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.

H8/300H Tiny Series

Simultaneous Transmission/Reception in Asynchronous Mode

Introduction

Four bytes of 8-bit data are simultaneously transmitted and received by serial data transfer in asynchronous mode.

Target Device

H8/300H Tiny Series H8/3664

Contents

1. Specification

1. Four bytes of 8-bit data are simultaneously transmitted and received by serial data transfer in asynchronous mode, as shown in figure 1.
2. The data transfer format set for transmit data is a data length of eight bits, an odd parity, and a stop bit length of one bit.
3. The bit rate for transmission is 31250 (bit/s). SCI3 stops after four bytes of data have been received.

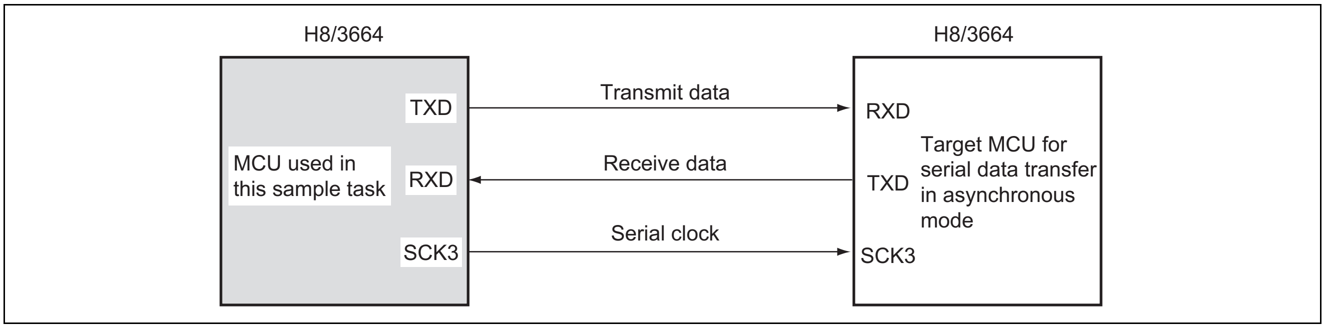
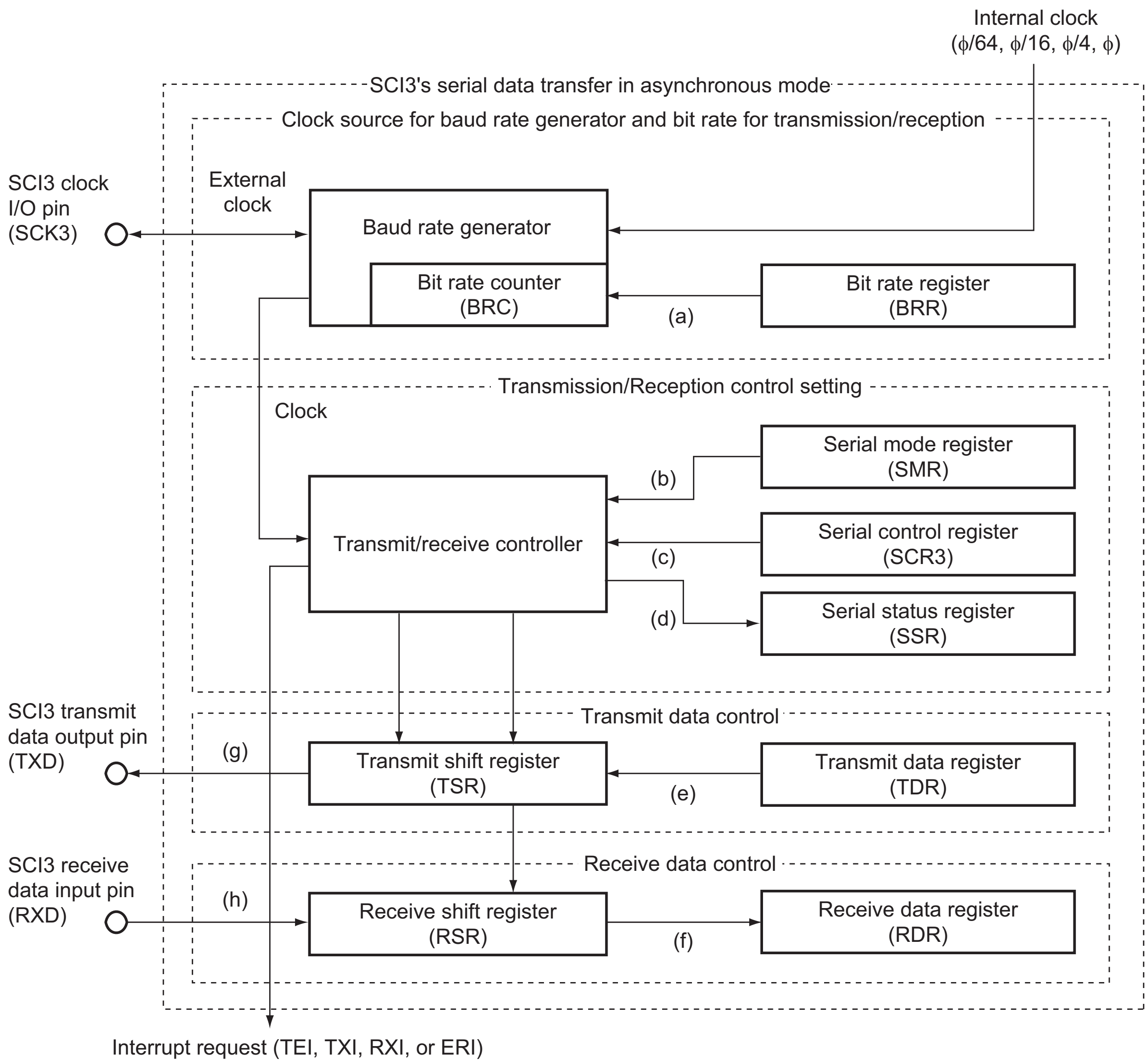


Figure 1 Simultaneous Serial Transmission/Reception in Asynchronous Mode

2. Description of Functions Used

1. In this sample task, serial data is simultaneously transmitted and received in asynchronous mode via the serial communication interface (SCI). Figure 2 is a block diagram of simultaneous serial transmission/reception in asynchronous mode. The elements of the block diagram are described below.

- In asynchronous mode, serial data communication is performed asynchronously, with synchronization provided character by character.
- In this mode, serial data can be exchanged with standard asynchronous communication LSIs such as a Universal Asynchronous Receiver/Transmitter (UART) or Asynchronous Communication Interface Adapter (ACIA).
- A multiprocessor communication function is also provided, enabling serial data communication among processors.
- There is a choice of 12 data transfer formats.
- Separate transmission and reception units are provided, enabling transmission and reception to be carried out simultaneously. The transmission and reception units are both double-buffered, allowing continuous transmission and reception.
- Any desired bit rate can be selected with the on-chip baud rate generator.
- An internal or external clock can be selected as the transmit/receive clock source.
- There are six interrupt sources: transmit end, transmit data empty, receive data full, overrun error, framing error, and parity error.
- The receive shift register (RSR) is a register used to receive serial data. Serial data input to RSR from the RXD pin is set in the order in which it is received, starting from the LSB (bit 0), and converted to parallel data. When one byte of data is received, it is transferred to RDR automatically. RSR cannot be read from or written to directly by the CPU.
- The receive data register (RDR) is an 8-bit register that stores received serial data. When reception of one byte of data is finished, the received data is transferred from RSR to RDR, and the receive operation is completed. RSR is then enabled for reception. RSR and RDR are double-buffered, allowing consecutive receive operations. RDR is a read-only register, and cannot be written to by the CPU.
- The transmit shift register (TSR) is a register used to transmit serial data. Transmit data is first transferred from TDR to TSR, and serial data transmission is carried out by sending the data to the TXD pin in order, starting from the LSB (bit 0). When one byte of data is transmitted, the next byte of transmit data is automatically transferred from TDR to TSR, and transmission is started. Data transfer from TDR to TSR is not performed if no data has been written to TDR (if bit TDRE is set to 1). TSR cannot be read from or written to directly by the CPU.
- The transmit data register (TDR) is an 8-bit register that stores transmit data. When TSR is found to be empty, the transmit data written in TDR is transferred to TSR, and serial data transmission is started. Continuous transmission is possible by writing the next transmit data to TDR during TSR serial data transmission. TDR can be read from or written to by the CPU at any time.
- The serial mode register (SMR) is an 8-bit register used to set the serial data transfer format and to select the clock source for the baud rate generator. SMR can be read from or written to by the CPU at any time.
- Serial control register 3 (SCR3) is an 8-bit register for selecting transmit or receive operation, the asynchronous mode clock output, to enable or disable interrupt requests, and the transmit/receive clock source. SCR3 can be read from or written to by the CPU at any time.



Notes: (a) Sets bit rate for transmission/reception according to clock source for baud rate generator set in SMR. In this sample task, transmission bit rate is set to 31250 (bit/s).
(b) Sets serial data transfer format and clock source for baud rate generator. In this sample task, communication mode is set to asynchronous mode, data transfer format is set to data length of eight bits, with odd parity, and stop bit length of one bit, and clock source for baud rate generator is set to φ.
(c) Selects transmission or reception and clock output in asynchronous mode, and enables or disables interrupt requests. In this sample task, communication mode is set to asynchronous mode, clock source is set to internal clock, and SCK3 pin is set to function as clock output pin. Interrupts requested by transmit data register empty and receive data register full are disabled.
(d) Indicates the operational status of SCI3 by status flags (transmit data register empty, receive data register full, overrun error, framing error, parity error, and transmit end).
(e) When TSR is found to be empty, sends the transmit data written in TDR to TSR.
(f) When one byte of data is received, sends receive data in RSR to RDR.
(g) Transmit data.
(h) Receive data.

Figure 2 Simultaneous Serial Transmission/Reception in Asynchronous Mode

- The serial status register (SSR) is an 8-bit register containing status flags that indicate the operational status of SCI3, and multiprocessor bits. SSR can be read from or written to by the CPU at any time. Bits TDRE, RDRF, OER, PER, and FER can only be cleared to 0. However, in order to clear these bits by writing 0, 1 must first be read. Bits TEND and MPBR are read-only bits, and cannot be modified.
- The bit rate register (BRR) is an 8-bit register that designates the transmit/receive bit rate in accordance with the baud rate generator operating clock selected by bits CKS1 and CKS0 of the serial mode register (SMR). BRR can be read from or written to by the CPU at any time.
- Table 1 shows examples of BRR settings in asynchronous mode. The values shown are for active (high-speed) mode and with an OSC of 16 MHz.

Table 1 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode)

Bit Rate (bit/s)	n	N	Error (%)
110	3	70	0.03
150	2	207	0.16
300	2	103	0.16
600	1	207	0.16
1200	1	103	0.16
2400	0	207	0.16
4800	0	103	0.16
9600	0	51	0.16
19200	0	25	0.16
31250	0	15	0.00
38400	0	12	0.16

Notes: 1. The BRR setting must be such that the error is within 1%.

2. The value set in BRR is given by the following equation:

$$N = \frac{OSC}{64 \times 2^{2n} \times B} \times 10^6 - 1$$

where

B: Bit rate (bit/s)

N: BRR setting ($0 \le N \le 255$)

OSC: Value of ϕOSC (MHz) = 16 MHz

n: Value set in bits CKS1 and CKS0 in SMR ($0 \le n \le 3$)
(The relation between n and the clock is shown in table 2.)

Table 2 Relation between n and Clock

n	Clock	SMR Setting	
		CKS1	CKS0
0	ϕ	0	0
1	ϕ/4	0	1
2	ϕ/16	1	0
3	ϕ/64	1	1

3. The bit rate error is given by the following equation (rounded off to two decimals):

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

4. When OSC is 16 MHz, the maximum bit rate (in asynchronous mode) is 500000 (bit/s). In this case, n = 0 and N = 0.

- In asynchronous mode, serial communication is performed with synchronization provided character by character. A start bit indicating the start of communication and one or two stop bits indicating the end of communication are added to each character before it is sent.
- SCI3 has separate transmission and reception units, allowing full-duplex communication. As the transmission and reception units are both double-buffered, data can be written during transmission and read during reception, making continuous transmission and reception possible.
- Figure 3 shows the general data transfer format in asynchronous communication. In asynchronous communication, the communication line is normally in the mark state (high level). SCI3 monitors the communication line and when it detects a space (low level), identifies this as a start bit and begins serial data communication.
- One transfer data character consists of a start bit (low level), followed by transmit/receive data (LSB-first format, starting from the least significant bit), a parity bit (high or low level), and finally one or two stop bits (high level).
- In asynchronous mode, synchronization is performed by the falling edge of the start bit during reception. The data is sampled on the 8th pulse of a clock with a frequency 16 times the bit period, so that the transfer data is latched at the center of each bit.

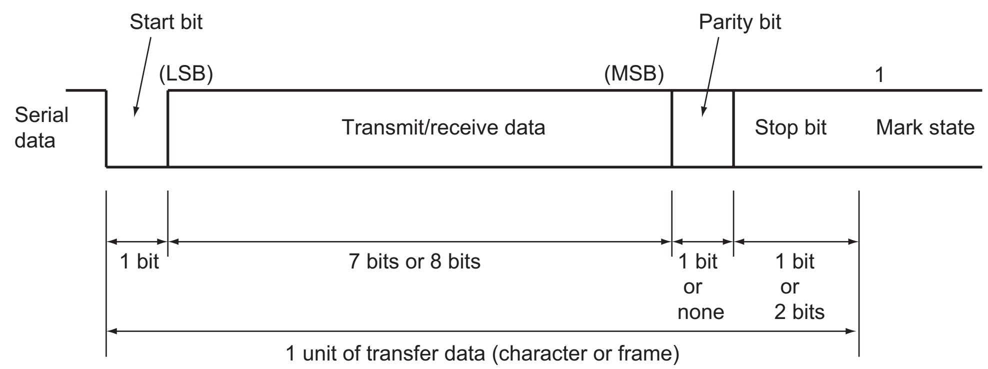


Figure 3 Data Format in Asynchronous Communication

- The SCI3 clock (SCK3) pin is the SCI3 clock I/O pin.
- The SCI3 receive data input (RXD) pin is the input pin for SCI3 receive data.
- The SCI3 transmit data output (TXD) pin is the output pin for SCI3 transmit data.
- SCI3 can generate six kinds of interrupts: transmit end, transmit data empty, receive data full, and three receive error interrupts (overrun error, framing error, and parity error). These interrupts have the same vector address.
- Each interrupt request can be enabled or disabled by means of bits TIE and RIE in SCR3.
- When bit TDRE is set to 1 in SSR, a TXI interrupt is requested. When bit TEND is set to 1 in SSR, a TEI interrupt is requested. These two interrupts are generated during transmission.
- The initial value of bit TDRE in SSR is 1. Therefore, if the transmit data empty interrupt request (TXI) is enabled by setting bit TIE to 1 in SCR3 before transmit data is transferred to TDR, a TXI interrupt will be requested even if the transmit data is not ready.
- The initial value of bit TEND in SSR is 1. Therefore, if the transmit end interrupt request (TEI) is enabled by setting bit TEIE to 1 in SCR3 before transmit data is transferred to TDR, a TEI interrupt will be requested even if the transmit data has not been sent.
- Effective use of these interrupt requests can be made by having processing that transfers transmit data to TDR carried out in the interrupt service routine. To prevent the generation of these interrupt requests (TXI and TEI), on the other hand, the enable bits for these interrupt requests (bits TIE and TEIE) should be set to 1 after transmit data has been transferred to TDR.
- When bit RDRF is set to 1 in SSR, an RXI interrupt is requested, and if any of bits OER, PER, and FER is set to 1, an ERI interrupt is requested. These two interrupt requests are generated during reception.

2. Table 3 lists the function allocation for this sample task. The functions listed in table 3 are allocated for simultaneous serial transmission/reception in asynchronous mode.

Table 3 Function Allocation

Function	Function Assignment
RSR	Receives serial data
RDR	Stores receive data
SMR	Sets the serial data transfer format and clock source for the baud rate generator
SSR	Status flags indicating the operational status of SCI3
BRR	Sets bit rate of transmission/reception
PMR1	Sets TXD output pin
SCK3	SCI3 clock output pin
TXD	SCI3 transmit data output pin
RXD	SCI3 receive data input pin

3. Operational Description

Figure 4 shows this sample task's principle of operation. Simultaneous serial transmission/reception in asynchronous mode is performed due to hardware and software processing shown in figure 4.

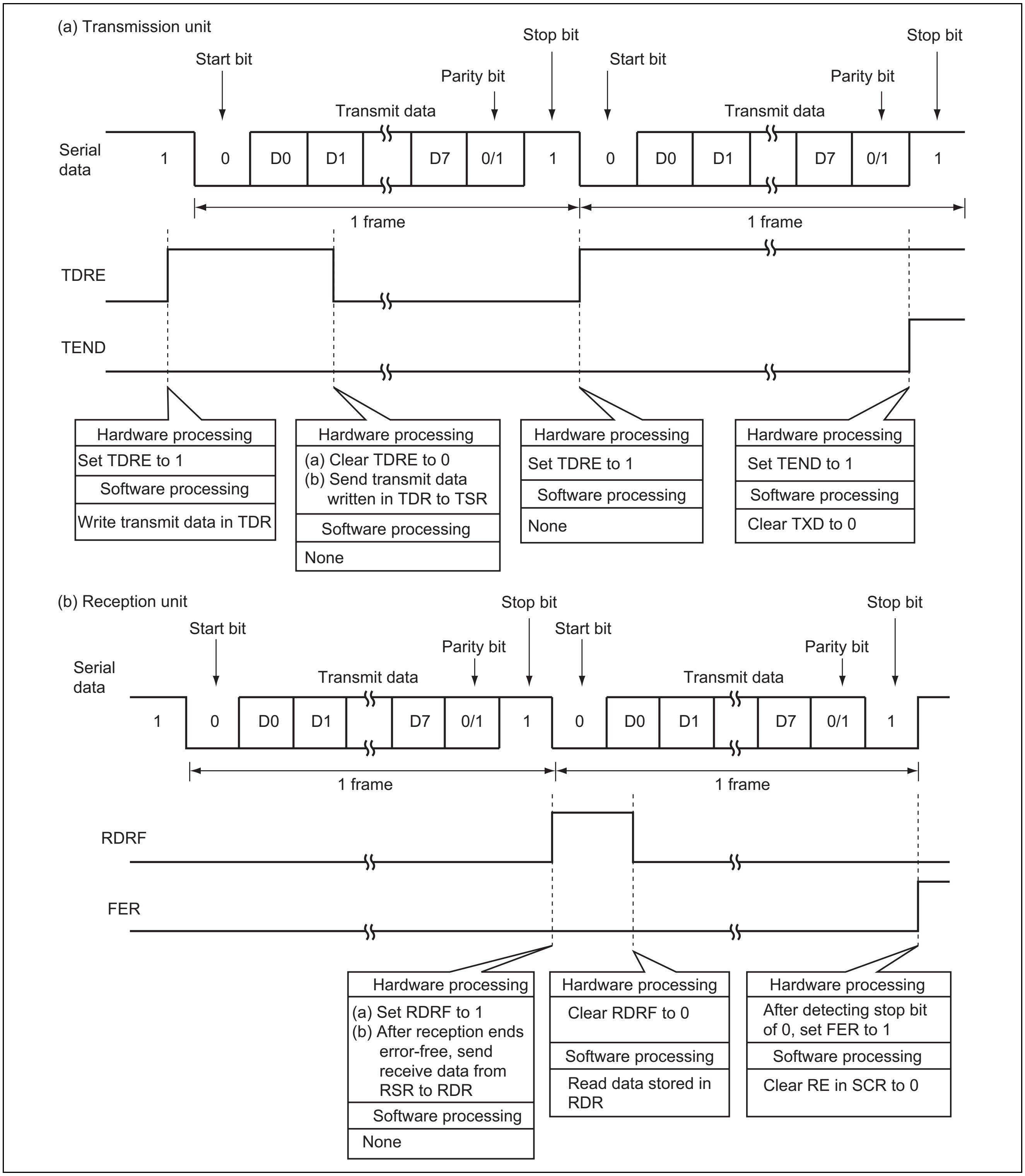


Figure 4 Operation Principle: Simultaneous Serial Transmission/Reception in Asynchronous Mode

4. Description of Software

4.1 Description of Modules

Table 4 describes the software used in this sample task.

Table 4 Description of Modules

Module Name	Label Name	Function
Main routine	main	Selects serial data transmission in asynchronous mode, branches to the receive error handling subroutine when a receive error occurs, and stops SCI3 after four bytes of data have been transmitted/received.
Receive error handling	er_sub	Judges which error has occurred, OER, FER, or PER, and carries out the corresponding error handling.

4.2 Description of Arguments

Table 5 describes the argument used in this sample task.

Table 5 Description of Arguments

Argument Name	Function	Used in	Data Length	I/O
STD0 to STD3	Serial transmit data in asynchronous mode	Main routine	1 byte	Output
SRD0 to SRD3	Serial receive data in asynchronous mode	Main routine	1 byte	Input

4.3 Description of Internal Registers

Table 6 describes the internal registers used in this sample task.

Table 6 Description of Internal Registers

Register Name		Functional Description	Address	Setting
SMR	COM	Serial mode register (communication mode): When COM is cleared to 0, the communication mode is set to asynchronous mode.	H'FFA8 Bit 7	0
	CHR	Serial mode register (character length): When CHR is cleared to 0, the data length in asynchronous mode is set to 8 bits.	H'FFA8 Bit 6	0
	PE	Serial mode register (parity enable): When PE is set to 1, parity bit addition and checking are enabled at transmission in asynchronous mode.	H'FFA8 Bit 5	1
	PM	Serial mode register (parity mode): When PM is set to 1, odd parity is to be used for parity addition and checking.	H'FFA8 Bit 4	1
	STOP	Serial mode register (stop bit length): When STOP is cleared to 0, the stop bit length in asynchronous mode is set to 1 bit.	H'FFA8 Bit 3	0
	MP	Serial mode register (multiprocessor mode): When MP is cleared to 0, the multiprocessor communication function is disabled.	H'FFA8 Bit 2	0
	CKS1 CKS0	Serial mode register (clock select 1 and 0): When CKS1 and CKS0 are both cleared to 0, the clock source for the baud rate generator is set to the system clock.	H'FFA8 Bit 1 Bit 0	CKS1 = 0 CKS0 = 0

Table 6 Description of Internal Registers

Register Name		Functional Description	Address	Setting
BRR		Bit rate register: When BRR is set to H'0F, the transmit bit rate that is in accordance with the baud rate generator operating clock selected by bits CKS1 and CKS0 in SMR is set to 31250 (bit/s).	H'FFA9	H'0F
SCR3	TE	Serial control register 3 (transmit enable): When TE is cleared to 0, transmit operation is disabled (TXD pin is the transmit data pin).	H'FFAA Bit 5	0
	RE	Serial control register 3 (receive enable): When RE is cleared to 0, receive operation is disabled. When RE is set to 1, receive operation is enabled.	H'FFAA Bit 4	0
	CKE1 CKE0	Serial control register 3 (clock enable): When CKE1 is cleared to 0 and CKE0 is set to 1, the clock source is set to an internal clock and the SCK3 pin functions as a clock output pin in asynchronous mode.	H'FFAA Bit 1 Bit 0	CKE1 = 0 CKE0 = 1
TDR		Transmit data register: 8-bit register that stores the transmit data.	H'FFAB	—
RDR		Receive data register: 8-bit register that stores the receive data.	H'FFAD	—
SSR	TDRE	Serial status register (transmit data register empty): When TDRE is cleared to 0, the transmit data written in TDR has not been sent to TSR. When TDRE is set to 1, the transmit data has not been written to TDR, or the transmit data written in TDR has been sent to TSR.	H'FFAC Bit 7	1
	RDRF	Serial status register (receive data register full): When RDRF is cleared to 0, no receive data is stored in RDR. When RDRF is set to 1, receive data is stored in RDR.	H'FFAC Bit 6	1
	OER	Serial status register (overrun error): When OER is cleared to 0, reception is in progress or completed. When OER is set to 1, an overrun error has occurred during reception.	H'FFAC Bit 5	0
	FER	Serial status register (framing error): When FER is cleared to 0, reception is in progress or completed. When FER is set to 1, a framing error has occurred during reception.	H'FFAC Bit 4	0
	PER	Serial status register (parity error): When PER is cleared to 0, reception is in progress or completed. When PER is set to 1, a parity error has occurred during reception.	H'FFAC Bit 3	0
	TEND	Serial status register (transmit end): When TEND is cleared to 0, transmission is in progress. When TEND is set to 1, transmission has completed.	H'FFAC Bit 2	1
PMR1	PMR11	Port mode register 1 (P22/TXD pin function switch): When PMR11 is set to 1, the P22/TXD pin functions as the TXD output pin.	H'FFE0 Bit 1	1

4.4 Description of RAM

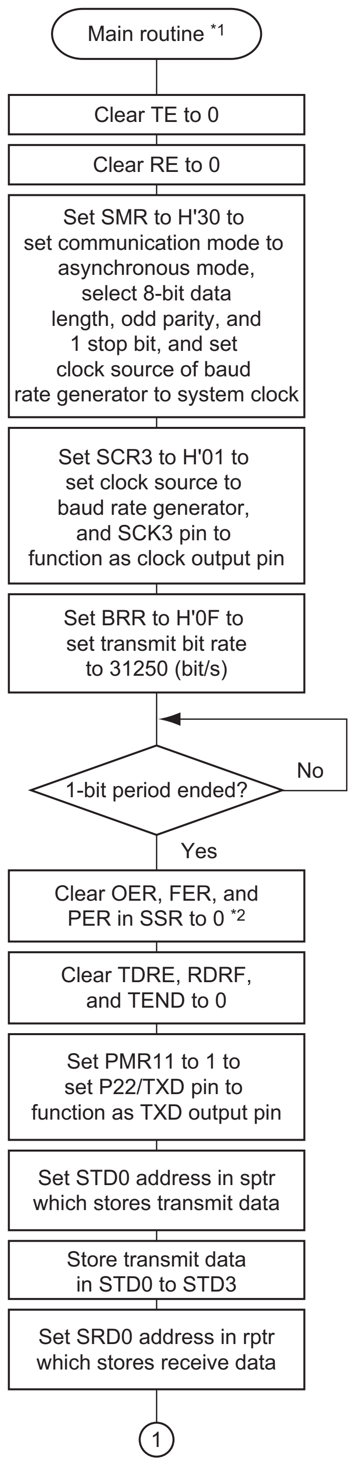
Table 7 describes the RAM used in this sample task.

Table 7 Description of RAM

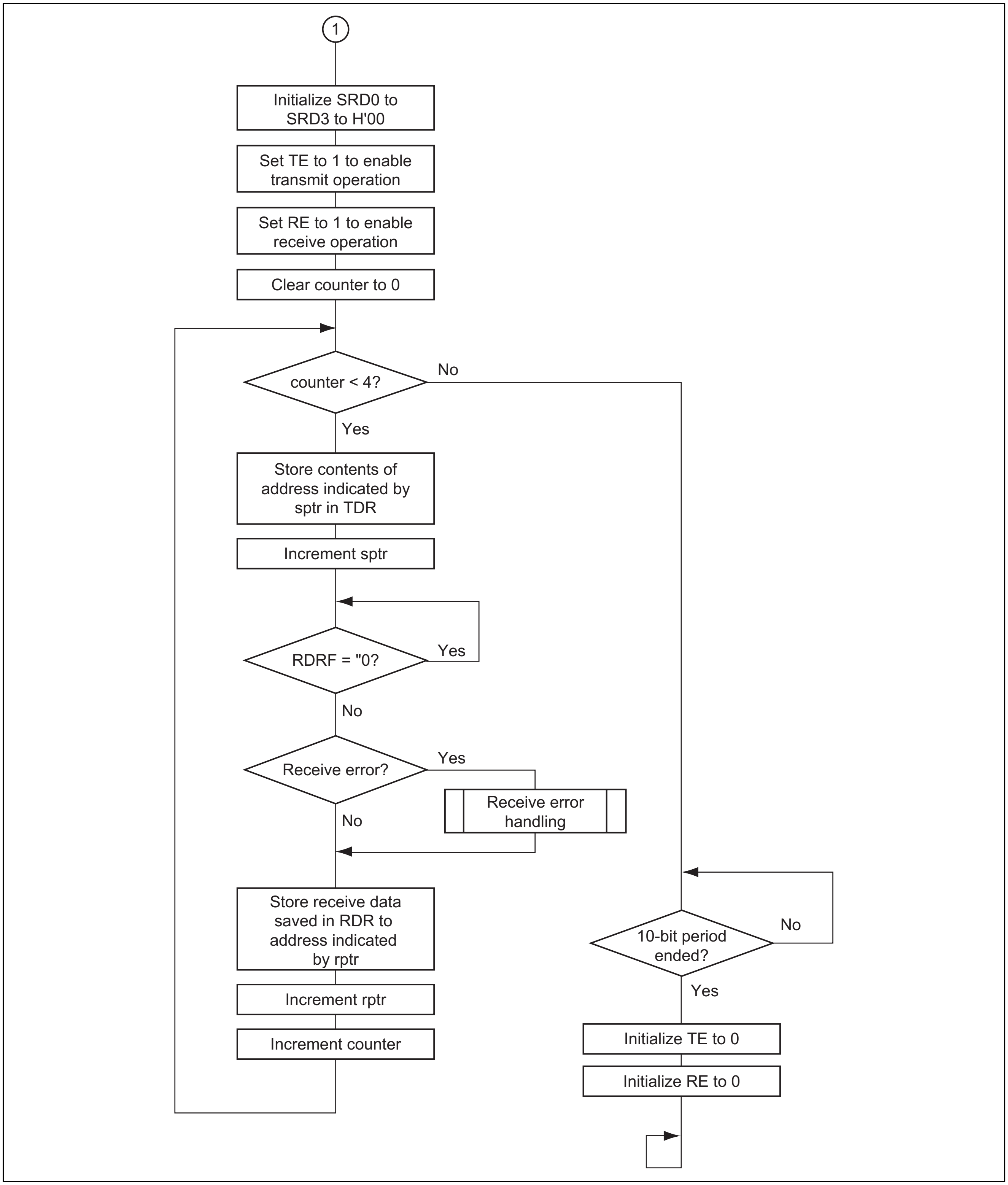
Label Name	Function	Address	Used in
STD0	Stores the first byte of transmit data in serial data transmission in asynchronous mode	H'FB80	Main routine
STD1	Stores the second byte of transmit data in serial data transmission in asynchronous mode	H'FB81	Main routine
STD2	Stores the third byte of transmit data in serial data transmission in asynchronous mode	H'FB82	Main routine
STD3	Stores the fourth byte of transmit data in serial data transmission in asynchronous mode	H'FB83	Main routine
SRD0	Receives the first byte of receive data in serial data reception in asynchronous mode	H'FB84	Main routine
SRD1	Receives the second byte of receive data in serial data reception in asynchronous mode	H'FB85	Main routine
SRD2	Receives the third byte of receive data in serial data reception in asynchronous mode	H'FB86	Main routine
SRD3	Receives the fourth byte of receive data in serial data reception in asynchronous mode	H'FB87	Main routine
counter	8-bit counter for counting four simultaneous serial transmit and receive operations in asynchronous mode	H'FB88	Main routine

5. Flowchart

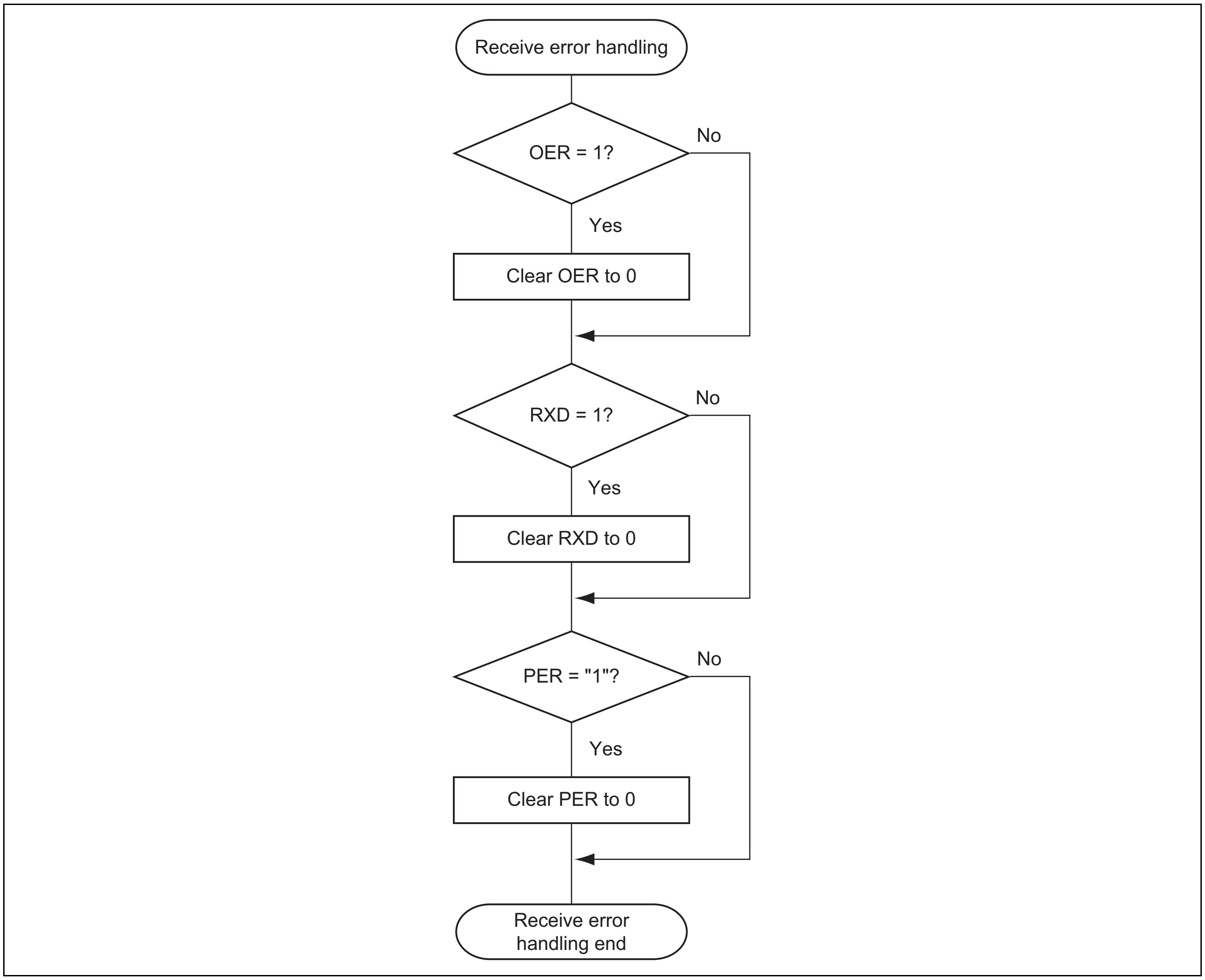
1. Main Routine



Notes: 1. In this sample task, the stack pointer is set in INIT.SRC (assembly).
2. In asynchronous mode (simultaneous transmission and reception), OER, FER, and PER must be cleared to 0.


1
Initialize SRD0 to SRD3 to H'00
Set TE to 1 to enable transmit operation
Set RE to 1 to enable receive operation
Clear counter to 0
counter < 4?
No
Yes
Store contents of address indicated by sptr in TDR
Increment sptr
RDRF = "0?
Yes
No
Receive error?
Yes
Receive error handling
No
Store receive data saved in RDR to address indicated by rptr
Increment rptr
Increment counter
10-bit period ended?
No
Yes
Initialize TE to 0
Initialize RE to 0

2. Sub Routine



5.1 Link Address Designation

Section Name	Address
CV1	H'0000
P	H'0100
B	H'FB80

6. Program Listing

INIT.SRC (Program listing)

```
      .EXPORT  _INIT
      .IMPORT  _main
 ;
      .SECTION    P,CODE
 _INIT:
      MOV.W    #H'FF80,R7
      LDC.B    #B'10000000,CCR
      JMP      @_main
 ;
      .END
```

```
/**********************************************************/
/*                                                        */
/*  H8/300H Tiny Series -H8/3664-                         */
/*  Application Note                                      */
/*                                                        */
/*  'Asynchronous Serial Data Simultaneous                */
/*   Transmission and Reception'                          */
/*                                                        */
/*  Function                                              */
/*  : Serial Communication Interface                      */
/*     Asynchronous Serial Interface                      */
/*     -Transmitting/Receiving                            */
/*                                                        */
/*  External Clock : 16MHz                                */
/*  Internal Clock : 16MHz                                */
/*  Sub Clock      : 32.768kHz                            */
/*                                                        */
/**********************************************************/

#include    <machine.h>
```

```
/*********************************************************/
/*  Symbol Defnition                                     */
/*********************************************************/
struct BIT {
    unsigned char    b7:1;      /* bit7 */
    unsigned char    b6:1;      /* bit6 */
    unsigned char    b5:1;      /* bit5 */
    unsigned char    b4:1;      /* bit4 */
    unsigned char    b3:1;      /* bit3 */
    unsigned char    b2:1;      /* bit2 */
    unsigned char    b1:1;      /* bit1 */
    unsigned char    b0:1;      /* bit0 */
};
#define    SMR      *(volatile unsigned char *)0xFFA8      /* Serial Mode Register            */
#define    SMR_BIT (*(struct BIT *)0xFFA8)                 /* Serial Mode Register            */
#define    COM      SMR_BIT.b7                             /* Communication Mode              */
#define    CHR      SMR_BIT.b6                             /* Character Length                */
#define    PE       SMR_BIT.b5                             /* Parity Enable                   */
#define    PM       SMR_BIT.b4                             /* Parity Mode                     */
#define    STOP     SMR_BIT.b3                             /* Stop Bit Length                 */
#define    MP       SMR_BIT.b2                             /* Multiprocesor Mode              */
#define    CKS1     SMR_BIT.b1                             /* Clock Select 1                  */
#define    CKS0     SMR_BIT.b0                             /* Clock Select 0                  */
#define    BRR      *(volatile unsigned char *)0xFFA9      /* Bit Rate Register               */
#define    SCR3     *(volatile unsigned char *)0xFFAA      /* Serial Control Register 3       */
#define    SCR3_BIT (*(struct BIT *)0xFFAA)                /* Serial Control Register 3       */
#define    TIE      SCR3_BIT.b7                            /* Transmit Interrupt Enable       */
#define    RIE      SCR3_BIT.b6                            /* Receive Interrupt Enable        */
#define    TE       SCR3_BIT.b5                            /* Transmit Enable                 */
#define    RE       SCR3_BIT.b4                            /* Receive Enable                  */
#define    MPIE     SCR3_BIT.b3                            /* Multiprocessor Interrupt Enable */
#define    TEIE     SCR3_BIT.b2                            /* Transmit End Interrupt Enable   */
#define    CKE1     SCR3_BIT.b1                            /* Clock Enable 1                  */
#define    CKE0     SCR3_BIT.b0                            /* Clock Enable 0                  */
#define    TDR      *(volatile unsigned char *)0xFFAB      /* Transmit Data Register          */
```

```
#define    SSR      *(volatile unsigned char *)0xFFAC      /* Serial Status Register          */
#define    SSR_BIT (*(struct BIT *)0xFFAC)                 /* Serial Status Register          */
#define    TDRE     SSR_BIT.b7                             /* Transmit Data Register Empty    */
#define    RDRF     SSR_BIT.b6                             /* Receive Data Register Full      */
#define    OER      SSR_BIT.b5                             /* Overrun Erorr                   */
#define    FER      SSR_BIT.b4                             /* Framing Erorr                   */
#define    PER      SSR_BIT.b3                             /* Parity Erorr                    */
#define    TEND     SSR_BIT.b2                             /* Transmit End                    */
#define    MPBR     SSR_BIT.b1                             /* Multiprocessor Bit Receive      */
#define    MPBT     SSR_BIT.b0                             /* Multiprocessor Bit Transfer     */
#define    PMR1_BIT (*(struct BIT *)0xFFE0)                /* Port Mode Register 1            */
#define    PMR11    PMR1_BIT.b1                            /* Port Mode Register 1 bit1       */
#define    RDR      *(volatile unsigned char *)0xFFAD      /* Receive data Register           */

/**********************************************************/
/*     Function Definition                               */
/**********************************************************/
extern  void    INIT( void );                              /* SP Set                          */
void    main    ( void );
void    er_sub  ( void );

/**********************************************************/
/*   RAM Allocation                                      */
/**********************************************************/
    unsigned char   STD[4];
    unsigned char   SRD[4];
    unsigned char   counter;

/**********************************************************/
/*  Vector Address                                       */
/**********************************************************/
#pragma section     V1                                     /* VECTOR SECTOIN SET              */
void (*const VEC_TBL1[])(void) = {
/* 0x00 - 0x0f */
    INIT                                                   /* 00 Reset                        */
};

#pragma section                                             /* P                              */
```

```
/**********************************************************/
/*   Main Program                                         */
/**********************************************************/
void main ( void )
{
    unsigned char   stus;
    unsigned char   *sptr,*rptr;

    TE = 0;                                            /* Clear Serial Transmitting                          */

    RE = 0;                                            /* Clear Serial Receiving                             */

    SMR = 0x30;                                        /* Initialize Serial Mode Register                    */

    SCR3 = 0x01;                                       /* Initialize Serial Control Register 3               */

    BRR = 0x0F;                                        /* Initialize Bit Rate  Register                      */

    for(counter = 0 ; counter < 1 ; counter++){        /* dummy wait                                         */
                     ;
    }

    OER = 0;                                          /* Clear OER                                           */
    FER = 0;                                          /* Clear FER                                           */
    PER = 0;                                          /* Clear PER                                           */

    TDRE = 0;                                         /* Clear TDRE                                          */
    RDRF = 0;                                         /* Clear RDRF                                          */
    TEND = 0;                                         /* Clear TEND                                          */

    PMR11 = 1;                                        /* Initialize Output Port TXD                          */

    sptr = &STD[0];                                   /* Initialize Serial Transmitting Data Addres          */

    STD[0] = 0x00;                                    /* Set Serial Transfer Data 0                          */
    STD[1] = 0x55;                                    /* Set Serial Transfer Data 1                          */
    STD[2] = 0xAA;                                    /* Set Serial Transfer Data 2                          */
    STD[3] = 0xFF;                                    /* Set Serial Transfer Data 3                          */

    rptr = &SRD[0];                                   /* Initialize Serial Receiving Data Address            */

    SRD[0] = 0x00;                                     /* Initialize Serial Receiving Data 0                 */
    SRD[1] = 0x00;                                    /* Initialize Serial Receiving Data 1                  */
    SRD[2] = 0x00;                                    /* Initialize Serial Receiving Data 2                  */
    SRD[3] = 0x00;                                    /* Initialize Serial Receiving Data 3                  */

    TE = 1;                                           /* Start Serial Transmitting                           */
    RE = 1;                                           /* Start Serial Receiving                              */

    for(counter = 0 ; counter < 4 ; counter++){       /* Serial Transmitting/Receiving Data Counter 4 Loop*/

        TDR = *sptr;                                  /* Save Serial Transmitting Data                       */
        sptr++;                                       /* Increment Serial Transmittimg Data Address          */
```

```
        while(RDRF == 0){                                /* End Serial Receive End ?                    */
            ;
        }

        if ((SSR & 0x38) != 0){                         /* Error Flag = 1 ?                            */
            er_sub();
        }
         else{
            *rptr = RDR;                                /* Save Serial Receiving Data                  */
         }

        rptr++;                                         /* Increment Serial Receiving Data Address     */

    }

    for(counter = 0 ; counter < 10 ; counter++){        /* dummy Loop                                  */
                    ;
    }

    TE = 0;                                             /* Initialize Transmitting Enable              */
    RE = 0;                                             /* Initialize Receiving Enable                 */

    while(1){
       ;
    }
}

void er_sub(void){

    if (OER != 0)   {
        OER = 0;                                         /* Clear OER                                  */
    }
    if (FER != 0)   {
        FER = 0;                                         /* Clear FER                                  */
    }
    if (PER != 0)   {
        PER = 0;                                         /* Clear PER                                  */
    }
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb.26.03	—	First edition issued
2.00	Jul.22.05	—	Second edition issued