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

H8/300L SLP Series

Simultaneous Transmission/Reception in Synchronous Mode

Introduction

Four bytes of 8-bit data is transmitted/received simultaneously using the serial transfer function in synchronous mode. Data is transmitted/received in the LSB first format, starting from the least significant bit of data.

Target Device

H8/38024

Contents

1. Specifications

1. As shown in figure 1.1, four bytes of 8-bit data is transmitted/received simultaneously using the serial transfer function in synchronous mode.
2. Data is transmitted and received simultaneously at a transfer clock period of 4 µs using an internal clock as a transfer clock.
3. The data length of transmitted/received data is eight bits and data is transmitted/received in the LSB first format, starting from the least significant bit of data.

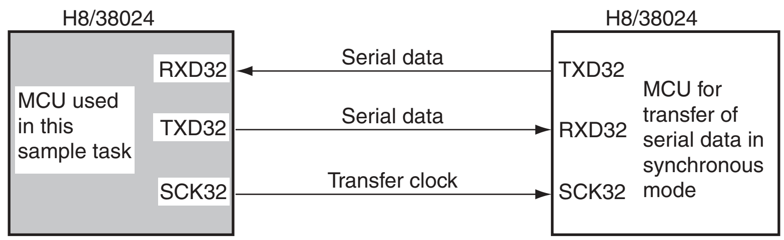
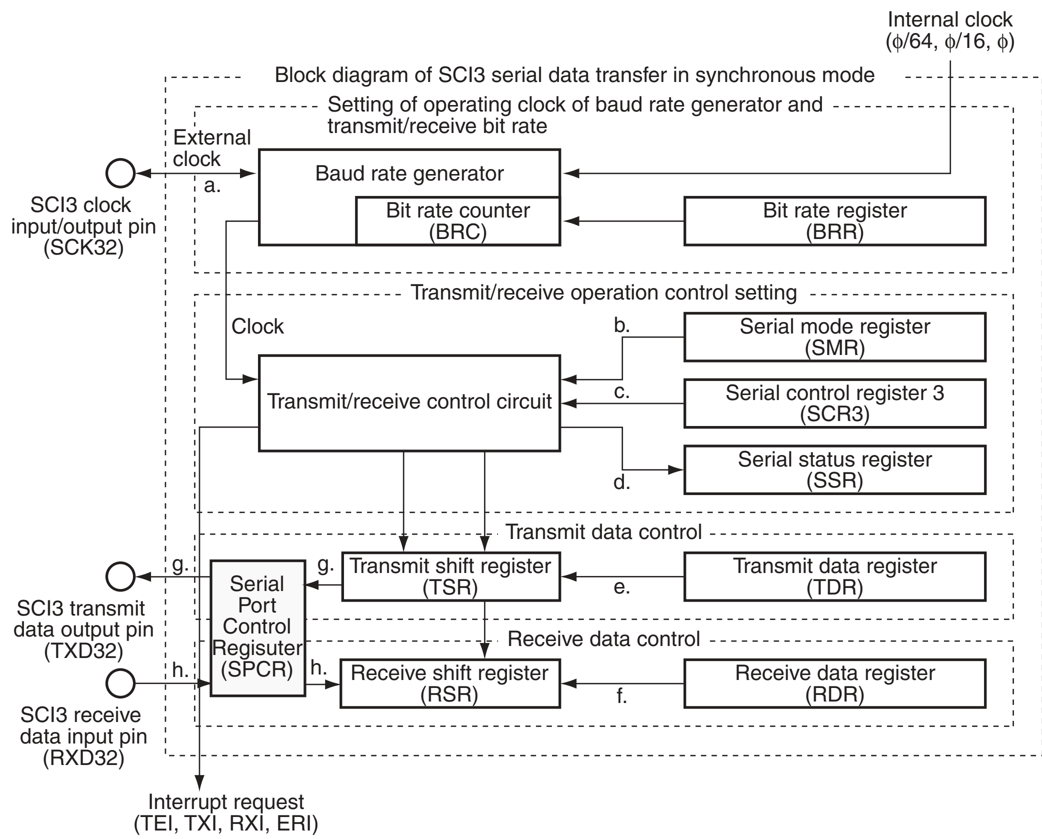


Figure 1.1 Simultaneous Serial Data Transmission/Reception in Synchronous Mode

2. Description of Functions

1. In this sample task, serial data is simultaneously transmitted/received in synchronous mode using the Serial Communication Interface (SCI). Figure 2.1 shows a block diagram of simultaneous serial data transmission/reception in synchronous mode which is described below.
   - The system clock (ϕ) is a 10-MHz OSC clock that is used as a reference clock for operating the CPU and peripheral functions.
   - Only overrun errors are detected as receive errors.
   - In synchronous mode, the data length is eight bits.
   - The receive shift register (RSR) is a register used to receive serial data. Serial data input to RSR from the RXD32 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 TXD32 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 internal baud rate generator.
   - Serial control register 3 (SCR3) is an 8-bit register for selecting transmit/receive operation and the transmit/receive clock source.
   - The serial status register (SSR) contains status flags that indicate the operational status of SCI3, and transmit/receive multiprocessor bits. Bits TDRE, RDRF, OER, PER, and FER can only be cleared to 0.
   - The transfer clock can be selected from a total of four clocks: three internal clocks and an external clock. When an internal clock is selected, the SCK32 pin functions as an output pin. When clock consecutive output mode is selected, the selected clock is consecutively output from the SCK32 pin. When an external clock is selected, the SCK32 pin functions as an input pin.

- In this sample task, the source of the transfer clock is system clock for the internal baud rate generator and the transfer clock cycle is 4 µs.
- As the SCI3 data transfer format, an 8-bit data can be selected, and data is transmitted in the LSB-first format, starting from the least significant bit. Transmit data is output from one falling edge to the next rising edge of the transfer clock. Receive data is latched at the rising edge of the transfer clock.
- In this sample task, the operation mode is set to an 8-bit mode, and 8-bit data is received.
- The SCI3 clock (SCK32) pin is the SCI3 clock I/O pin.
- The SCI3 receive data input (RXD32) pin is the input pin for SCI3 receive data.
- The SCI3 transmit data output (TXD32) pin is the output pin for SCI3 transmit data.
- The serial port control register (SPCR) is an 8-bit register to control RXD32 pin and P42/TXD32 pin. In this sample task, P42/TXD32 pin is set to TXD32 pin, and input/output data of RXD32 and TXD32 is set not to be inverted.



Notes: a. The operating clock (φ) for baud rate generator selected by SMR is output.
b. The serial data transfer format is set and the clock source for baud rate generator is selected.
c. The transmit/receive operation and clock output pin in synchronous mode are selected.
d. The status flags (transmit data register empty, receive data register full, overrun error) indicate the operation status of SCI3.
e. Detecting that TSR is "empty", transmit data written in TDR is transferred to TSR.
f. The received data is transferred from RSR to RDR when receiving of one-byte data is completed.
g. Transmit data.
h. Receive data.

Figure 2.1 Block Diagram of Simultaneous Serial Data Transmission/Reception in Synchronous Mode

2. Table 2.1 shows assignment of functions in this sample task. Serial data is simultaneously transmitted and received in synchronous mode by assigning the functions as shown in table 2.1.

Table 2.1 Assignment of Functions

Function	Assignment
TSR	A register to transmit serial data
TDR	A register to store transmit data
RSR	A register to receive serial data
RDR	A register to store receive data
SMR	Sets the serial data transfer format and clock source for the baud rate generator
SSR	Status flags to indicate operation status of SCI3
BRR	Sets transmit/receive bit rate
SCR3	Enables transmit/receive operation, sets TXD32 output pin, sets RXD32 input pin, and sets SCK32 pin function as clock output pin
SCK32	SCI3 clock output pin
TXD32	SC13 transmit data output pin
RXD32	SC13 receive data input pin
SPCR	Sets TXD32 output pin

3. Principle of Operation

1. Figure 3.1 illustrates the principle of operation of this sample task. Serial data is simultaneously transmitted and received by hardware and software processing as shown in figure 3.1.

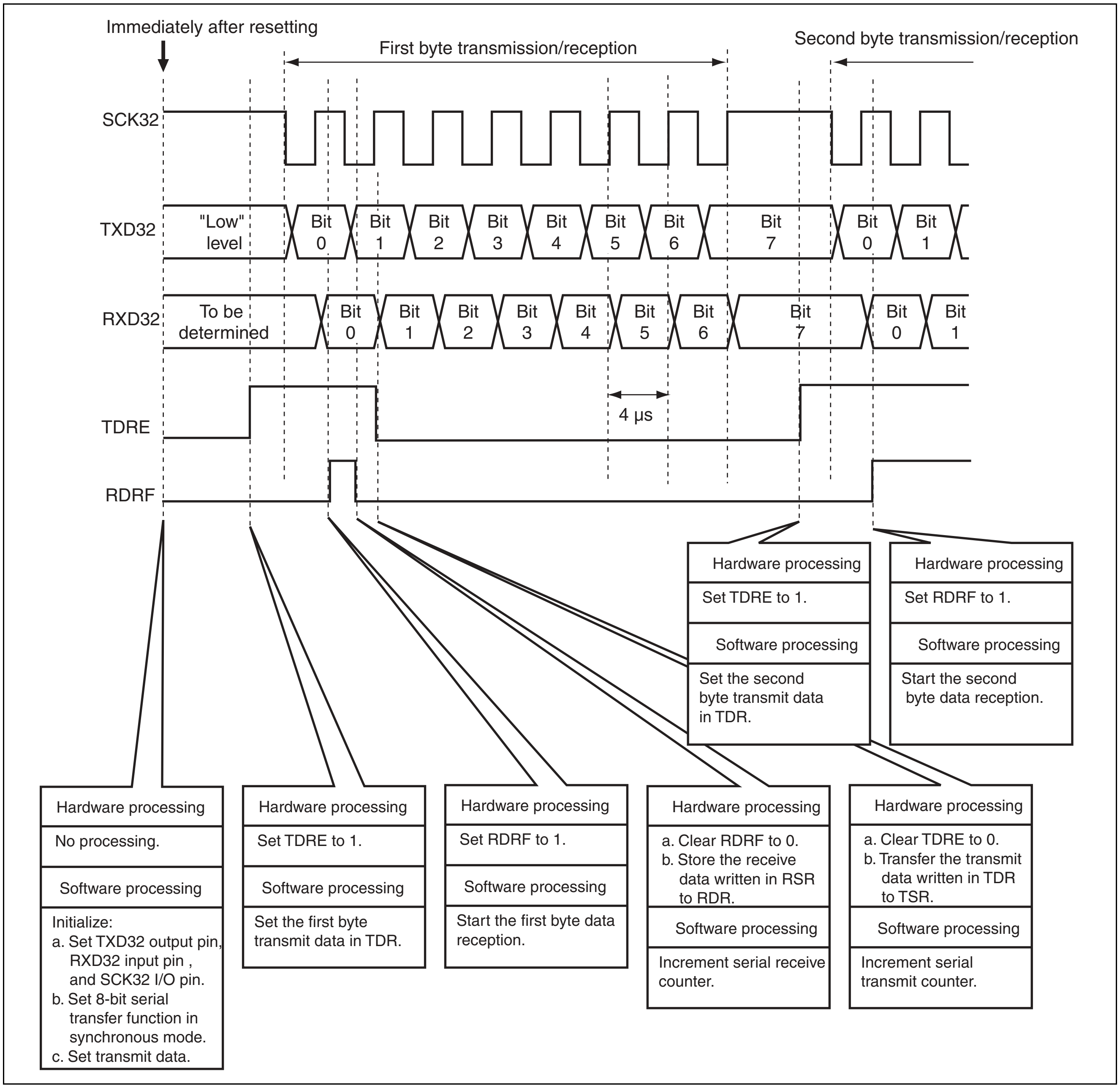


Figure 3.1 Operation Principle of Simultaneous Serial Data Transmission/Reception in Synchronous Mode

4. Description of Software

4.1 Modules

Table 4.1 describes the module in this sample task.

Table 4.1 Description of Modules

Module	Label	Function
Main Routine	main	Transfer data setting, serial data transmission and reception in synchronous mode setting, receive data storage in the RAM and ending operation when 4 byte data is transmitted and received.

4.2 Arguments

Table 4.2 describes the arguments used in this sample task.

Table 4.2 Description of Arguments

Arguments	Function	Used in	Data Length	Input/ Output
STD[0] to STD[3]	Serial transmit data in synchronous mode	Main Routine	1 byte	Input
SRD[0] to SRD[3]	Serial receive data in synchronous mode	Main Routine	1 byte	Output

4.3 Internal Registers

Table 4.3 describes the internal registers in this sample task.

Table 4.3 Description of Internal Registers

Register		Function	Address	Setting
SPCR	SPC32	Serial Port Control Register (P42/TXD32 pin function switch) If SPC32 = 0, P42/TXD32 pin is set to P42 pin. If SPC32 = 1, P42/TXD32 pin is set to TXD32 pin.	H'FF91 Bit 5	1
	SCINV3	Serial Port Control Register (TXD32 Pin Output Data Inversion Switch) If SCINV3 = 0, TXD32 output data is not inverted. If SCINV3 = 1, TXD32 output data is inverted.	H'FF91 Bit 3	0
	SCINV2	Serial Port Control Register (RXD32 Pin Input Data Inversion Switch) If SCINV2 = 0, RXD32 input data is not inverted. If SCINV2 = 1, RXD32 input data is inverted.	H'FF91 Bit 2	0
SMR	COM	Serial Mode Register (Communication Mode) If COM = 0, the communication mode is set to asynchronous mode. If COM = 1, the communication mode is set to synchronous mode.	H'FFA8 Bit 7	1
	CKS1 CKS0	Serial Mode Register (Clock Select 1, 0) If CKS1 = 0 and CKS0 = 0, the clock source for the internal baud rate generator is set to ϕ clock.	H'FFA8 Bit 1 Bit 0	CKS1 = 0 CKS0 = 0
	MP	Serial Mode Register (Multi-Processor Mode) In synchronous mode, MP is set to 0.	H'FFA8 Bit 2	0

Register		Function	Address	Setting
BRR		Bit Rate Register If BRR = H'04, the transmit bit rate matched to the external operating clock is set to 250kbps.	H'FFA9	H'04
SCR3	TE	Serial Control Register 3 (Transmit Enable) If TE = 0, transmit operation is disabled. If TE = 1, transmit operation is enabled.	H'FFAA Bit 5	0
	RE	Serial Control Register 3 (Receive Enable) If RE = 0, receive operation is disabled. If RE = 1, receive operation is enabled.	H'FFAA Bit 4	0
	CKE1 CKE0	Serial Control Register 3 (Clock Enable 1, 0) If CKE1 = 0 and CKE0 = 0, the clock source is set to an internal clock and SCK32 pin function to clock output pin	H'FFAA Bit 1 Bit 0	CKE1 = 0 CKE0 = 0
TDR		Transmit Data Register An 8-bit register to store transmit data	H'FFAB	—
SSR	TDRE	Serial Status Register (Transmit Data Register Empty) If TDRE = 0, transmit data written in TDR is not transferred to TSR. If TDRE = 1, transmit data is not written in TDR or transmit data written in TDR is transferred to TSR.	H'FFAC Bit 7	1
	RDRF	Serial Status Register (Receive Data Register Full) If RDRF = 0, receive data is not stored in RDR. If RDRF = 1, receive data is stored in RDR.	H'FFAC Bit 6	0
	OER	Serial Status Register (Overrun Error) If OER = 0, data reception is in progress or completed. If OER = 1, an overrun error has occurred during reception.	H'FFAC Bit 5	0
	TEND	Serial Status Register (Transmit End) If TEND = 0, data is being transmitted. If TEND = 1, data transmission has been completed.	H'FFAC Bit 2	—
RDR		Receive Data Register An 8-bit register to store receive data	H'FFAD	—

4.4 Description of RAM

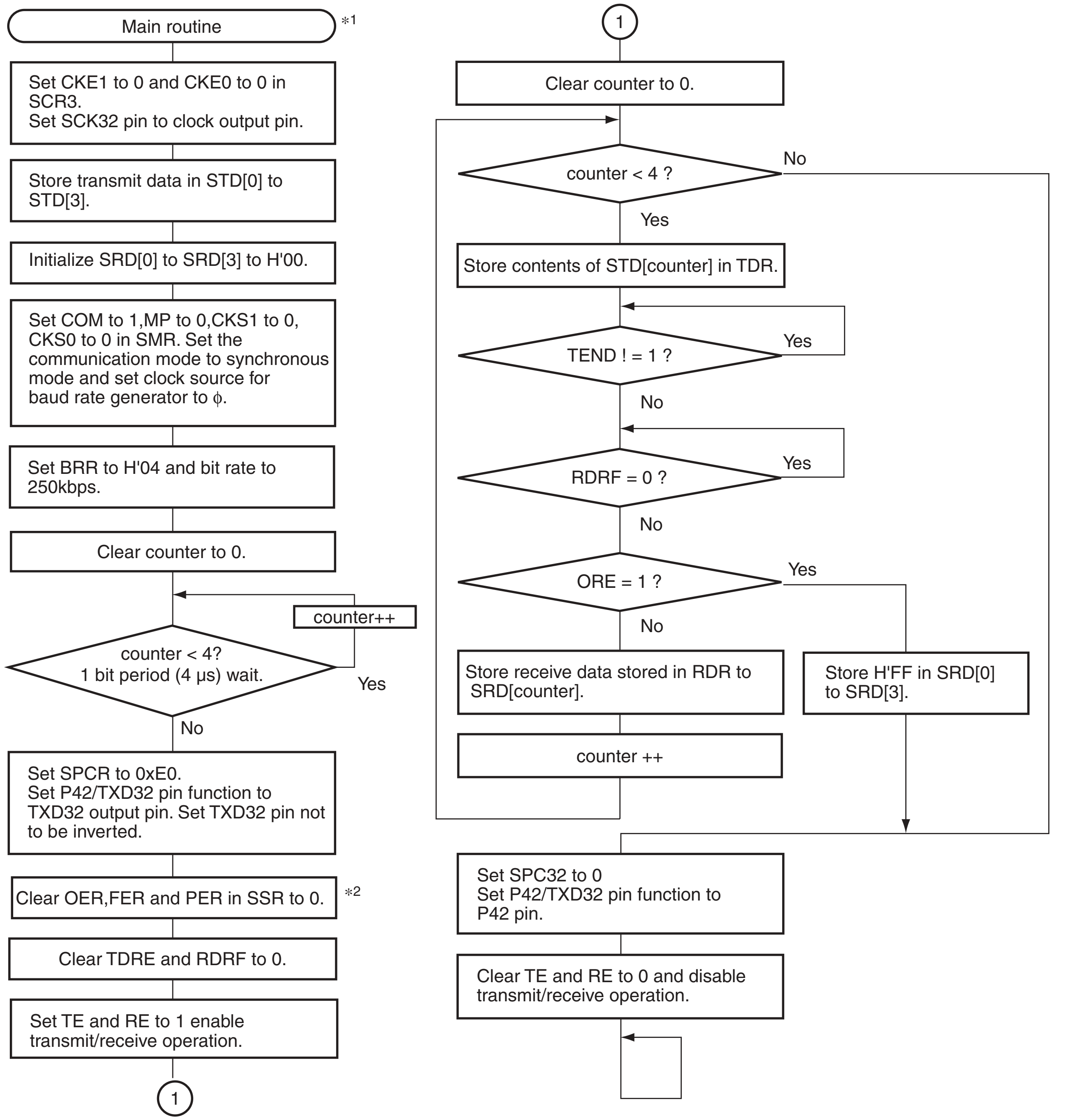
Table 4.4 describes the RAMs used in this sample task.

Table 4.4 Description of RAM

Label	Function	Address	Used in
STD[0]	Stores the first byte of serial transmit data in synchronous mode.	H'FB80	Main Routine
STD[1]	Stores the second byte of serial transmit data in synchronous mode.	H'FB81	Main Routine
STD[2]	Stores the third byte of serial transmit data in synchronous mode.	H'FB82	Main Routine
STD[3]	Stores the fourth byte of serial transmit data in synchronous mode.	H'FB83	Main Routine
SRD[0]	Stores the first byte of serial receive data in synchronous mode.	H'FB84	Main Routine
SRD[1]	Stores the second byte of serial receive data in synchronous mode.	H'FB85	Main Routine
SRD[2]	Stores the third byte of serial receive data in synchronous mode.	H'FB86	Main Routine
SRD[3]	Stores the fourth byte of serial receive data in synchronous mode.	H'FB87	Main Routine

5. Flowchart

1. Main routine



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

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
```

```
/*************************************************************************************************/
/*                                                                                               */
/*  H8/300L Super Low Power Series                                                               */
/*      -H8/38024 Series-                                                                        */
/*  Application Note                                                                             */
/*                                                                                               */
/*  'Synchronous Serial Data Simultaneous                                                        */
/*   Transmission/Reception'                                                                     */
/*                                                                                               */
/*  Function                                                                                     */
/*  : Serial Communication Interface                                                             */
/*     Synchronous Serial Interface                                                              */
/*     -Transmitting/Receiving                                                                   */
/*                                                                                               */
/*  External Clock : 10MHz                                                                       */
/*  Internal Clock : 5MHz                                                                        */
/*  Sub Clock      : 32.768kHz                                                                   */
/*                                                                                               */
/*************************************************************************************************/

#include    <machine.h>

/*************************************************************************************************/
/*  Symbol Definition                                                                            */
/*************************************************************************************************/
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_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                         /* Multiprocessor 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 Error                           */
#define     FER         SSR_BIT.b4                              /* Framing Error                           */
#define     PER         SSR_BIT.b3                              /* Parity Error                            */
#define     TEND        SSR_BIT.b2                              /* Transmit End                            */
#define     MPBR        SSR_BIT.b1                              /* Multiprocessor Bit Receive              */
#define     MPBT        SSR_BIT.b0                              /* Multiprocessor Bit Transfer             */
#define     SPCR        *(volatile unsigned char *)0xFF91       /* Transmit Data Register                  */
#define     SPCR_BIT    (*(struct BIT *)0xFF91)                 /* Port Mode Register 1                    */
#define     SPC32       SPCR_BIT.b5                             /* TXD Output Terminal                     */
#define     RDR         *(volatile unsigned char *)0xFFAD       /* Receive Data Register                   */

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

/********************************************************************************************************/
/*  RAM define                                                                                          */
/********************************************************************************************************/
unsigned char   STD[4];
unsigned char   SRD[4];

/********************************************************************************************************/
/*  Vector Address                                                                                      */
/********************************************************************************************************/
#pragma section     V1                                          /* Vector Section Set                      */
void (*const VEC_TBL1[])(void) = {                              /* 0x0000 - 0x000F                         */
    INIT                                                        /* 0x0000 Reset Vector                     */
};

#pragma section                                                 /* P                                       */
```

```
/*******************************************************************************************************/
/*  Main Program                                                                                       */
/*******************************************************************************************************/
void    main ( void )
{
    unsigned char   counter;

    CKE1 = 0;                                               /* Initialize Clock Enable 1 Output         */
    CKE0 = 0;                                               /* Initialize Clock Enable 0 Output         */

    STD[0] = 0x99;                                          /* 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               */

    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       */

    COM = 1;                                                /* Initialize Communication Mode            */
    MP = 0;                                                 /* Initialize Multiprocessor Mode           */
    CKS1 = 0;                                               /* Initialize Clock Select 1 phi            */
    CKS0 = 0;                                               /* Initialize Clock Select 0 phi            */

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

    SPCR = 0xE0;                                            /* Initialize Output Port TXD               */

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

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

    SCR3 = 0x30;                                            /* Start Serial Transmitting/Receiving      */

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

        TDR = STD[counter];                                 /* Save Serial Transmitting Data            */

        while(TEND != 1){                                   /* End Serial Transmitting                  */
           ;
        }

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

        if (OER == 1){                                      /* Overrun Error Flag = 1 ?                 */
            SRD[0] = 0xFF;                                  /* Overrun Error 0                          */
            SRD[1] = 0xFF;                                  /* Overrun Error 1                          */
            SRD[2] = 0xFF;                                  /* Overrun Error 2                          */
            SRD[3] = 0xFF;                                  /* Overrun Error 3                          */
            break;
        }
```

```
        else{
            SRD[counter] = RDR;                                  /* Save Serial Receiving Data               */
        }
    }

    SPC32 = 0;
    SCR3 = 0x00;                                                 /* Initialize Transmitting/Receiving Enable */

    while(1){
        ;
    }
}
```

Link address specifications

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.19.03	—	First edition issued

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

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.