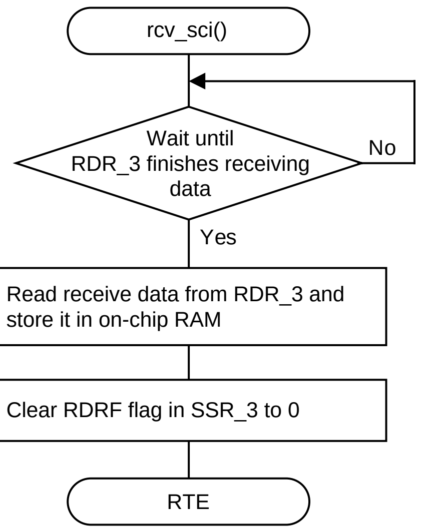
(1) Main Routine



(2) SCI1 Initialize Routine



(3) Data Receive Routine



6. Program Listing

```
/************************************************************/
/* SH7145F Application Note                                 */
/*                                                          */
/* Function                                                 */
/*    :SCI3                                                 */
/*    :Clock Synchronous Receive Mode                       */
/*                                                          */
/* External input clock         :12.5MHz                    */
/* Internal CPU clock           :50MHz                      */
/* Internal peripheral clock    :25MHz                      */
/*                                                          */
/* Written                      :2003/7     Rev.1.0         */
/************************************************************/

#include "iodefine.h"
#include <machine.h>

/*------------ Symbol Definition ----------------------------------------------*/

#define COUNT 3

/*------------ Function Definition --------------------------------------------*/
void main(void);

void init_sci(void);
unsigned char rcv_sci(unsigned char);

void err_int(void);
void dummy_f(void);

/*------------ RAM allocation Definition --------------------------------------*/

volatile unsigned char Rev_data[COUNT];

/************************************************************/
/* main Program                                             */
/************************************************************/
void main( void )
{
   unsigned char i = 0;

   init_sci();             /* Initialize SCI             */

   i = rcv_sci(i);         /* Receive 1st byte of data   */

   i = rcv_sci(i);         /* Receive 2nd byte of data   */

   i = rcv_sci(i);         /* Receive 3rd byte of data   */
```

```
   while(1);               /* LOOP                        */
}

/***************************************************************/
/* Function  : init_sci                                        */
/* Operation : Initialize serial (sci3)                        */
/*      sci3 : Synchronous receive operation                   */
/***************************************************************/
void init_sci(void)
{
   unsigned long i;

   P_STBY.MSTCR1.BIT.MSTP19 = 0;  /* disable SCI3 standby mode       */

   /* Initialize SCI       */
   P_SCI3.SCR_3.BYTE &= 0x03;     /* clear TIE,RIE,TE,RE,MPIE,TEIE   */
   P_SCI3.SCR_3.BIT.CKE = 3;      /* clock:external,SCK:input     */
   P_SCI3.SMR_3.BYTE = 0x80;      /* Clock synchronous mode          */
             //   CA = 1;         /* Clock synchronous mode          */

   P_SCI3.BRR_3 = 24;             /* 250kbps@25MHz(Peripheral)       */

   for( i=0; i < 0x0300 ; i++);   /* Wait 1bit                       */

   P_SCI3.SCR_3.BIT.RIE = 0;      /* RXI_3,ERI_3 interrupt disable  */

   /* Initialize SCI3 PORT        */
   P_PORTE.PECRL2.BIT.PE6MD = 2;  /* set SCK3(PE6:116pin@SH7145)     */
   P_PORTE.PECRL2.BIT.PE4MD = 2;  /* set RXD3(PE4:114pin@SH7145)     */

   P_SCI3.SCR_3.BIT.RE = 1;       /* RE=1,Receive Enable(SCI3)       */
}

/*************************************************************************/
/* Function       : rcv_sci                                              */
/* Operation      : Store receive data in on-chip RAM                    */
/* Argument       : rev_count                                            */
/* Value returned : rev_count                                            */
/*************************************************************************/
unsigned char rcv_sci(unsigned char rev_count)
{
   while(P_SCI3.SSR_3.BIT.RDRF == 0);    /* Wait until reception is finished  */

   Rev_data[rev_count] = P_SCI3.RDR_3;   /* get receive data                  */

   P_SCI3.SSR_3.BIT.RDRF = 0;            /* Clear RDRF                        */

   rev_count++ ;                         /* Increment storage address         */

   return(rev_count);
}
```

```
/*************************************
         Interrupt handling
*************************************/
#pragma interrupt(err_int)
void err_int(void)
{
   if(P_SCI1.SSR_1.BIT.ORER == 1){      /* Overrun error               */
      P_SCI1.SSR_1.BIT.ORER = 0;        /* ORER flag clear             */
   }
}

#pragma interrupt(dummy_f)
void dummy_f(void)
{
      /*   Other Interrupt    */
}
```

7. Usage Notes

SCK continues to be output until RE is cleared to 0. To halt SCK output at a specified number of bytes, perform a dummy transfer in the transmit or receive mode.

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials