H8/300H Tiny Series

Serial Data Reception in Asynchronous Mode

Introduction

Four bytes of 8-bit data are 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 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.

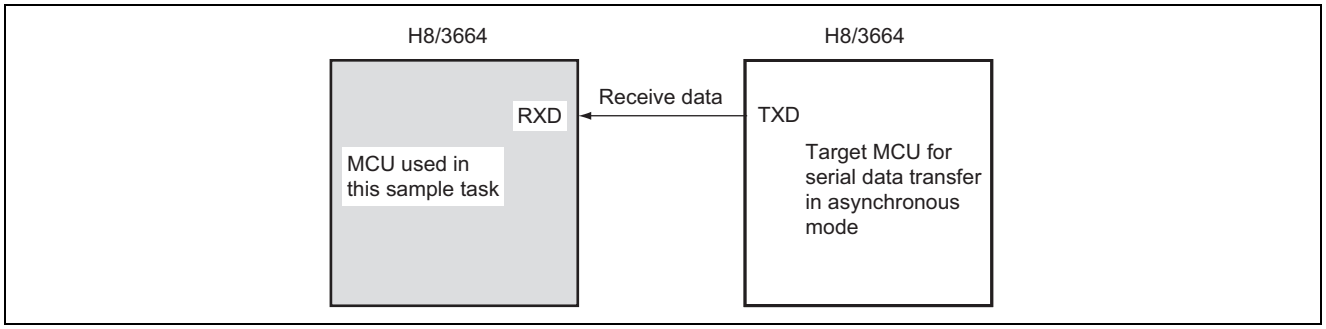
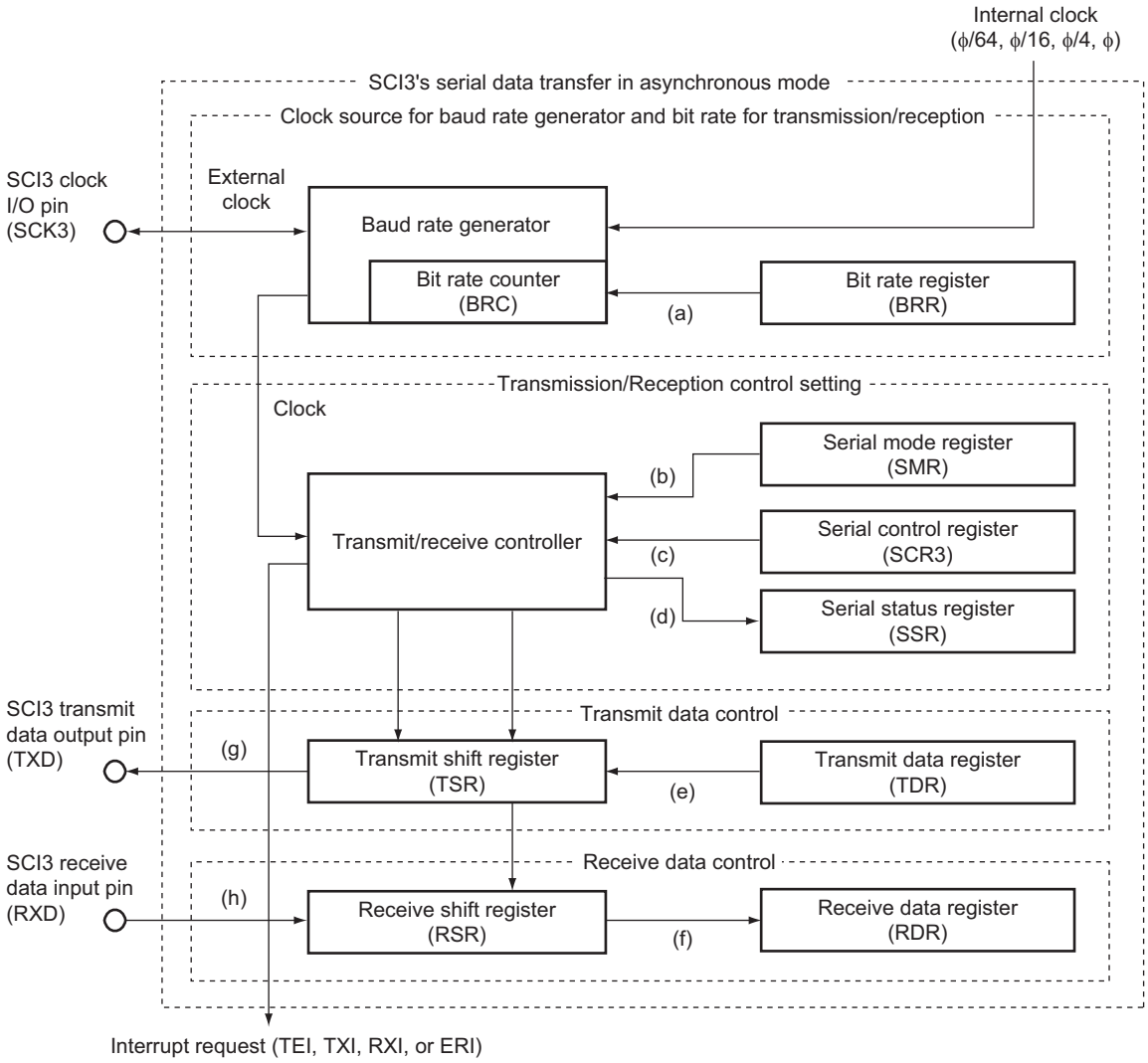


Figure 1 Serial Data Reception in Asynchronous Mode

2. Description of Functions Used

1. In this sample task, serial data is received in asynchronous mode via the serial communication interface (SCI). Figure 2 is a block diagram of serial data 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 Serial Data 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 ($0 \leq N \leq 255$)

OSC: Value of ϕOSC (MHz) = 16 MHz

n: Value set in bits CKS1 and CKS0 in SMR ($0 \leq n \leq 3$)
(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.

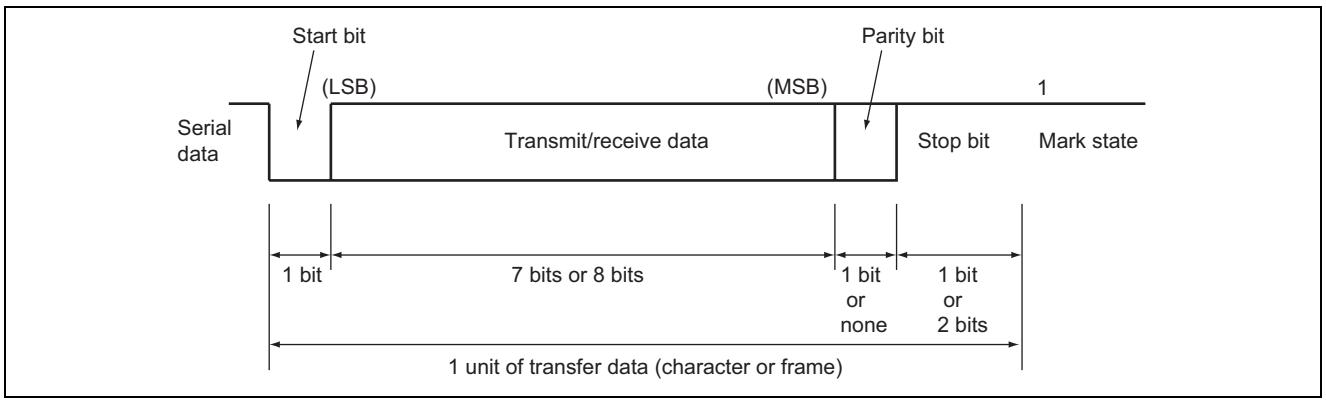


Figure 3 Data Format in Asynchronous Communication

- 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 serial data 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
SCK3	I/O port
RXD	SCI3 receive data input pin

3. Operational Description

Figure 4 shows this sample task's principle of operation. The hardware and software processing shown in figure 4 performs serial data reception in asynchronous mode.

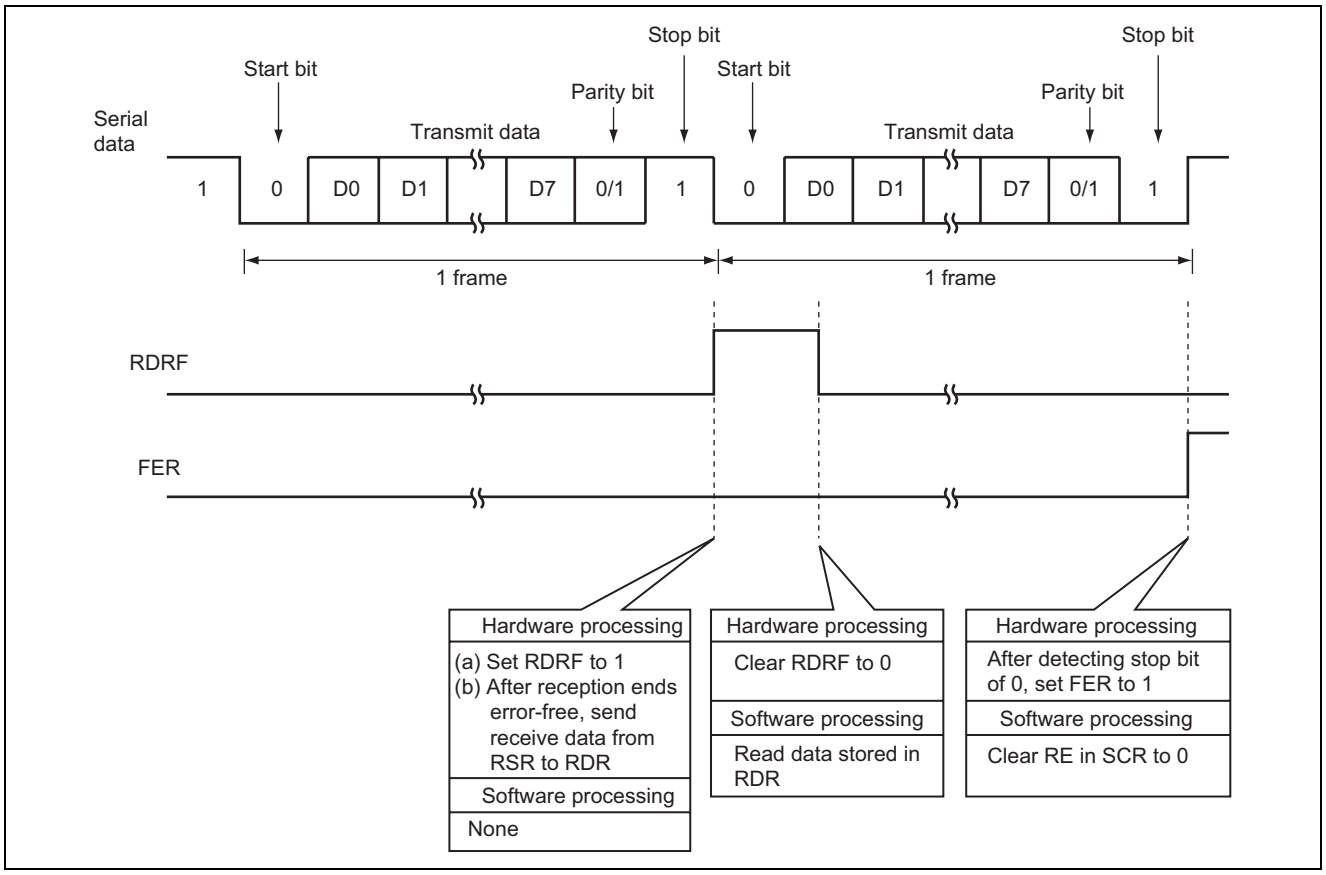


Figure 4 Operation Principle: Serial Data Reception in Asynchronous Mode

4. Description of Software

4.1 Description of Modules

Table 4 describes the software used in this sample task.

Table 4 Description of Module

Module Name	Label Name	Function
Main routine	main	Selects serial data reception in asynchronous mode, stores H'FF in SRD0 to SRD3 when a receive error occurs, and stops SCI3 after four bytes of data have been received.

4.2 Description of Arguments

Table 5 describes the argument used in this sample task.

Table 5 Description of Argument

Argument Name	Function	Used in	Data Length	I/O
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 (cont)

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	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 and CKE0 are both cleared to 0, the clock source is set to an internal clock and the SCK3 pin functions as an I/O port in asynchronous mode.	H'FFAA Bit 1 Bit 0	CKE1 = 0 CKE0 = 0
RDR		Receive data register: 8-bit register that stores the receive data.	H'FFAD	—
SSR	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

4.4 Description of RAM

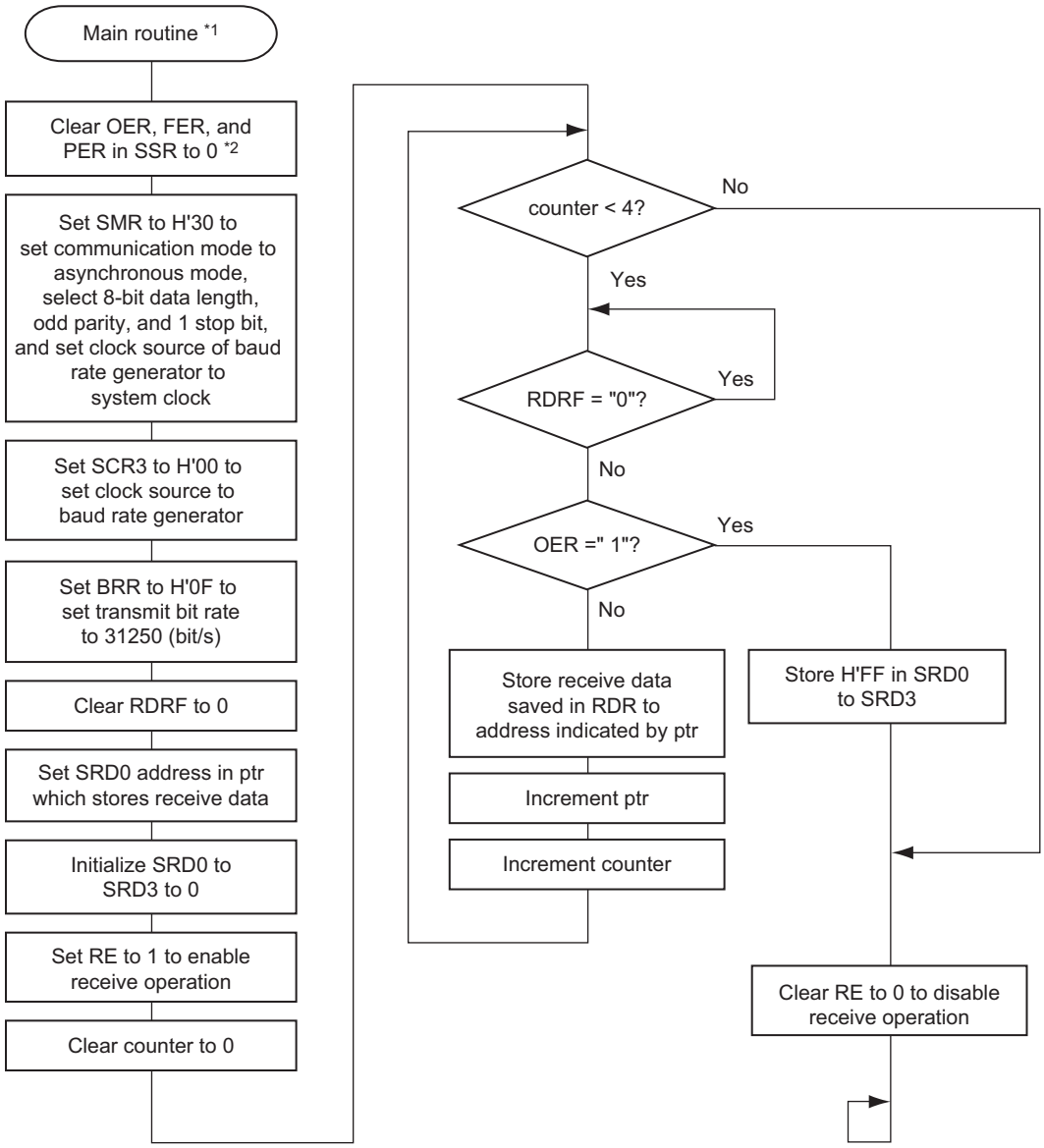
Table 7 describes the RAM used in this sample task.

Table 7 Description of RAM

Label Name	Function	Address	Used in
SRD0	Receives the first byte of receive data in serial data reception in asynchronous mode	H'FB80	Main routine
SRD1	Receives the second byte of receive data in serial data reception in asynchronous mode	H'FB81	Main routine
SRD2	Receives the third byte of receive data in serial data reception in asynchronous mode	H'FB82	Main routine
SRD3	Receives the fourth byte of receive data in serial data reception in asynchronous mode	H'FB83	Main routine
counter	8-bit counter for counting four receive operations in serial data reception in asynchronous mode	H'FB84	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 (reception), OER, FER, and PER must be cleared to 0.

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 Reception'                    */
/*                                                          */
/*  Function                                                */
/*  : Serial Communication Interface                        */
/*     Asynchronous Serial Interface                        */
/*     -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    RDR      *(volatile unsigned char *)0xFFAD  /* Receive data Register           */

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

/************************************************************/
/*   RAM Allocation                                     */
/************************************************************/
    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    *ptr;

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

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

    SCR3 = 0x00;                                /* Initialize Serial Control Register 3   */

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

    RDRF = 0;                                   /* Clear RDRF                             */


    ptr = &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     */

    RE = 1;                                     /* Start Serial Receiving                 */

    counter = 0;                                /* Clear counter                          */

    while (counter < 4){                        /* Serial Receiving Data Counter 4 Loop   */

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

```
if (OER == 1){                                  /* Overrun Erorr Flag = 1 ?                */
            SRD[0] = 0xFF;                      /* Overrun Erorr 0                         */
            SRD[1] = 0xFF;                      /* Overrun Erorr 1                         */
            SRD[2] = 0xFF;                      /* Overrun Erorr 2                         */
            SRD[3] = 0xFF;                      /* Overrun Erorr 3                         */
            break;
        }
        else    {
            *ptr = RDR;                         /* Save Serial Receiving Data              */
            ptr++;                              /* Increment Serial Receiving Data Address */
            counter++;                          /* Increment counter                       */
        }
    }

    RE = 0;                                     /* Initialize Receiving Enable             */

    while(1){
       ;
    }
}
```

Revision Record

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