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.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

32176 Group

Application Using the Serial Interface (UART Mode)

1. Overview

This documentation describes a sample program using 32176 Group microcomputer's on-chip serial interface (serial I/O).

2. Introduction

The example application described in this documentation uses the following microcomputer under the respective conditions.

- Microcomputer: 32176 Group (M32176FnVFP, M32176FnTFP)
- Operating frequency: 20 to 40 MHz (The sample program is compiled assuming a frequency of 40 MHz)
- Operating Board: Starter kit for 32176 Group

3. Explanation of the Technology Applied

3.1 Outline of the Serial Interface

The 32176 contains a total of four serial interface channels, SIO0, SIO1, SIO2, and SIO3. SIO0 and SIO1 can be selected between CSIO mode (clock-synchronous serial interface) and UART mode (clock-asynchronous serial interface). Channels SIO2 and SIO3 are UART mode only.

- CSIO Mode (clock-synchronous serial interface)
  Communication is performed synchronously with a transfer clock, using the same clock for both transmit and receive sides. The transfer data is 8 bits long (fixed).
- UART Mode (clock-asynchronous serial interface)
  Communication is performed at any transfer rate in any transfer data format. The transfer data length can be selected from 7, 8, or 9 bits.

Channels SIO0, SIO1, SIO2, and SIO3 each have transmit DMA transfer and a receive DMA transfer request. These serial interfaces, when combined with the internal DMA Controller (DMAC), allow serial communication to be performed at high speed, as well as reduce the data communication load of the CPU.

In the UART mode, the transfer rate is decided by clock divider frequency division value and the values set in the individual baud rate registers (BRG) for each channel. The table below shows sample setting values for each baud rate.

Table 3.1.1 Sample Baud Rate Register Settings

Item / Baud Rate (bps)	Clock Divider Count Source = 20 MHz			
	Clock Divider Frequency Division Value (Frequency Division)	BRG Setting Value	Error (%)	Actual Baud Rate (bps)
300	32	129	0.16	300.48
600	32	64	0.16	600.96
1200	8	129	0.16	1201.92
2400	8	64	0.16	2403.85
4800	8	32	–1.36	4734.85
9600	1	129	0.16	9615.38
12500	1	99	0.00	12500.00
14400	1	86	–0.22	14367.82
19200	1	64	0.16	19230.77
28800	1	42	0.94	29069.77
31250	1	39	0.00	31250.00
38400	1	32	–1.36	37878.79
57600	1	21	–1.36	56818.18
62500	1	19	0.00	62500.00
115200	1	10	–1.36	113636.36
125000	1	9	0.00	125000.00
250000	1	4	0.00	250000.00
500000	1	2	–16.67	416666.67
625000	1	1	0.00	625000.00
1000000	—	—	—	—
1250000	1	0	0.00	1250000.00

In the UART mode, the value to be set in the baud rate register can be calculated as follows:

$$\text{BRG setting value} = \frac{f(\text{BCLK})}{\text{Baud rate (bps)} \times \text{clock divider frequency division value} \times 16} - 1$$

f (BCLK) : Peripherial clock operating frequency
Baud rate register setting value (BRG) : H'00 to H'FF
Clock divider frequency division value : 1, 8, 32, 256

For details of the serial interface, refer to the 32176 Group User's Manual.

4. Sample Program Using the Serial Interface

4.1 Outline of the Sample Program

In the sample program, data is output from the SIO0 and data is received on the SIO1. Both communications use 8-bit UART. For transmission, data for the number of transmission bytes is transmitted from the address of the transmit data storage area specified by the parameter. For reception, interrupt processing is used. By connecting the SIO0 and the SIO1, full duplex communication is enabled by the on-chip UART functions of the 32176.

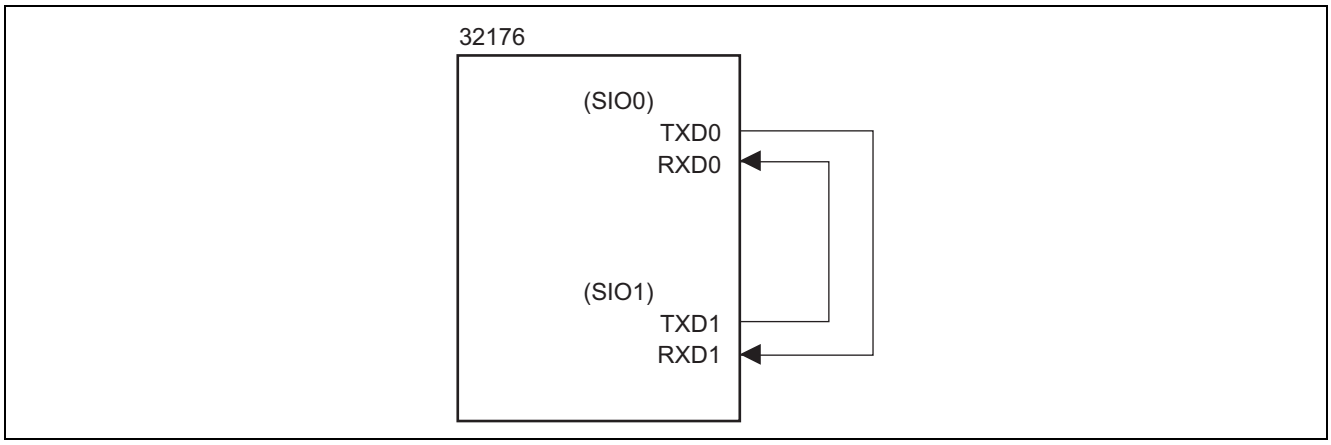


Figure 4.1.1 Connection Diagram

4.2 Transmit/Receive Processing

The flowchart for the initial settings of transmission and reception in the UART mode is shown in figure 4.2.1.

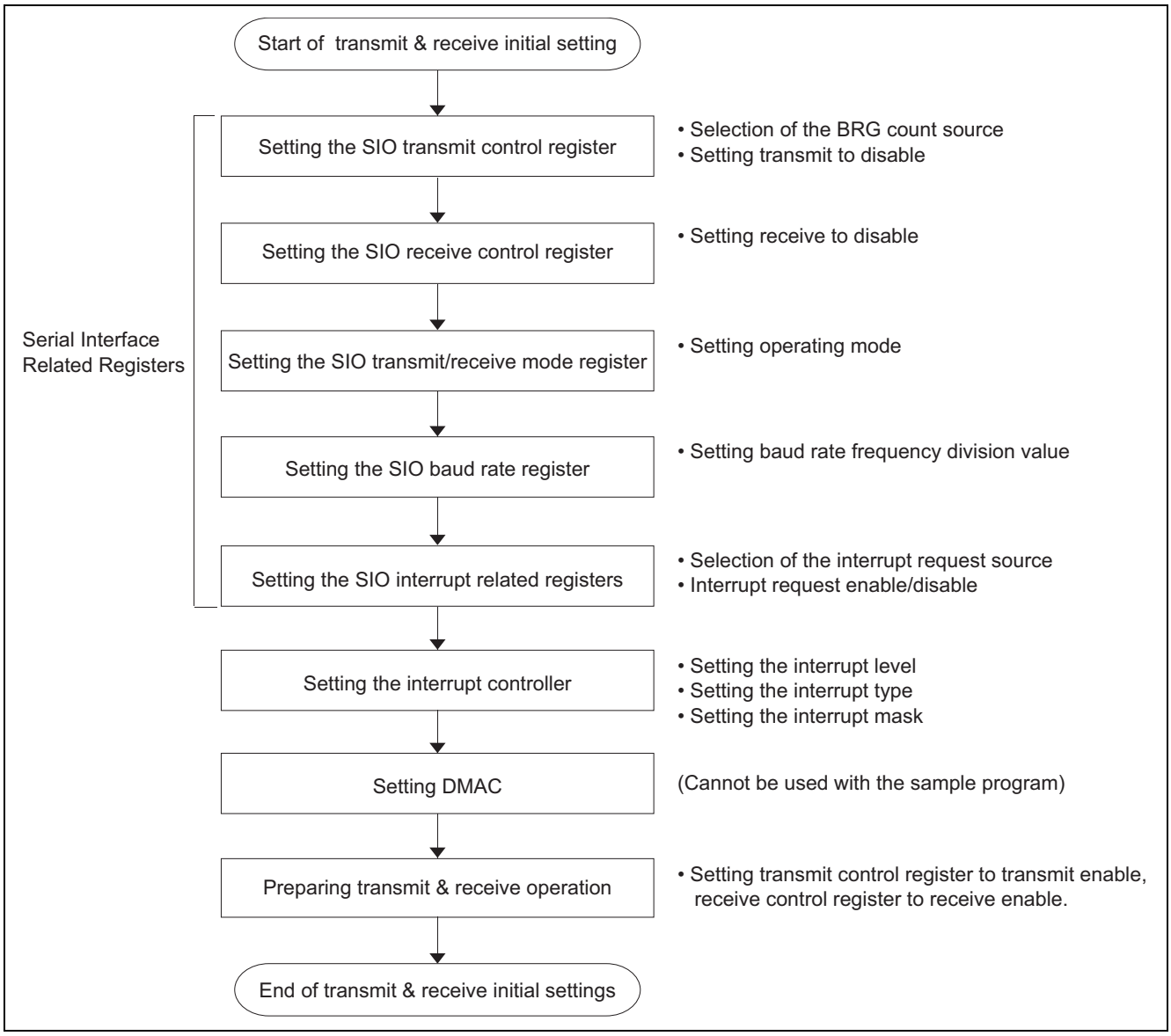


Figure 4.2.1 Flowchart for the Initial Settings of Transmission and Reception in the UART Mode

4.3 Description of the Sample Program

Note: Registers used are indicated as (register name: bit name).

4.3.1 Various Initialization Function (init_func ())

(1) Call the function for port initialization.

4.3.2 Port Initialization Function (port_init ())

(1) Set the output port.

- Set the port input enable bit of the port input special function control register to input enabled. (PICNT: PIEN0)
- Initialize the P11 data register. (P11DATA)
- Set the P11 direction register to the output mode. (P11DIR)
- Set the P11 operation mode register to input/output port. (P11MOD)

Note: Setting a direction register to output before setting the corresponding data register, causes undefined values to be output until the data register is written to.

4.3.3 Main Function (main ())

(1) Initialize local variables.

- Received data buffer pointer (rx_cnt1), previous received data buffer pointer (rx_cnt1_bak), and transmission data (send_data) are initialized to “0”.

(2) Call the function for disabling interrupts.
(3) Call the function for various initialization.
(4) Call the function for SIO0 transmit processing initialization.
(5) Call the function for SIO1 receive processing initialization.
(6) Call the function for enabling interrupts.
(7) Transmit/receive processing infinite loop.

- Transmit one byte of variable send_data from SIO0.
- Wait for the SIO1 receive interrupt.
- Retrieve received data from SIO1 receive data buffer.
- Re-set to 0 when the SIO1 receive data buffer pointer overflow occurs.
- Store the current receive data buffer pointer.
- When SIO0 transmission data and SIO1 received data have the same value:
  1) Displays the received data on LED.
  2) Increments the transmission data and start next transmission processing.
- When SIO0 transmission data and SIO1 received data differ:
  1) Enters an infinite loop, and displays H'55 and H'AA alternatively on the LED.

4.3.4 SIO0 Initialization Function (SIO0_init ())

(1) Setting the transfer mode.

- Set the receive enable bit of the SIO0 receive control register to receive disabled. (S0RCNT: REN)
- Set the SIO0 transmit control register. (S0TCNT: CDIV, TEN)
  Set the BRG count source is set to f (BCLK), transmission disabled.
- Set the P8 operation mode register. (P8MOD: P82MOD, P83MOD)
  Set port P82 to TXD0, and port P83 to RXD0.
- Set the SIO0 transmit/receive mode register. (S0MOD: SMOD, CKS, STB, PEN, SEN)
  Set the 8-bit UART using the internal clock, with one stop bit, parity disabled, sleep function invalid.
- Set the baud rate frequency division value to SIO0 baud rate register. (S0BAUR)

(2) Interrupt settings.

- Set the interrupt level in the SIO0 transmission interrupt control register to interrupt disabled.
  (ISIO0TXCR: ILEVEL)
- Set the interrupt level in the SIO0 receive interrupt control register to “0” (highest priority).
  (ISIO0RXCR: ILEVEL)
- Set the SIO0 receive interrupt request source select bit of the SIO03 interrupt request source select register to reception finished interrupt. (SI03SEL: ISR0)
- Set the SIO03 interrupt request mask register. (SI03MASK: T0MASK, R0MASK)
  Disable SIO0 transmit interrupts, and enable SIO0 receive interrupts.

(3) Setting to enable transmission and reception.

- Set the transmit enable bit of the SIO0 transmit control register to transmit enabled. (S0TCNT: TEN)
- Set the receive enable bit of the SIO0 receive control register to receive enabled. (S0RCNT: REN)

4.3.5 SIO1 Initialization Function (SIO1_init ())

(1) Setting the transfer mode.

- Set the receive enable bit of the SIO1 receive control register to receive disabled. (S1RCNT: REN)
- Set the SIO1 transmit control register. (S0TCNT: CDIV, TEN)
  Set the BRG count source to f (BCLK), transmit disabled.
- Set the P8 operation mode register. (P8MOD: P85MOD, P86MOD)
  Set the port P85 to TXD1, and port P86 to RXD1.
- Set the SIO1 transmit/receive mode register. (S1MOD: SMOD, CKS, STB, PEN, SEN)
  Set the 8-bit UART using the internal clock, with one stop bit, parity disabled, sleep function invalid.
- Set the baud rate frequency division value in the SIO1 baud rate register. (S1BAUR)

(2) Interrupt settings.

- Set the interrupt level in the SIO1 transmission interrupt control register to interrupts disabled.
  (ISIO1TXCR:ILEVEL)
- Set the interrupt level in the SIO1 receive interrupt control register to “0” (highest priority).
  (ISIO1RXCR:ILEVEL)
- Set the SIO1 receive interrupt request source select bit of the SIO03 interrupt request source select register to reception finished interrupt. (SI03SEL: ISR1)
- Set the SIO03 interrupt request mask register. (SI03MASK: T1MASK, R1MASK)
  Disable SIO1 transmit interrupts, and enable SIO1 reception interrupts.

(3) Enable transmission and reception.

- Set the transmit enable bit of the SIO1 transmit control register to transmission enabled. (S1TCNT: TEN)
- Set the receive enable bit of the SIO1 receive control register to reception enabled. (S1RCNT: REN)

4.3.6 SIO2 Initialization Function (SIO2_init ())

Set in the same way as SIO0_init.

Note: This function is not used in this sample program.

4.3.7 SIO3 Initialization Function (SIO3_init ())

Set in the same way as SIO0_init.

Note: This function is not used in this sample program.

4.3.8 UART Transmit Processing Function (SIO0_Tr (), SIO1_Tr (), SIO2_Tr (), SIO3_Tr ())

(1) Transmit processing.

- Confirm that the SIO transmit buffer is empty. (S0TCNT: TBE)
- Write one byte data of transmission data buffer to transmit buffer register. (S0TXB_L)
- Repeat transmission for a specified number of times.
- Wait for SIO transmit completion. (S0TCNT: TSTAT)

Note: SIO1_Tr (), SIO2_Tr (), and SIO3_Tr () are not used in this sample program.

4.3.9 The UART Receive Processing Function by SIO Receive Interrupt (SIO0_RcvInt (), SIO1_RcvInt (), SIO2_RcvInt (), SIO3_RcvInt ())

(1)Receive processing.

- Read the SIO receive control register (receive status). (S1RCNT)
- Read the SIO receive buffer register. (S1RXB_L)
- Re-read the SIO receive control register.
  For detecting overrun occurring between reading the receive status and reading the receive buffer register.
- Check SIO receive error sum bit. (S1RCNT: ERS)
  1) When no error occurs, the received data is stored in the receive data buffer and the pointer is incremented.
  2) When an error occurs, the receive enable bit is first set to receive disabled and then set to receive enabled. (Each bit in the receive control registers is cleared.)

Note: SIO0_RcvInt (), SIO2_RcvInt (), SIO3_RcvInt () are not used in this sample program.

4.3.10 Interrupt Handler Function of SIO2, 3 (IntHand_SIO23 ())

Since the SIO2 and SIO3 interrupts are a group interrupt, which interrupt request occurred is determined by the interrupt request status bit of the SIO23 interrupt request status register.
(SI23STAT: IRQR2, IRQR3)

Note: This function is not used in this sample program.

4.3.11 Startup Routine (startup.ms)

(1) Interrupt settings.

- Set the start address of the SIO1 receive interrupt routine (SIO1_RcvInt ()) in the SIO1 receive interrupt handler (H'0000 00CC) as the interrupt source in the ICU vector table.

4.4 Sample Program

The transmit/receive sample program for the UART mode is shown below.

Note that the sample program below requires the SFR definition file. The latest SFR definition file can be downloaded from Renesas Technology website. When using the SFR definitions file, adjust the path setting to match the operating computer environment.

4.4.1 uart_main.c

```
/*""FILE COMMENT""*****************************************************
 *      M32R C Programming              Rev. 1.01
 *               < Sample Program for 32176 >
 *               < Serial I/O (UART) (main routine) >
 *
 *      Copyright (c) 2004 Renesas Technology Corporation
 *                         All Rights Reserved
 *********************************************************************/

/*********************************************************************/
/*              Include file                                         */
/*********************************************************************/

#include                "..\inc\sfr32176_pragma.h"

/*********************************************************************/
/*              Function prototype declaration                       */
/*********************************************************************/

        void            main(void);                             /* Main function */
        void            init_func(void);                        /* Initial setup function */
        void            port_init(void);                        /* Initialize port */

/*********************************************************************/
/*              Definition of external reference                     */
/*********************************************************************/

extern  void            DisInt( void );                         /* Interrupt disable function
*/
extern  void            EnInt( void );                          /* Interrupt enable function
*/

extern  void            SIO0_init( void);                       /* Initialize SIO0 */
extern  void            SIO1_init( void);                       /* Initialize SIO1 */
extern  void            SIO0_Tr( UCHAR *TrBuf, ULONG TrNum);    /* Transmit from SIO0 */
extern  void            SIO1_Tr( UCHAR *TrBuf, ULONG TrNum);    /* Transmit from SIO1 */

/*              Global variable                                      */
/*              Global variable                                      */
/*********************************************************************/

        volatile UCHAR  RcvBuf0[10];                            /* Receive buffer */
        volatile UCHAR  RcvBuf1[10];                            /* Receive buffer */
        volatile UCHAR  RcvBuf2[10];                            /* Receive buffer */
        volatile UCHAR  RcvBuf3[10];                            /* Receive buffer */
        volatile UCHAR  rx_cnt0;                                /* Pointer to receive buffer
*/
        volatile UCHAR  rx_cnt1;                                /* Pointer to receive buffer
*/
        volatile UCHAR  rx_cnt2;                                /* Pointer to receive buffer
*/
        volatile UCHAR  rx_cnt3;                                /* Pointer to receive buffer
*/

/*""FUNC COMMENT""*****************************************************
 * Function name: init_func()
 *---------------------------------------------------------------------
 * Description  : Call various initialization functions
 *---------------------------------------------------------------------
 * Argument     : -
 *---------------------------------------------------------------------
 * Returns      : -
 *---------------------------------------------------------------------
```

```
 58   * Notes        : -
 59   *""FUNC COMMENT END""************************************************/
 60  void init_func(void)
 61  {
 62          port_init();                                        /* Initialize port */
 63  }
 64
 65  /*""FUNC COMMENT""*****************************************************
 66   * Function name: port_init()
 67   *---------------------------------------------------------------------
 68   * Description  : Port initialization
 69   *---------------------------------------------------------------------
 70   * Argument     : -
 71   *---------------------------------------------------------------------
 72   * Returns      : -
 73   *---------------------------------------------------------------------
 74   * Notes        : -
 75   *""FUNC COMMENT END""************************************************/
 76  void port_init(void)
 77  {
 78          PICNT = PIEN0;                                      /* Enable port input */
 79
 80  /*** LED output port ***/
 81
 82          P11DATA = 0x00;                                     /* Output data(must be set
prior to mode)*/
 83          P11DIR = 0xff;                                      /* P110-P117 : Output mode */
 84          P11MOD = 0x00;                                      /* P110-P117 : Input/output
port */
 85  }
 86
 87  /*""FUNC COMMENT""*****************************************************
 88   * Function name: main()
 89   *---------------------------------------------------------------------
 90   * Description  : Receives data (increment pattern) from SIO1 after being transmitted in 500
Kbps clock-synchronized
 91   *              : serial mode (when the source clock frequency = 10 MHz) from SIO0 and
 92   *              : if the received data is the same as transmitted, displays it on LED(port
11).
 93   *              : If unable to receive the same data, it displays 0x55 in blinking inverse
video.
 94   *---------------------------------------------------------------------
 95   * Argument     : -
 96   *---------------------------------------------------------------------
 97   * Returns      : -
 98   *---------------------------------------------------------------------
 99   * Notes        : -
100   *""FUNC COMMENT END""************************************************/
101  void    main(void)
102  {
103  /*              +-------+
104   *              |       |
105   *      (SIO0)  |   TXD0| --+
106   *              |       |   |
107   *              |       |   |
108   *      (SIO1)  |   RXD1| <-+
109   *              |       |
110   *      (SIO1)  |   TXD1| --+
111   *              |       |   |
112   *              |       |   | (not use)
113   *      (SIO0)  |   RXD0| <-+
114   *              |       |
115   *              +-------+
116          SI03MASK &= ~T0MASK;                                /*Disable SIO0 transmit
interrupt request*/
117
118          ULONG   j;
119          UCHAR   send_data;
120          UCHAR   rcv_data;
121          UCHAR   rx_cnt1_bak;
122
123          rx_cnt1 = 0;
124          rx_cnt1_bak = 0;
125
126          send_data = 0;
127
128          DisInt();                                           /* Disable interrupt */
```

```
129
130         init_func();                                        /* Initialize microcomputer
*/
131
132         SIO0_init();                                        /* Initialize SIO0 */
133         SIO1_init();                                        /* Initialize SIO1 */
134
135         EnInt();                                            /* Enable interrupt */
136
137         while(1) {
138                 SIO0_Tr( &send_data, 1ul);                  /* Send data from SIO0 */
139
140                 while( rx_cnt1 == rx_cnt1_bak){             /* Wait for data to receive
*/
141                         ;
142                 }
143                 rcv_data = RcvBuf1[ rx_cnt1 - 1];           /* Read out received data */
144
145                 if( rx_cnt1 == 10) {                        /* Check for receive counter
overflow */
146                         rx_cnt1 = 0;
147                 }
148                 rx_cnt1_bak = rx_cnt1;
149
150                 if( rcv_data == send_data) {
151                         P11DATA = rcv_data;                 /* Display received
(transmitted) data */
152                         send_data++;                        /* Increment transmit data */
153                 } else {                                    /* Received data in error */
154                         P11DATA = 0x55;
155                         while(1) {
156                                 for( j = 0ul; j < 1000000ul; j++){      /* Wait */
157                                         ;
158                                 }
159                                 P11DATA ^= 0xffu;           /* Display 0x55 in blinking
inverse video*/
160                         }
161                 }
162         }
163  }
```

4.4.2 uart.c

```
  1  /*""FILE COMMENT""*****************************************************
  2   *      M32R C Programming              Rev. 1.01
  3   *               < Sample Program for 32176 >
  4   *              < Serial Interface (UART) >
  5   *
  6   *      Copyright (c) 2004 Renesas Technology Corporation
  7   *                         All Rights Reserved
  8   ************************************************************************/
  9
 10  /************************************************************************/
 11  /*              Include file                                            */
 12  /************************************************************************/
 13
 14  #include               "..\inc\sfr32176_pragma.h"
 15
 16  /************************************************************************/
 17  /*              Function prototype declaration                          */
 18  /************************************************************************/
 19
 20          void            SIO0_init( void);                       /* Initialize SIO0 */
 21          void            SIO1_init( void);                       /* Initialize SIO1 */
 22          void            SIO2_init( void);                       /* Initialize SIO2 */
 23          void            SIO3_init( void);                       /* Initialize SIO3 */
 24          void            SIO0_Tr( UCHAR *TrBuf, ULONG TrNum);    /* Transmit from SIO0 */
 25          void            SIO1_Tr( UCHAR *TrBuf, ULONG TrNum);    /* Transmit from SIO1 */
 26          void            SIO2_Tr( UCHAR *TrBuf, ULONG TrNum);    /* Transmit from SIO2 */
 27          void            SIO3_Tr( UCHAR *TrBuf, ULONG TrNum);    /* Transmit from SIO3 */
 28          void            SIO0_RcvInt(void);                      /* SIO0 receive interrupt */
 29          void            SIO1_RcvInt(void);                      /* SIO1 receive interrupt */
 30          void            SIO2_RcvInt(void);                      /* SIO2 receive interrupt */
 31          void            SIO3_RcvInt(void);                      /* SIO3 receive interrupt */
 32          void            IntHand_SIO23(void);                    /* SIO2,3 transmit/receive
interrupt handler */
 33
 34  /************************************************************************/
 35  /*              Externally referenced variable                          */
 36  /*              Global variable                                         */
 37
 38  extern  volatile UCHAR  RcvBuf0[];                              /* Receive buffer */
 39  extern  volatile UCHAR  RcvBuf1[];                              /* Receive buffer */
 40  extern  volatile UCHAR  RcvBuf2[];                              /* Receive buffer */
 41  extern  volatile UCHAR  RcvBuf3[];                              /* Receive buffer */
 42  extern  volatile UCHAR  rx_cnt0;                                /* Pointer to receive buffer
*/
 43  extern  volatile UCHAR  rx_cnt1;                                /* Pointer to receive buffer
*/
 44  extern  volatile UCHAR  rx_cnt2;                                /* Pointer to receive buffer
*/
 45  extern  volatile UCHAR  rx_cnt3;                                /* Pointer to receive buffer
*/
 46
 47  /************************************************************************/
 48  /*              Define macro                                            */
 49  /************************************************************************/
 50
 51  /* Setting port operation mode */
 52                                                                  /* 0123 4567    */
 53  #define P8MOD_SCI0      0x30u                                   /* 0011 0000B   P8 operation mode
register      */
 54                                                                  /* |||| |||+--- P87
*/
 55                                                                  /* |||| ||+---- P86
*/
 56                                                                  /* |||| |+----- P85
*/
 57                                                                  /* |||| +------ P84
*/
 58                                                                  /* |||+-------- RXD0
*/
 59                                                                  /* ||+--------- TXD0
*/
 60                                                                  /* ++---------- don't care
*/
 61
```

```
 62                                                    /* 0123 4567    */
 63  #define P8MOD_SCI1      0x06u                     /* 0000 0110B   P8 operation mode
register      */
 64                                                    /* |||| |||+--- P87
*/
 65                                                    /* |||| ||+---- RXD1
*/
 66                                                    /* |||| |+----- TXD1
*/
 67                                                    /* |||| +------ P84
*/
 68                                                    /* |||+-------- P83
*/
 69                                                    /* ||+--------- P82
*/
 70                                                    /* ++---------- don't care
*/
 71
 72                                                    /* 0123 4567    */
 73  #define P17MOD_SCI2     0x0cu                     /* 0000 1100B   P17 operation mode
register     */
 74                                                    /* |||| |||+--- P177
*/
 75                                                    /* |||| ||+---- P176
*/
 76                                                    /* |||| |+----- RXD2
*/
 77                                                    /* |||| +------ TXD2
*/
 78                                                    /* |||+-------- P173
*/
 79                                                    /* ||+--------- P172
*/
 80                                                    /* ++---------- don't care
*/
 81
 82                                                    /* 0123 4567    */
 83  #define P17MOD_SCI3     0x03u                     /* 0000 0011B   P17 operation mode
register     */
 84                                                    /* |||| |||+--- RXD3
*/
 85                                                    /* |||| ||+---- TXD3
*/
 86                                                    /* |||| |+----- P175
*/
 87                                                    /* |||| +------ P174
*/
 88                                                    /* |||+-------- P173
*/
 89                                                    /* ||+--------- P172
*/
 90                                                    /* ++---------- don't care
*/
 91
 92  /* Setting serial IO */
 93                                                    /* 0123 4567    */
 94  #define SnTCNT_INI      0x00u                     /* 0000 0000B   SIOn transmit control
register  */
 95                                                    /* |||| |||+--- Disable transmission
*/
 96                                                    /* |||| +++----- don't care
*/
 97                                                    /* ||++-------- f(BCLK)
*/
 98                                                    /* ++---------- don't care
*/
 99
100                                                    /* 0123 4567    */
101  #define SnMOD_INI       0x20u                     /* 0010 0000B   SIOn mode register
*/
102                                                    /* |||| |||+--- Sleep function
disabled */
103                                                    /* |||| ||+---- Parity inhibited
*/
104                                                    /* |||| |+----- don't care(odd)
*/
```

```
105                                                     /* |||| +------ 1 stop bit
*/
106                                                     /* |||+-------- Internal clock
*/
107                                                     /* +++--------- 8-bit UART
*/
108
109  /* Setting interrupt priority level */
110
111  #define SioILEVEL       0x0u                       /* Serial IO transmit/receive
interrupt priority level*/
112  #define ILEVEL_7        0x7u                       /* Interrupt Disable
*/
113
114  /* Setting baud rate (Be sure to check actually set value when using) */
115
116  #define XIN             10                         /* 10MHz */
117
118  #define BAUD_62_5       (XIN * 2000000 / 16 / 62500 - 1)        /* 62.5Kbps */
119  #define BAUD_15_6       (XIN * 2000000 / 16 / 15600 - 1)        /* 15.6Kbps */
120  #define BAUD_9615       (XIN * 2000000 / 16 / 9615 - 1)         /* 9615bps */
121
122
123  /*""FUNC COMMENT""************************************************
124   * Function name: SIO0_init()
125   *-----------------------------------------------------------------
126   * Description  : Sets SIO0 for 8-bit UART
127   *              : - Reception by interrupt
128   *              : - Program transmission
129   *-----------------------------------------------------------------
130   * Argument     : -
131   *-----------------------------------------------------------------
132   * Returns      : -
133   *-----------------------------------------------------------------
134   * Notes        : Port input function must be enabled
135   *              : Must be executed while interrupts are disabled
136   *""FUNC COMMENT END""*******************************************/
137  void    SIO0_init( void)
138  {
139
140  /*** Setting transfer mode */
141
142          S0RCNT = 0x00;                                  /* Disable reception */
143          S0TCNT = SnTCNT_INI;                            /* f(BCLK) and disable
transmission */
144          P8MOD |= P8MOD_SCI0;                            /* Set P8 for SIO mode */
145          S0MOD = SnMOD_INI;                              /* Set data format */
146          S0BAUR = BAUD_62_5;                             /* Set baud rate */
147
148  /*** Interrupt related settings ***/
149
150          ISIO0TXCR = ILEVEL_7;                           /* Set SIO0 transmit interrupt
priority level*/
151          ISIO0RXCR = SioILEVEL;                          /* Set SIO0 receive interrupt
priority level */
152          SI03SEL &= ~ISR0;                               /* Select receive-finished
interrupt */
153          SI03MASK &= ~T0MASK;                            /* Disable SIO0 transmit
interrupt request */
154          SI03MASK |= R0MASK;                             /* Enable SIO0 receive
interrupt request */
155
156  /*** Starting transmission/reception ***/
157
158          S0TCNT |= TEN;                                  /* Enable transmission */
159          S0RCNT |= REN;                                  /* Enable reception */
160  }
161
162  /*""FUNC COMMENT""************************************************
163   * Function name: SIO1_init()
164   *-----------------------------------------------------------------
165   * Description  : Sets SIO1 for 8-bit UART
166   *              : - Reception by interrupt
167   *              : - Program transmission
168   *-----------------------------------------------------------------
169   * Argument     : -
170   *-----------------------------------------------------------------
```

```
171   * Returns      : -
172   *------------------------------------------------------------------------
173   * Notes        : Port input function must be enabled
174   *              : Must be executed while interrupts are disabled
175   *""FUNC COMMENT END""***************************************************/
176  void    SIO1_init( void)
177  {
178
179  /*** Setting transfer mode */
180
181           S1RCNT = 0x00;                                          /* Disable reception */
182           S1TCNT = SnTCNT_INI;                                    /* f(BCLK) and disable
transmission */
183           P8MOD |= P8MOD_SCI1;                                    /* Set P8 for SIO mode */
184           S1MOD = SnMOD_INI;                                      /* Set data format */
185           S1BAUR = BAUD_62_5;                                     /* Set baud rate */
186
187  /*** Interrupt related settings ***/
188
189           ISIO1TXCR = ILEVEL_7;                                   /* Set SIO1 transmit interrupt
priority level*/
190           ISIO1RXCR = SioILEVEL;                                  /* Set SIO1 receive interrupt
priority level */
191           SI03SEL &= ~ISR1;                                       /* Select receive-finished
interrupt */
192           SI03MASK &= ~T1MASK;                                    /* Disable SIO1 transmit
interrupt request */
193           SI03MASK |= R1MASK;                                     /* Enable SIO1 receive
interrupt request */
194
195  /*** Starting transmission/reception ***/
196
197           S1TCNT |= TEN;                                          /* Enable transmission */
198           S1RCNT |= REN;                                          /* Enable reception */
199  }
200
201  /*""FUNC COMMENT""*******************************************************
202   * Function name: SIO2_init()
203   *------------------------------------------------------------------------
204   * Description  : Sets SIO2 for 8-bit UART
205   *              : - Reception by interrupt
206   *              : - Program transmission
207   *------------------------------------------------------------------------
208   * Argument     : -
209   *------------------------------------------------------------------------
210   * Returns      : -
211   *------------------------------------------------------------------------
212   * Notes        : Port input function must be enabled
213   *              : Must be executed while interrupts are disabled
214   *              : Interrupt priority levels are common to both SIO2,3 transmission and
reception
215   *""FUNC COMMENT END""***************************************************/
216  void    SIO2_init( void)
217  {
218
219  /*** Setting transfer mode */
220
221           S2RCNT = 0x00;                                          /* Disable reception */
222           S2TCNT = SnTCNT_INI;                                    /* f(BCLK) and disable
transmission */
223           P17MOD |= P17MOD_SCI2;                                  /* Set P17 for SIO mode */
224           S2MOD = SnMOD_INI;                                      /* Set data format */
225           S2BAUR = BAUD_62_5;                                     /* Set baud rate */
226
227  /*** Interrupt related settings ***/
228
229           ISIO23CR = SioILEVEL;                                   /* Set SIO2,3 transmission
and reception interrupt priority level */
230           SI03SEL &= ~ISR2;                                       /* Select receive-finished
interrupt */
231           SI03MASK &= ~T2MASK;                                    /* Disable SIO2 transmit
interrupt request */
232           SI03MASK |= R2MASK;                                     /* Enable SIO2 receive
interrupt request */
233           SI23STAT = (~( IRQT2 | IRQR2)) & 0x0fu;                 /* Clear interrupt request */
234
235  /*** Starting transmission/reception ***/
```

```
236
237          S2TCNT |= TEN;                                          /* Enable transmission */
238          S2RCNT |= REN;                                          /* Enable reception */
239   }
240
241   /*""FUNC COMMENT""*****************************************************
242    * Function name: SIO3_init()
243    *----------------------------------------------------------------------
244    * Description  : Sets SIO3 for 8-bit UART
245    *              : - Reception by interrupt
246    *              : - Program transmission
247    *----------------------------------------------------------------------
248    * Argument     : -
249    *----------------------------------------------------------------------
250    * Returns      : -
251    *----------------------------------------------------------------------
252    * Notes        : Port input function must be enabled
253    *              : Must be executed while interrupts are disabled
254    *              : Interrupt priority levels are common to both SIO2,3 transmission and
reception
255    *""FUNC COMMENT END""*************************************************/
256   void    SIO3_init( void)
257   {
258
259   /*** Setting transfer mode */
260
261          S3RCNT = 0x00;                                          /* Disable reception */
262          S3TCNT = SnTCNT_INI;                                    /* f(BCLK) and disable
transmission */
263          P17MOD |= P17MOD_SCI3;                                  /* Set P17 for SIO mode */
264          S3MOD = SnMOD_INI;                                      /* Set data format */
265          S3BAUR = BAUD_62_5;                                     /* Set baud rate */
266
267   /*** Interrupt related settings ***/
268
269          ISIO23CR = SioILEVEL;                                   /* Set SIO2,3 transmission
and reception interrupt priority level */
270          SI03SEL &= ~ISR3;                                       /* Select receive-finished
interrupt */
271          SI03MASK &= ~T3MASK;                                    /* Disable SIO3 transmit
interrupt request */
272          SI03MASK |= R3MASK;                                     /* Enable SIO3 receive
interrupt request */
273          SI23STAT = (~( IRQT3 | IRQR3)) & 0x0fu;                 /* Clear interrupt request */
274
275   /*** Starting transmission/reception ***/
276
277          S3TCNT |= TEN;                                          /* Enable transmission */
278          S3RCNT |= REN;                                          /* Enable reception */
279   }
280
281   /*""FUNC COMMENT""*****************************************************
282    * Function name: SIO0_Tr()
283    *----------------------------------------------------------------------
284    * Description  : Transmits data from SIO0
285    *----------------------------------------------------------------------
286    * Argument     : unsigned char *TrBuf  Pointer to transmit data buffer
287    *              : unsigned int TrNum    Number of transmit bytes
288    *----------------------------------------------------------------------
289    * Returns      : -
290    *----------------------------------------------------------------------
291    * Notes        : Do not always need to wait for end of transmission
292    *""FUNC COMMENT END""*************************************************/
293   void  SIO0_Tr( UCHAR *TrBuf, ULONG TrNum)
294   {
295          ULONG   i;
296
297          for( i = 0ul; i < TrNum; i++) {
298                 while(( S0TCNT & TBE) == 0u){                    /* Wait until transmit buffer
is empty */
299                        ;
300                 }
301                 S0TXB_L = *TrBuf++;                              /* Transfer return data */
302          }
303          while(( S0TCNT & TSTAT) != 0u){                         /* Wait for end of
transmission */
304                 ;
```

```
305          }
306   }
307
308   /*""FUNC COMMENT""*****************************************************
309    * Function name: SIO1_Tr()
310    *---------------------------------------------------------------------
311    * Description  : Transmits data from SIO1
312    *---------------------------------------------------------------------
313    * Argument     : unsigned char *TrBuf  Pointer to transmit data buffer
314    *              : unsigned int TrNum    Number of transmit bytes
315    *---------------------------------------------------------------------
316    * Returns      : -
317    *---------------------------------------------------------------------
318    * Notes        : Do not always need to wait for end of transmission
319    *""FUNC COMMENT END""*************************************************/
320   void  SIO1_Tr( UCHAR *TrBuf, ULONG TrNum)
321   {
322          ULONG   i;
323
324          for( i = 0ul; i < TrNum; i++) {
325                 while(( S1TCNT & TBE) == 0u){                  /* Wait until transmit buffer
is empty */
326                        ;
327                 }
328                 S1TXB_L = *TrBuf++;                            /* Transfer return data */
329          }
330          while(( S1TCNT & TSTAT) != 0u){                       /* Wait for end of
transmission */
331                 ;
332          }
333   }
334
335   /*""FUNC COMMENT""*****************************************************
336    * Function name: SIO2_Tr()
337    *---------------------------------------------------------------------
338    * Description  : Transmits data from SIO2
339    *---------------------------------------------------------------------
340    * Argument     : unsigned char *TrBuf  Pointer to transmit data buffer
341    *              : unsigned int TrNum    Number of transmit bytes
342    *---------------------------------------------------------------------
343    * Returns      : -
344    *---------------------------------------------------------------------
345    * Notes        : Do not always need to wait for end of transmission
346    *""FUNC COMMENT END""*************************************************/
347   void  SIO2_Tr( UCHAR *TrBuf, ULONG TrNum)
348   {
349          ULONG   i;
350
351          for( i = 0ul; i < TrNum; i++) {
352                 while(( S2TCNT & TBE) == 0u){                  /* Wait until transmit buffer
is empty */
353                        ;
354                 }
355                 S2TXB_L = *TrBuf++;                            /* Transfer return data */
356          }
357          while(( S2TCNT & TSTAT) != 0u){                       /* Wait for end of
transmission */
358                 ;
359          }
360   }
361
362   /*""FUNC COMMENT""*****************************************************
363    * Function name: SIO3_Tr()
364    *---------------------------------------------------------------------
365    * Description  : Transmits data from SIO3
366    *---------------------------------------------------------------------
367    * Argument     : unsigned char *TrBuf  Pointer to transmit data buffer
368    *              : unsigned int TrNum    Number of transmit bytes
369    *---------------------------------------------------------------------
370    * Returns      : -
371    *---------------------------------------------------------------------
372    * Notes        : Do not always need to wait for end of transmission
373    *""FUNC COMMENT END""*************************************************/
374   void  SIO3_Tr( UCHAR *TrBuf, ULONG TrNum)
375   {
376          ULONG   i;
377
```

```
378          for( i = 0ul; i < TrNum; i++) {
379                  while(( S3TCNT & TBE) == 0u){                   /* Wait until transmit buffer
is empty */
380                          ;
381                  }
382                  S3TXB_L = *TrBuf++;                             /* Transfer return data */
383          }
384          while(( S3TCNT & TSTAT) != 0u){                         /* Wait for end of
transmission */
385                  ;
386          }
387  }
388
389  /*""FUNC COMMENT""*****************************************************
390   * Function name: SIO0_RcvInt()
391   *-----------------------------------------------------------------------
392   * Description  : Reads out received data
393   *-----------------------------------------------------------------------
394   * Argument     : -
395   *-----------------------------------------------------------------------
396   * Returns      : -
397   *-----------------------------------------------------------------------
398   * Notes        : -
399   *""FUNC COMMENT END""*************************************************/
400  void    SIO0_RcvInt(void)
401  {
402          UCHAR   data;
403          UCHAR   status;
404
405          status = S0RCNT;                                        /* Read receive status */
406          data = S0RXB_L;                                         /* Read out received data */
407          status |= S0RCNT;                                       /* Handle overrun */
408
409          if(( status & ERS) == 0u) {                             /* Check receive error sum */
410                  RcvBuf0[ rx_cnt0++] = data;                     /* Store received data */
411
412          } else {                                                /* Disable reception */
413
414                  /* Error processing by user */
415
416                  S0RCNT &= ~REN;                                 /* Disable reception */
417                  S0RCNT |= REN;                                  /* Enable reception */
418          }
419  }
420
421  /*""FUNC COMMENT""*****************************************************
422   * Function name: SIO1_RcvInt()
423   *-----------------------------------------------------------------------
424   * Description  : Reads out received data
425   *-----------------------------------------------------------------------
426   * Argument     : -
427   *-----------------------------------------------------------------------
428   * Returns      : -
429   *-----------------------------------------------------------------------
430   * Notes        :
431   *""FUNC COMMENT END""*************************************************/
432  void    SIO1_RcvInt(void)
433  {
434          UCHAR   data;
435          UCHAR   status;
436
437          status = S1RCNT;                                        /* Read receive status */
438          data = S1RXB_L;                                         /* Read out received data */
439          status |= S1RCNT;                                       /* Handle overrun */
440
441          if(( status & ERS) == 0u) {                             /* Check receive error sum */
442                  RcvBuf1[ rx_cnt1++] = data;                     /* Store received data */
443
444          } else {                                                /* Disable reception */
445
446                  /* Error processing by user */
447
448                  S1RCNT &= ~REN;                                 /* Disable reception */
449                  S1RCNT |= REN;                                  /* Enable reception */
450          }
451  }
452
```

```
453  /*""FUNC COMMENT""*****************************************************
454   * Function name: SIO2_RcvInt()
455   *-----------------------------------------------------------------------
456   * Description  : Reads out received data
457   *-----------------------------------------------------------------------
458   * Argument     : -
459   *-----------------------------------------------------------------------
460   * Returns      : -
461   *-----------------------------------------------------------------------
462   * Notes        : -
463   *""FUNC COMMENT END""*************************************************/
464  void    SIO2_RcvInt(void)
465  {
466          UCHAR   data;
467          UCHAR   status;
468
469          status = S2RCNT;                                    /* Read receive status */
470          data = S2RXB_L;                                     /* Read out received data */
471          status |= S2RCNT;                                   /* Handle overrun */
472
473          if(( status & ERS) == 0u) {                         /* Check receive error sum */
474                  RcvBuf2[ rx_cnt2++] = data;                 /* Store received data */
475
476          } else {                                            /* Disable reception */
477
478                  /* Error processing by user */
479
480                  S2RCNT &= ~REN;                             /* Disable reception */
481                  S2RCNT |= REN;                              /* Enable reception */
482          }
483  }
484
485  /*""FUNC COMMENT""*****************************************************
486   * Function name: SIO3_RcvInt()
487   *-----------------------------------------------------------------------
488   * Description  : Reads out received data
489   *-----------------------------------------------------------------------
490   * Argument     : -
491   *-----------------------------------------------------------------------
492   * Returns      : -
493   *-----------------------------------------------------------------------
494   * Notes        : -
495   *""FUNC COMMENT END""*************************************************/
496  void    SIO3_RcvInt(void)
497  {
498          UCHAR   data;
499          UCHAR   status;
500
501          status = S3RCNT;                                    /* Read receive status */
502          data = S3RXB_L;                                     /* Read out received data */
503          status |= S3RCNT;                                   /* Handle overrun */
504
505          if(( status & ERS) == 0u) {                         /* Check receive error sum */
506                  RcvBuf3[ rx_cnt3++] = data;                 /* Store received data */
507
508          } else {                                            /* Disable reception */
509
510                  /* Error processing by user */
511
512                  S3RCNT &= ~REN;                             /* Disable reception */
513                  S3RCNT |= REN;                              /* Enable reception */
514          }
515  }
516
517  /*""FUNC COMMENT""*****************************************************
518   * Function name: IntHand_SIO23()
519   *-----------------------------------------------------------------------
520   * Description  : Processing by SIO2,3 transmit/receive interrupt handlers
521   *-----------------------------------------------------------------------
522   * Argument     : -
523   *-----------------------------------------------------------------------
524   * Returns      : -
525   *-----------------------------------------------------------------------
526   * Notes        : Transmit interrupt is not considering
527   *""FUNC COMMENT END""*************************************************/
528  void    IntHand_SIO23(void)
529  {
```

```
530          if(( SI23STAT & IRQR3) != 0u) {                        /* Judge interrupt request */
531                  SI23STAT = (~IRQR3) & 0x0fu;                   /* Clear interrupt request
status */
532                  SIO3_RcvInt();                                 /* Process SIO3 receive
interrupt */
533          }
534
535          if(( SI23STAT & IRQR2) != 0u) {                        /* Judge interrupt request */
536                  SI23STAT = (~IRQR2) & 0x0fu;                   /* Clear interrupt request
status */
537                  SIO2_RcvInt();                                 /* Process SIO2 receive
interrupt */
538          }
539  }
540
```

4.4.3 startup.ms (partially omitted)

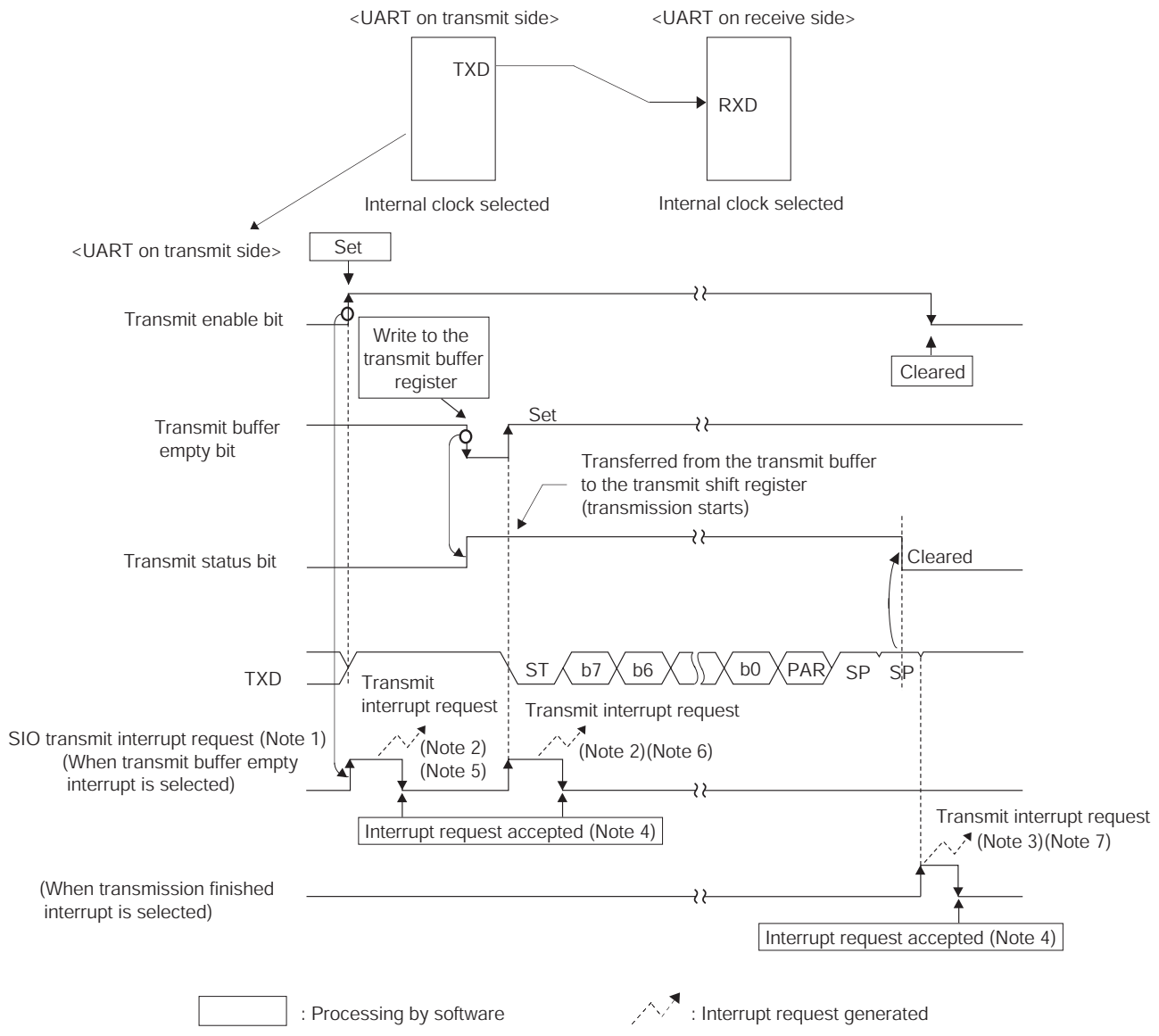
(Omitted)

```
 69  ;**************************************
 70  ; ICU Vector Table
 71  ;**************************************
 72  ;
 73          .SECTION        ICUVECT, DATA, ALIGN=4
 74  ;
 75          .IMPORT         $SIO1_RcvInt
 76  ;
 77  vectbl:
 78          .DATA.W         EIT_reset               ; H'0000 0094   MJT Input Interrupt 4:TIN3-
TIN6
 79          .DATA.W         EIT_reset               ; H'0000 0098   MJT Input Interrupt 3:TIN20-
TIN23
 80          .DATA.W         EIT_reset               ; H'0000 009C   MJT Input Interrupt 2:TIN12-
TIN19
 81          .DATA.W         EIT_reset               ; H'0000 00A0   MJT Input Interrupt 1:TIN0-
TIN2
 82          .DATA.W         EIT_reset               ; H'0000 00A4   MJT Input Interrupt 0:TIN7-
TIN11
 83          .DATA.W         EIT_reset               ; H'0000 00A8   MJT Output Interrupt
7:TMS0,TMS1
 84          .DATA.W         EIT_reset               ; H'0000 00AC   MJT Output Interrupt
6:TOP8,TOP9
 85          .DATA.W         EIT_reset               ; H'0000 00B0   MJT Output Interrupt 5:TOP10
 86          .DATA.W         EIT_reset               ; H'0000 00B4   MJT Output Interrupt 4:TIO4-
TIO7
 87          .DATA.W         EIT_reset               ; H'0000 00B8   MJT Output Interrupt
3:TIO8,TIO9
 88          .DATA.W         EIT_reset               ; H'0000 00BC   MJT Output Interrupt 2:TOP0-
TOP5
 89          .DATA.W         EIT_reset               ; H'0000 00C0   MJT Output Interrupt
1:TOP6,TOP7
 90          .DATA.W         EIT_reset               ; H'0000 00C4   MJT Output Interrupt 0:TIO0-
TIO3
 91          .DATA.W         EIT_reset               ; H'0000 00C8   DMAC0-4 Interrupt:DMA0-DMA4
 92          .DATA.W         $SIO1_RcvInt            ; H'0000 00CC   SIO1 Receive Interrupt
 93          .DATA.W         EIT_reset               ; H'0000 00D0   SIO1 Transmit Interrupt
 94          .DATA.W         EIT_reset               ; H'0000 00D4   SIO0 Receive Interrupt
 95          .DATA.W         EIT_reset               ; H'0000 00D8   SIO0 Transmit Interrupt
 96          .DATA.W         EIT_reset               ; H'0000 00DC   A-D0 Conversion Interrupt
 97          .DATA.W         EIT_reset               ; H'0000 00E0   TID0 Output Interrupt
 98          .DATA.W         EIT_reset               ; H'0000 00E4   TOD0 Output Interrupt
 99          .DATA.W         EIT_reset               ; H'0000 00E8   DMAC5-9 Interrupt:DMA5-DMA9
100          .DATA.W         EIT_reset               ; H'0000 00EC   SIO2,3 Transmit/Receive
Interrupt
101          .DATA.W         EIT_reset               ; H'0000 00F0   RTD Interrupt
102          .DATA.W         EIT_reset               ; H'0000 00F4   TID1 Output Interrupt
103          .DATA.W         EIT_reset               ; H'0000 00F8   TOD1,TOM0 Output Interrupt
104          .DATA.W         EIT_reset               ; H'0000 00FC   SIO4,5 Transmit/Receive
Interrupt
105          .DATA.W         EIT_reset               ; H'0000 0100   A-D1 Conversion Interrupt
106          .DATA.W         EIT_reset               ; H'0000 0104   TID2 Output Interrupt
107          .DATA.W         EIT_reset               ; H'0000 0108   TML1 Input Interrupt
108          .DATA.W         EIT_reset               ; H'0000 010C   CAN0 Transmit/Receive & Error
Interrupt
109          .DATA.W         EIT_reset               ; H'0000 0110   CAN1 Transmit/Receive & Error
Interrupt
110  ;
```

(The remainder of the program has been omitted)

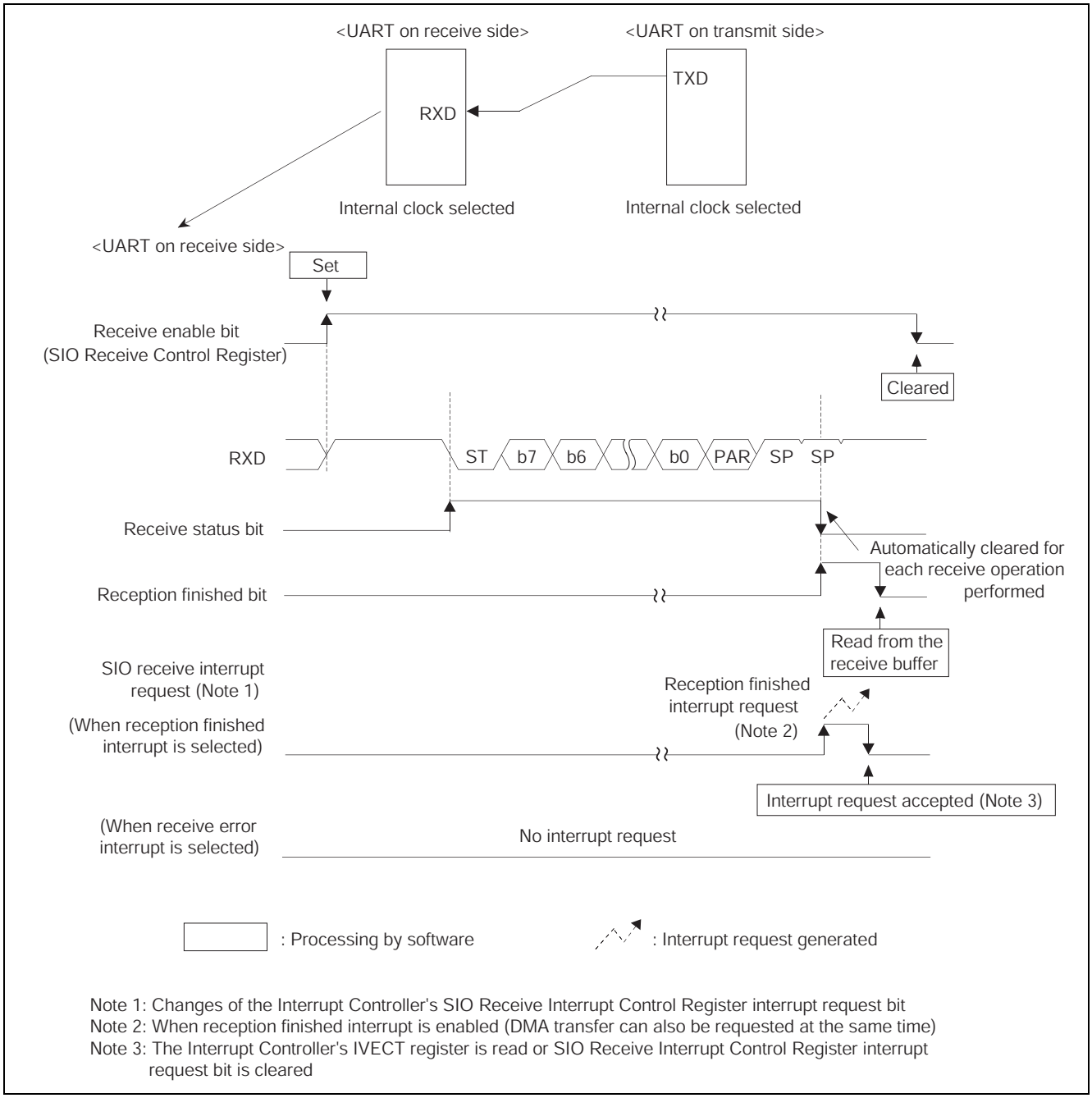
4.5 UART Operation Timing

The UART mode transmission and reception timing diagrams are shown below.



Note 1: Changes of the Interrupt Controller's SIO Transmit Interrupt Control Register interrupt request bit
Note 2: When transmit buffer empty interrupt is enabled (DMA transfer can also be requested at the same time)
Note 3: When transmission finished interrupt is enabled
Note 4: The Interrupt Controller's IVECT register is read or SIO Transmit Interrupt Control Register interrupt request bit is cleared.
Note 5: A transmit interrupt request is generated when transmission is enabled.
Note 6: Be aware that even after transmit data is written to the transmit buffer, a transmit interrupt request is generated when the data is transferred from the transmit buffer to the transmit shift register and the transmit buffer is thereby emptied.
Note 7: A transmission finished interrupt request is generated when data in the transmit shift register has all been transmitted.

Figure 4.5.1 UART Mode Transmission Timing Diagram



Note 1: Changes of the Interrupt Controller's SIO Receive Interrupt Control Register interrupt request bit
Note 2: When reception finished interrupt is enabled (DMA transfer can also be requested at the same time)
Note 3: The Interrupt Controller's IVECT register is read or SIO Receive Interrupt Control Register interrupt request bit is cleared

Figure 4.5.2 UART Mode Receive Timing Diagram

5. Reference Documents

- 32176 Group User's Manual (Rev.1.01)
- M32R Family Software Manual (Rev.1.20)
- M3T-CC32R V.4.30 User's Manual (Compiler)
- M3T-CC32R V.4.30 User's Manual (Assembler)

(Please get the latest one from Renesas Technology Corp. website.)

6. Website and Support Center

- Renesas Technology Corp. website.

  http://www.renesas.com/

- Inquires for all Renesas products and technical inquiries for the M32R Family products, please contact:

  Customer Support Center: csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jan.12.06	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corporation 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 Corporation 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 Corporation or a third party.
2. Renesas Technology Corporation 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 Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation by various means, including the Renesas Technology Corporation 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 Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation 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 Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation for further details on these materials or the products contained therein.