To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

H8/3687

Transmission/Reception with Terminal Software (H8/3687)

Introduction

The H8/3687 group are single-chip microcomputers based on the high-speed H8/300H CPU, and integrate all the peripheral functions necessary for system configuration. The H8/300H CPU employs an instruction set which is compatible with the H8/300 CPU.

The H8/3687 group incorporates, as peripheral functions necessary for system configuration, a timer, I^2C bus interface, serial communication interface, and 10-bit A/D converter. These devices can be utilized as embedded microcomputers in sophisticated control systems.

These H8/300 H Series -H8/3687- Application Notes consist of a "Basic Edition" which describes operation examples when using the onboard peripheral functions of the H8/3687 group in isolation; they should prove useful for software and hardware design by the customer.

The operation of the programs and circuits described in these Application Notes has been verified, but in actual applications, the customer should always confirm correct operation prior to actual use.

Target Device

H8/3687

Contents

1. Overview

The SCI3 interface of the H8/3687 is used to perform transmission and reception processing with the terminal software of a personal computer. By using this sample program, debugging messages can be output to the terminal software, and commands can easily be received.

2. Configuration

Figure 2.1 is a diagram showing connection with the terminal software.

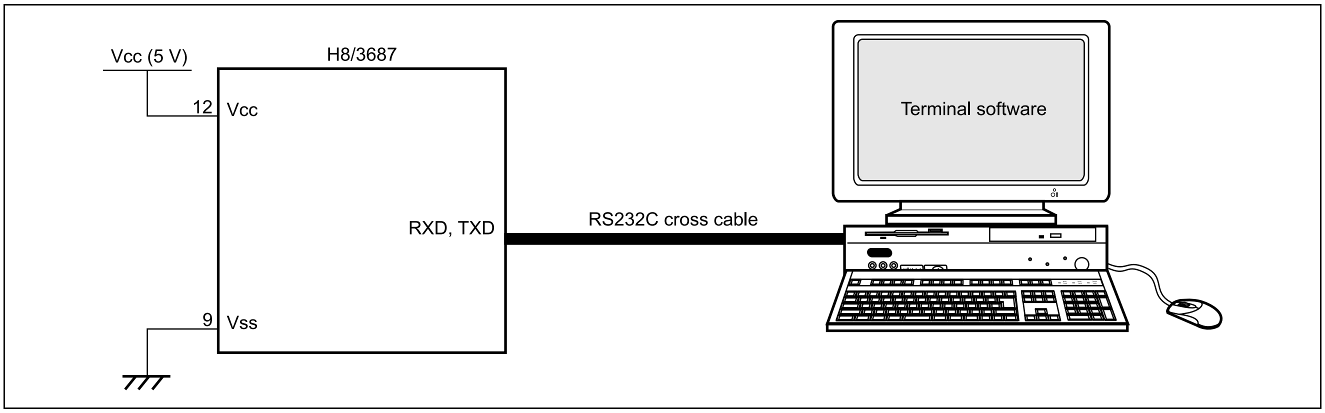
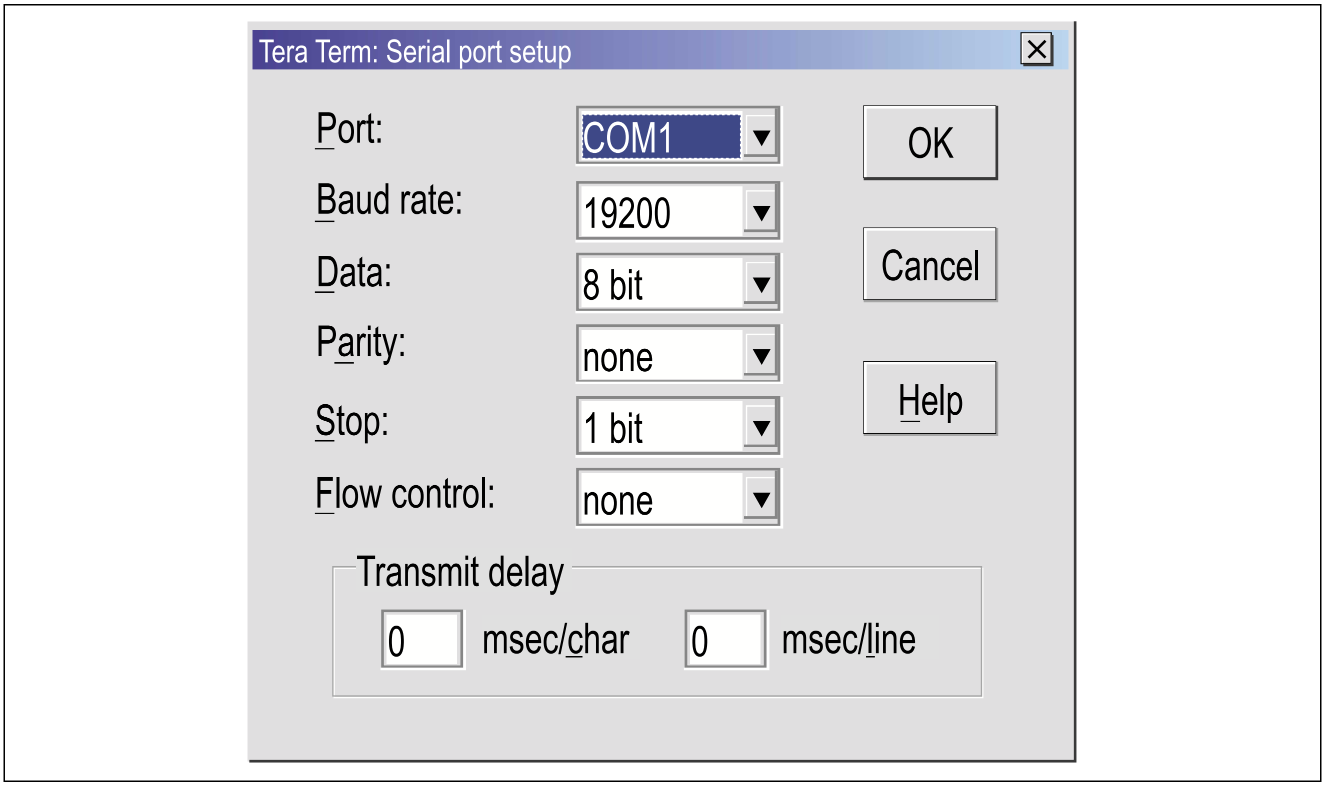


Figure 2.1 Connection with the terminal software, I^2C EEPROM

Specifications:

- H8/3687 operating frequency: 16 MHz
- Terminal software used: Tera Term
- Communication specifications



3. Sample Program

3.1 Functions

1. Messages are output to the terminal software.
2. Commands are received from the terminal software, and reading and writing of registers within the H8 microcomputer are performed.

3.2 Embedding the sample programs

1. Sample program 10-A
   Incorporate #define directives.
2. Sample program 10-B
   Incorporate common variable declarations.
3. Sample program 10-C
   Incorporate prototype declarations.
4. Sample program 10-D
   Incorporate prototype declarations.
5. Sample program 10-B
   Incorporate prototype declarations.
   — 4.1 Add the h8_sci3 reset vector.
   — 4.2 Add the SCI3 initial setting processing.
   — 4.3 Add the common routine.
   — 4.4 The SCI3 interrupt processing is added.

3.3 Modifications to sample programs

Without modifications to the sample program, the system may not run. Modifications must be made according to the customer's program and system environment.

1. By using a file with definitions of IO register structures which can be obtained free of charge from the following Renesas web site,http://www.renesas.com/eng/products/mpumcu/tool/crosstool/iodef/index.html
   the sample program can be used without further changes. When creating definitions independently, the customer should modify the IO register structures used in the sample program as appropriate.
2. In the same program, the timer Z is started every 10 ms and times out in order to monitor the state of the SCI3 interface. Timer processing may be modified according to the customer's needs. Of course the program can be used without changes. When the timer processing of the sample program is used as is, the following modifications should be made.
   A. Sample program 10-E
      - 5.1 The TimerZ reset vector should be added.
      - com_timer should be added as a common variable.
      - The TimerZ initial setting processing should be added.
      - (The GRA setting should be changed according to the operating frequency of the microcomputer being used, so that the TimerZ interrupt occurs in 10 ms. For setting values, refer to the H8/3687 Hardware Manual; for the location of the setting to be changed, refer to the program notes in the sample program.)
      - The TimerZ interrupt processing should be added.

3. The transfer rate BRR of the SCI3 interface should be set according to the speed of the line used and the microcomputer operating frequency. For settings, refer to the H8/3687 Hardware Manual; refer to the program notes in the sample program for places to be changed. In this sample program, the line speed is set to 19200 bps.

3.4 Method of use

1. Messages are output to the terminal software.

```
void com_cnsl_msg(char *fmt)
```

Argument	Description
*fmt	Directly states the message to be output to the console. For details of the format, refer to sprintf in the stdio.h library. The number of characters in the message must be 80 or less.

Return value	None
0	Normal termination
0x104	Data transmission error

Example:

```
MSC_VER = 0x02
MSC_MAJOR_REV = 0x00
MSC_MINOR_REV = 0x01
ret = com_cnsl_msg("Ver-Rev-Mrev = %02hX-%02hX-%02hX\n\r", MSC_VER, MSC_MAJOR_REV,
                   MSC_MINOR_REV);
if(ret !=0){/* data transmission error */}
  else{/* data transmission normal termination */}
[Console output result]
Ver-Rev-Mrev = 02-00-01
```

2. Commands are received from the terminal software, and reading and writing of registers in the H8 microcomputer are performed.

- Read of registers in the H8 microcomputer (read all registers)

```
[Console input]  hr
[Output example]
Key in >hr
               0        4        8       C
addr:F700 = 238888E0 E1F8FFFF FFFFFFFF FFFFFFFF
addr:F710 = 008888C0 E0F8FFFF FFFFFFFF FFFFFFFF
addr:F720 = FD0E8880 FF00FFFF FFFFFFFF FFFFFFFF
addr:F730 = 70FCFFFF FFFFFFFF FFFFFFFF FFFFFFFF
                        :
                        :
                        :
```

- Read of an H8 microcomputer register (register specified)

```
[Console input]  hrΔ(address: 2 bytes)       (Δ: space)
[Output example]
Key in >hr f700
   addr:F700 = 23
```

- Write of an H8 microcomputer register (register specified)

```
[Console input]  hwΔ(address: 2 bytes)  (data: 1 byte)     (Δ: space)
[Output example]
Key in >hw f700 34
   (H8 REG WRITE F700 <- 34)
   addr:F700 = 34
```

3.5 Description of operation

Here program lists for various processing are explained, and matters requiring attention when the program is employed by a customer are described.

1. Messages are output to the terminal software.
   Refer to the program note for com_cnsl_msg in the program source list.
2. Commands are received from the terminal software, and reading and writing of registers within the H8 microcomputer are performed.
   Refer to the program note for h8_sci3 (SCI interrupt processing) in the program source list.

3.6 List of registers used

The internal registers of the H8 microcomputer used in the sample program are listed below. For detailed information, refer to the H8/3687 Group Hardware Manual.

1. SCI3-related registers

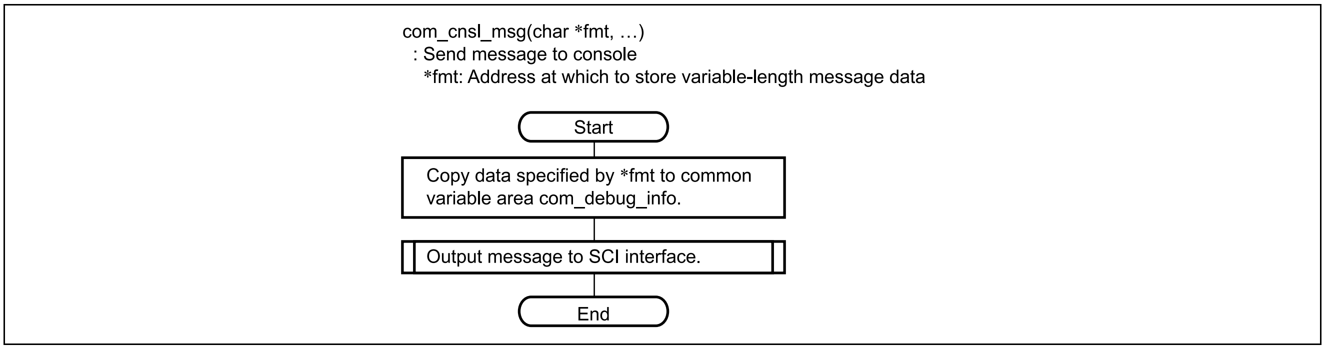
Name	Summary
Receive data register (RDR)	8-bit register to store received data
Transmit data register (TDR)	8-bit register to store transmission data
Serial mode register (SMR)	Register to select the serial data communication format and internal baud rate generator clock source
Serial control register 3 (SCR3)	Register to control transmission/reception operation and interrupts, and to select the transmission/reception clock source
Serial status register (SSR)	SCI3 status flags and transmission/reception multiprocessor bits
Bit rate register (BRR)	8-bit register to set the bit rate

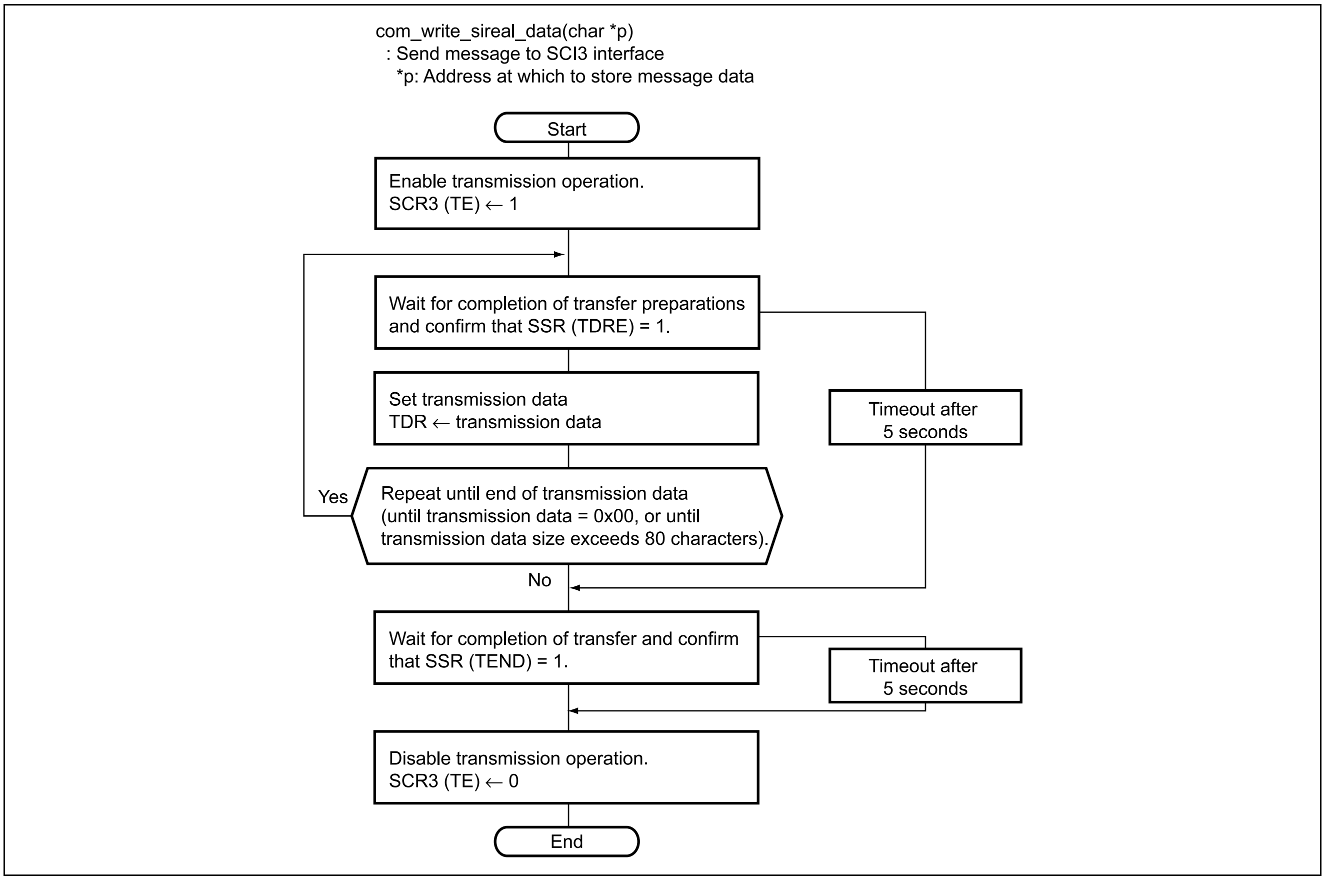
2. Timer Z-related registers
   The timer Z has various functions, but in this sample program is made to generate an interrupt every 10 ms using the GRA register compare-match function.

Name	Summary
Timer start register (TSTR)	Selects TCNT operation/termination
Timer mode register (TMDR)	Sets buffer operation and selects synchronous operation
Timer PWM mode register (TPMR)	Sets a pin to PWM mode. Not used in this sample program.
Timer function control register (TFCR)	Sets each operating mode and selects output level. Not used in this sample program.
Timer output master enable register (TOER)	Enables/disables output from channels 0 and 1
Timer output control register (TOCR)	Sets initial output until a first compare match occurs
Timer counter (TCNT)	16-bit readable/writable register which performs counting operation by an input clock
General registers A, B, C, D (GRA, GRB, GRC, GRD)	These are 16-bit readable/writable registers. There are eight GRs: four GRs for each channel. This 16-bit readable/writable register can be used as both an output-compare register and as an input-capture register through switching by TIORA and TIORC.
Timer control register (TCR)	Selects the TCNT counter clock, selects the counter clear conditions and edge when an external clock is selected
Timer I/O control register (TIORA)	Selects functions of GRA and GRB as either the output compare register or the input capture register. Not used in this sample program.
Timer status register (TSR)	Indicates occurrence of TCNT overflow/underflow, and the occurrence of compare match/input capture of GRA, GRB, GRC, GRD
Timer interrupt enable register (TIER)	Enables/disables an overflow interrupt request and GR compare match/input capture interrupt requests
PWM mode output level control register (POCR)	Controls active level in PWM mode. Not used in this sample program.

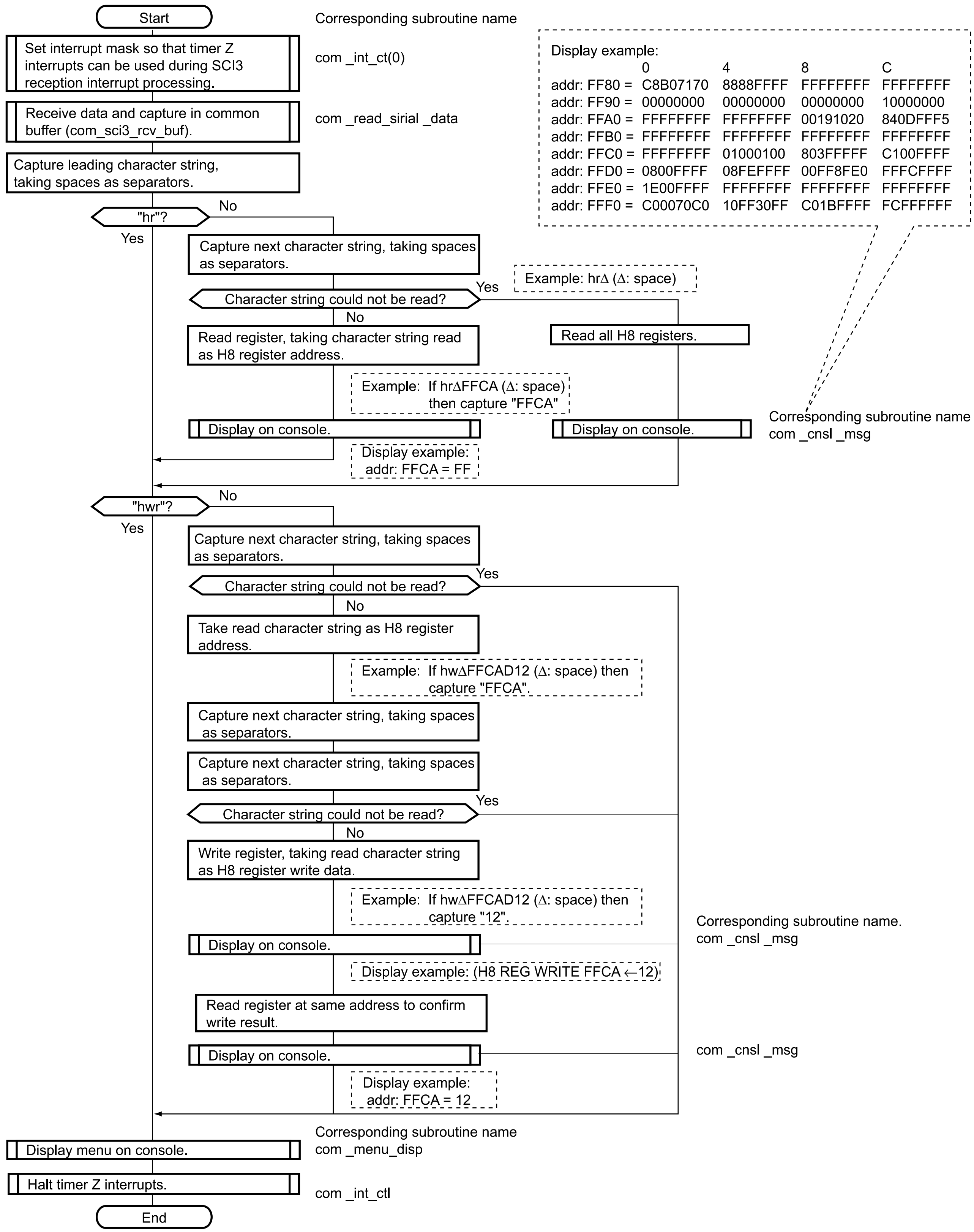
3.7 Flowcharts

The flow of program processing is indicated below.

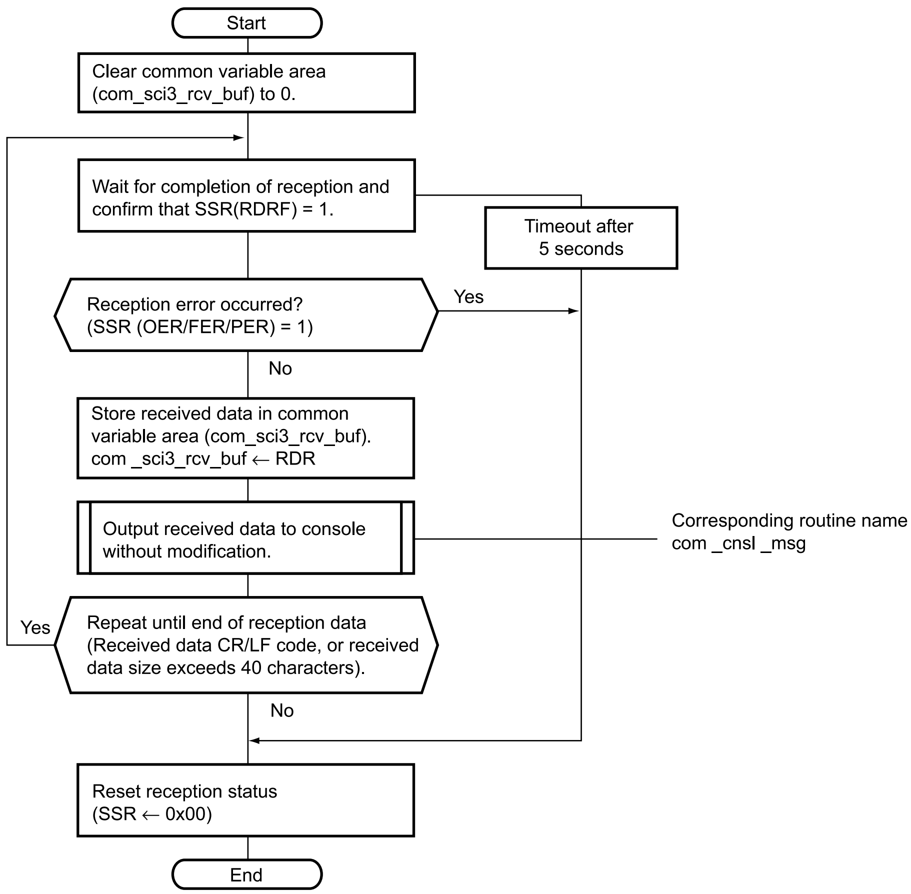


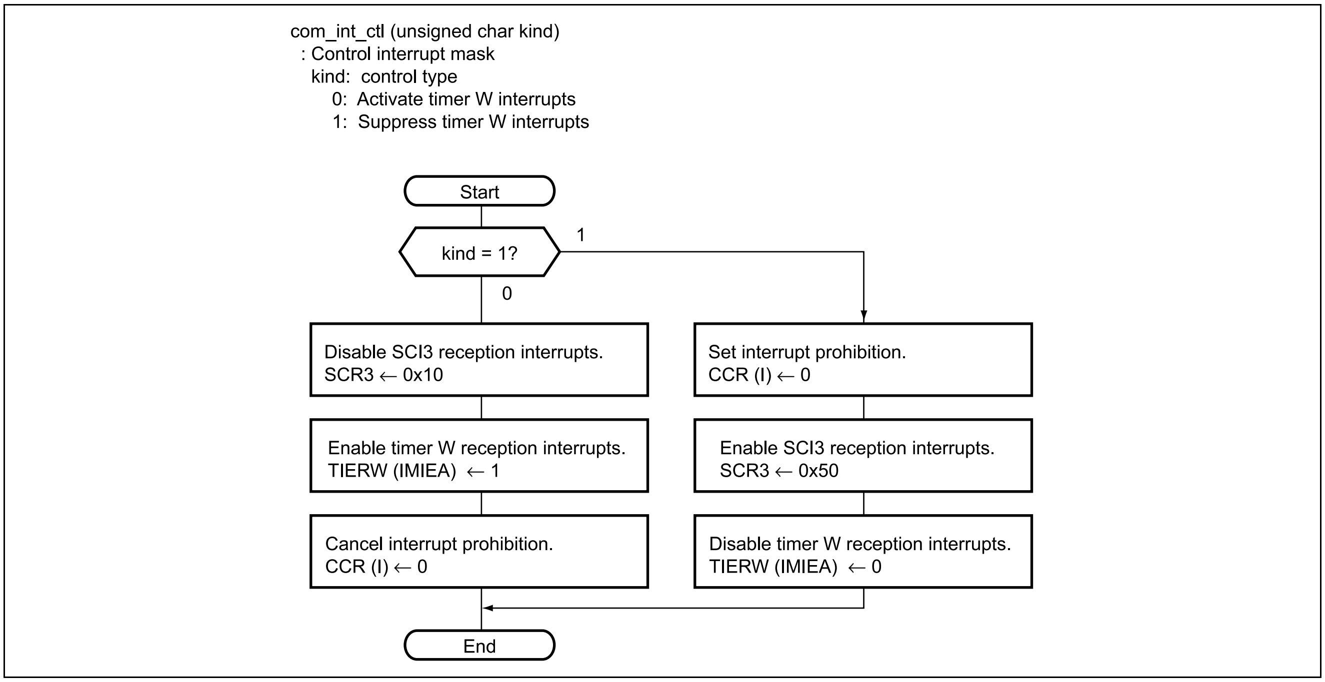


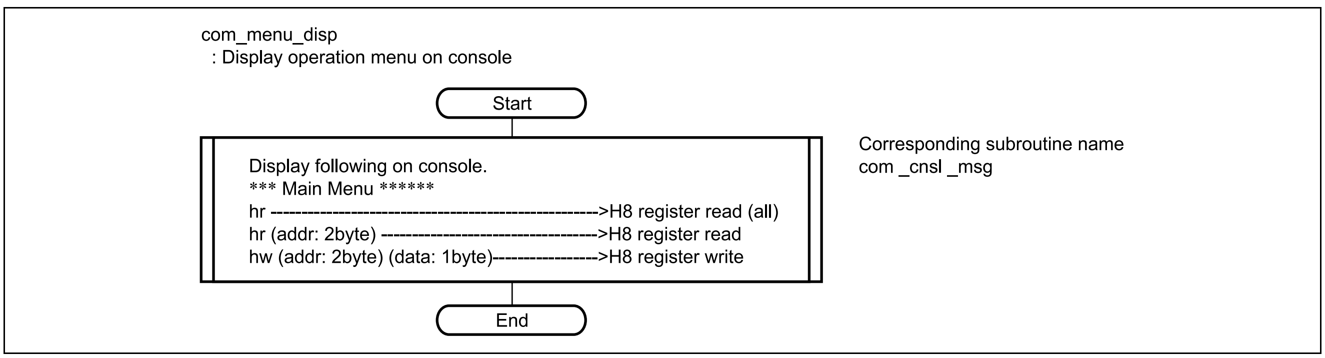
h8_sci3
: SCI3 reception interrupt processing



com_read_sirial_dat
: Receive message from SCI3 interface
Up to 40 characters of reception data is received
in the common variable area (com_sci3_rcv_buf)




com_int_ctl (unsigned char kind)
: Control interrupt mask
kind: control type
0: Activate timer W interrupts
1: Suppress timer W interrupts
Start
kind = 1?
1
0
Disable SCI3 reception interrupts.
SCR3 ← 0x10
Enable timer W reception interrupts.
TIERW (IMIEA) ← 1
Cancel interrupt prohibition.
CCR (I) ← 0
Set interrupt prohibition.
CCR (I) ← 0
Enable SCI3 reception interrupts.
SCR3 ← 0x50
Disable timer W reception interrupts.
TIERW (IMIEA) ← 0
End


com_menu_disp
: Display operation menu on console
Start
Display following on console.
*** Main Menu ******
hr -->H8 register read (all)
hr (addr: 2byte) ---------------------------------->H8 register read
hw (addr: 2byte) (data: 1byte)---------------->H8 register write
End
Corresponding subroutine name
com _cnsl _msg

3.8 Program Listing

```
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 1. Sample Program 10-A  define definition ---------------------------------------------------------------------------------  */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/******************************************************************************************************************************/
/*   RS232C access error           code                                                                                       */
/******************************************************************************************************************************/
#define   SCI3_RCV_OVERRUN         0x0101
#define   SCI3_RCV_FRAME_ERR       0x0102
#define   SCI3_RCV_PARITY_ERR      0x0103
#define   SCI3_SND_ENBL_TOUT       0x0104
#define   SCI3_RCV_FULLL_TOUT      0x0105
#define   SCI3_INVALID_PARM        0x0106

/******************************************************************************************************************************/
/*   Value definition                                                                                                         */
/******************************************************************************************************************************/
#define   DBG_INFO_SIZE            80



/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 2. Sample program 10-B  Prototype declaration ------------------------------------------------------------------------------ */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* Console processing                                                                                                         */
void com_menu_disp(void) ;
unsigned int com_read_sireal_data ();
unsigned int com_write_sireal_data (char *p);
void com_int_ctl (unsigned char kind) ;
void com_cnsl_msg(char *fmt, ...)  ;
```

```
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 3. Sample program 10 -c Common variable declaration ----------------------------------------------------------------------  */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */

    /*************************************************************************************************************************/
    /*   common variable                                                                                                     */
    /*************************************************************************************************************************/
    #ifndef        _DEFINE_COMMON_TABLE
         extern
    #endif                                                                  /* _DEFINE_COMMON_TABLE                          */
         struct    {
              unsigned char cw_dipsw ;                                      /* cw_dipsw                                      */
              unsigned char batt_off ;                                      /* battery off                                   */
              unsigned char dummy ;                                         /* dummy                                         */
         }com_global;

    /*************************************************************************************************************************/
    /*   Console interface                                                                                                   */
    /*************************************************************************************************************************/
    #ifndef        _DEFINE_COMMON_TABLE
         extern
    #endif                                                                  /* _DEFINE_COMMON_TABLE                          */
         char com_sci3_rcv_buf[40] ;                                        /* SCI3 buffer f                                 */

    #ifndef        _DEFINE_COMMON_TABLE
         extern
    #endif                                                                  /* _DEFINE_COMMON_TABLE                          */
         char com_debug_info[DBG_INFO_SIZE] ;                               /* Debugging message information                 */
```

```
/* ------------------------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------- */
/* 4. Sample program 10-D Common processing source codes --------------------------------------------------------------------  */
/* ------------------------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------- */

/* ------------------------------------------------------------------------------------------------------------------------- */
/* 4.1 Addition of reset vectors ---------------------------------------------------------------------------------------------  */
/* ------------------------------------------------------------------------------------------------------------------------- */

/*   Set the jump destination to h8_sci3.                                                                                   */

/* ------------------------------------------------------------------------------------------------------------------------- */
/* 4.2 Initialization processing   ------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------- */

     /* ######################################################################################################################## */
     /* ######################################################################################################################## */
     /*                                                                                                                      */
     /*         Sets the serial interface (SCI3)                                                                             */
     /*                                                                                                                      */
     /* ######################################################################################################################## */
     /* ######################################################################################################################## */
     /*************************************************************************************************************************/
     /*   PMR1      Specifies the IO port 1 usage                                                                            */
     /*   TXD2      = 0  Used as the TXD_2 pin (SCI3_2 transmit pin)                                                         */
     /*   TXD       = 1  Used as the TXD pin (SCI3 transmit pin)                                                             */
     /*************************************************************************************************************************/
     IO.PMR1.BIT.TXD     = 1 ;
          /* ##(program note)################################################################################################ */
          /* ## Defines the P22/TXD port to be used as the TXD pin.                                                        ## */
          /* ################################################################################################################ */

     /*************************************************************************************************************************/
     /*   SCR3      Specifies the SCI3 control registers.                                                                    */
     /*        TIE       = 0  Disables transmit end interrupts.                                                              */
     /*        RIE       = 0  Receive interrupts (disabled at this time; set before sleep)                                   */
     /*        TE        = 0  Transmit enable (set when it is actually used; here, set to 0)                                 */
     /*        RE        = 0  Receive enable (set when it is actually used; here, set to 0)                                  */
     /*        MPIE      = 0  (Not used)                                                                                     */
     /*        TEIR      = 0  (Not used)                                                                                     */
     /*        CKE1:0    = 00 Specifies the internal baud rate generator as the clock source.                                */
     /*************************************************************************************************************************/
     SCI3.SCR3.BYTE = 0x00 ;

     /*************************************************************************************************************************/
     /*   SMR            Sets SCI3 mode                                                                                      */
     /*        COM       = 0  Operates in asynchronous mode                                                                  */
     /*        CHR       = 0  Data length: 8 bits                                                                            */
     /*        PE        = 0  No parity bits                                                                                 */
     /*        PM        = 0  (Not used)                                                                                     */
     /*        STOP      = 0  1 stop bit                                                                                     */
     /*        MP        = 0  (Not used)                                                                                     */
     /*        CKS1:0    = 00 Internal baud rate generator clock source: ϕ clock                                             */
     /*************************************************************************************************************************/
     SCI3.SMR.BYTE  = 0x00 ;

     /*************************************************************************************************************************/
     /*   BRR            Specifies 19200 bps                                                                                 */
     /*************************************************************************************************************************/
     SCI3.BRR = 25;
          /* ##(program note)################################################################################################ */
          /* ## The value set for BRR should be changed based on the necessary transfer rate.                              ## */
          /* ## For detailed information, please refer to the H8/3687 Hardware Manual.                                     ## */
          /* ################################################################################################################ */
```

```
/* ----------------------------------------------------------------------------------------------------------------------- */
/* 4.3 Common processing      ------------------------------------------------------------------------------------------    */
/* ----------------------------------------------------------------------------------------------------------------------- */

/*************************************************************************************************************************/
/*************************************************************************************************************************/
/*************************************************************************************************************************/
/*                                                                                                                       */
/*                                 Serial interface                                                                      */
/*                                                                                                                       */
/*************************************************************************************************************************/
/*************************************************************************************************************************/
/*************************************************************************************************************************/
/*************************************************************************************************************************/
/*   1. Module name : com_cnsl_msg                                                                                       */
/*   2. Function overview : Displays a message on the debugging console.                                                 */
/*************************************************************************************************************************/
void com_cnsl_msg(char *fmt, ...) {

     va_list   argp ;
     va_start(argp,fmt) ;
     vsprintf(&(com_debug_info[0]),fmt,argp) ;
          /* ##(program note)################################################################################################ */
          // ##  Copies the variable length character string specified by fmt to com_debug_info.                           ## */
          // ##  The character string that can be specified by fmt is 80 characters or less because com_debug_info defines   ## */
          // ##  an area of 80 characters.                                                                                  ## */
          // ##  If com_debug_info is re-defined, a character string longer than 80 characters can be displayed.            ## */
          /* ################################################################################################################ */

     com_write_sireal_data(com_debug_info) ;
          /* ##(program note)################################################################################################ */
          /* ##  Outputs the contents of com_debug_info to the console.                                                     ## */
          /* ################################################################################################################ */

     va_end(argp) ;
}

/*************************************************************************************************************************/
/*   1. Module name : com_menu_disp                                                                                      */
/*   2. Function overview: Displays the main menu of the console command                                                 */
/*************************************************************************************************************************/
void com_menu_disp (void)
{

     com_cnsl_msg("\n\r") ; // A carriage return
     com_cnsl_msg("*** Main Menu ***** \n\r") ;
     com_cnsl_msg(" hr ---------------------------> H8 register read(all)\n\r") ;
     com_cnsl_msg(" hr (addr:2byte) --------------> H8 register read \n\r") ;
     com_cnsl_msg(" hw (addr:2byte) (data:1byte) -> H8 register write \n\r") ;
     com_cnsl_msg("\n\r") ; // A carriage return
     com_cnsl_msg("Key in >") ;

}
```

```
/*****************************************************************************************************************************/
/*    1. Module name : _write_sireal_data                                                                                    */
/*    2. Function overview: Writes data to serial if                                                                         */
/*    return code                                                                                                            */
/*    NORMAL_END          :    Normal end                                                                                    */
/*    SCI3_SND_ENBL_TOUT  :    Transmit processing is not enabled.                                                           */
/*****************************************************************************************************************************/
unsigned int com_write_sireal_data ( char *p )
{
     unsigned char   send_data;
     int   i;
     unsigned int   ret;

     i = 0 ;
     ret = NORMAL_END ;

     SCI3.SCR3.BIT.TE = 1     ;                                   /* Enables the transmit operation                */
     do {
          if((*p == 0x00) || (i>DBG_INFO_SIZE)) {                 /* Detects the end of the transmit data          */
               break ;
          }

          com_timer.wait_100ms_sci3 = 50 ;
          while(SCI3.SSR.BIT.TDRE == 0){                          /* Waits until the next data transfer is enabled.   */
               if (com_timer.wait_100ms_sci3 == 0){               /* If data cannot be transferred for 5 seconds,     */
                                                                  /* it can be escaped.                               */
                    ret = SCI3_SND_ENBL_TOUT ;                    /* Performs no operation even if an error occurs.   */
                    goto exit ;
               }
          }
          SCI3.TDR = *p++  ;                                      /* Data transmission ->                          */
                                                                  /* This operation resets the SCI3.SSR.BIT.TDRE.  */
          i++ ;
     } while(1);

exit :
     com_timer.wait_100ms_sci3 = 50 ;
     while(SCI3.SSR.BIT.TEND == 0){                               /* Waits until the transmission has completed.   */
          if (com_timer.wait_100ms_sci3 == 0){                    /* If data cannot be transferred for 5 seconds,  */
                                                                  /* it can be escaped.                            */
               ret = SCI3_SND_ENBL_TOUT ;                         /* Performs no operation even if an error occurs. */
          }
     }

     SCI3.SCR3.BIT.TE =  0    ;                                   /* Disables the transmit operation.              */

     return (ret) ;

}
```

```
/************************************************************************************************************************/
/*   1. Module name : com_read_sireal_data                                                                              */
/*   2. Function overview: Receives data from serial if.                                                               */
/*   return code                                                                                                        */
/*   NORMAL_END           :    Normal end                                                                               */
/*   SCI3_RCV_OVERRUN     :    over run                                                                                 */
/*   SCI3_RCV_FRAME_ERR   :    frame err                                                                                */
/*   SCI3_RCV_PARITY_ERR :    parity error                                                                              */
/************************************************************************************************************************/
unsigned int com_read_sireal_data (void)
{
     unsigned char   rcv_data;
     int i ;

     unsigned int   ret;

     ret = NORMAL_END ;

     for (i=0; i<40; i++){                                        /* Clears the receive buffer.                    */
          com_sci3_rcv_buf[0] = 0;
     }

     i = 0 ;
     do {
          com_timer.wait_100ms_sci3 = 300 ;
          while(SCI3.SSR.BIT.RDRF == 0){                          /* Waits until receive data is received (max 30 sec)*/
               if (com_timer.wait_100ms_sci3 == 0){               /* If data cannot be transferred for 1 second,      */
                                                                  /* it can be escaped.                               */
                    ret =SCI3_RCV_FULLL_TOUT ;                    /* Performs no operation even if an error occurs.   */
                    goto exit ;
               }

               if ((SCI3.SSR.BYTE & 0x38) !=0) {                  /* An error occurs.                               */
                    if (SCI3.SSR.BIT.OER == 1){                   /* over run                                       */
                         SCI3.SSR.BIT.OER = 0;                    /* Resets the source.                             */
                         SCI3.SSR.BIT.RDRF = 0;                   /* Resets the receive data full bit.              */
                         ret = SCI3_RCV_OVERRUN ;
                         goto exit ;
                    }
                    if (SCI3.SSR.BIT.FER == 1){                   /* framing err                                    */
                         SCI3.SSR.BIT.FER = 0;                    /* Resets the source.                             */
                         SCI3.SSR.BIT.RDRF = 0;                   /* Resets the receive data full bit.              */
                         ret = SCI3_RCV_FRAME_ERR ;
                         goto exit ;
                    }
                    if (SCI3.SSR.BIT.PER == 1){                   /* parity err                                     */
                         SCI3.SSR.BIT.PER = 0;                    /* Resets the source.                             */
                         SCI3.SSR.BIT.RDRF = 0;                   /* Resets the receive data full bit.              */
                         ret = SCI3_RCV_PARITY_ERR ;
                         goto exit ;
                    }
               }
          }

          rcv_data = SCI3.RDR ;                                   /* Data reception ->                              */
                                                                  /* This operation resets the SCI3.SSR.BIT.RDRF.     */

                                                                  /* Returns the received data directly to the console*/
                                                                  /* → To display input data on the screen.           */
          com_cnsl_msg("%c",rcv_data) ;
```

```
        if (i<40){                                          /* Reads data but does not store data          */
                                                            /* if the buffer overflows.                    */
            com_sci3_rcv_buf[i] = rcv_data ;                /* Stores data in receive buffer.              */
        }

        i ++ ;
    } while((rcv_data != 0x0a) && (rcv_data != 0x0d));      /* Ends the reception if a carriage return is received. */

exit :
    SCI3.SSR.BYTE = 00 ;                                    /* Resets the receive data full bit.           */

    return (ret) ;

}

/**************************************************************************************************************/
/*   1. Module name : com_int_ctl                                                                            */
/*   2. Function overview: Clears set_imask_ccr to 0 to enable only the TimerZ interrupts                    */
/**************************************************************************************************************/
void com_int_ctl (unsigned char kind)
{

    if (kind == 0){
        /*****************************************************************************************************/
        /*   Disables SCI3 receive interrupts                                                                */
        /*****************************************************************************************************/
        SCI3.SCR3.BYTE      = 0x10;                         /* RCV int disable                             */

        /*****************************************************************************************************/
        /*   Enables TimerZ interrupts                                                                       */
        /*****************************************************************************************************/
        TZ0.TIER.BIT.IMIEA  = 1 ;                           /* timerZ IMFA  enable                         */

        /*****************************************************************************************************/
        /*   Cancels interrupt disable                                                                       */
        /*****************************************************************************************************/
        set_imask_ccr(0);                                   /* Enables interrupts                          */
    }
    else{
        /*****************************************************************************************************/
        /*   Sets interrupt disable (reason: to prevent other interrupts from coming in while an interrupt is being processed)*/
        /*****************************************************************************************************/
        set_imask_ccr(1);                                   /* Disables interrupts                          */
        /*****************************************************************************************************/
        /*   Enables SCI3 receive interrupts                                                                 */
        /*****************************************************************************************************/
        SCI3.SCR3.BYTE      = 0x50;                         /* Enables RCV int only                        */

        /*****************************************************************************************************/
        /*   Disables TimerZ interrupts                                                                      */
        /*****************************************************************************************************/
        TZ0.TIER.BIT.IMIEA  = 0 ;                           /* timerz IMFA  disable                        */
    }
}
```

```
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 4.4  SCI receive interrupt processing --------------------------------------------------------------------------------------  */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/******************************************************************************************************************************/
/*   1. Module name : h8_sci3                                                                                                 */
/*   2. Function overview: Interrupts from RS232C                                                                             */
/******************************************************************************************************************************/
#pragma interrupt( h8_sci3 )
void h8_sci3( void )
{
          // ##(program note)################################################################################################## */
          // ##  Generates an SCI interrupt if a key is entered into the console and jumps to here.                           ## */
          // ################################################################################################################## */
     int i , j ;
     unsigned int ret ;
     char *cmd_ptr, *addr_ptr , *data_ptr ;

     unsigned int    h8_addr;

     union {
      unsigned long  d_long ;
      unsigned int       d_int[2] ;
      unsigned char      d_byte[4];
     } buf;

     ret = NORMAL_END ;
     /*************************************************************************************************************************/
     /*   Clears set_imask_ccr to 0 to close the IREQ0-3 and SCI rcvint masks.                                                */
     /*   Makes TimerZ interrupts valid                                                                                       */
     /*************************************************************************************************************************/
     com_int_ctl(0) ;                                                  /* Clears ccr to 0 to make only TimerZ interrupts valid */
          // ##(program note)################################################################################################## */
          // ## During interrupts, the H8/3664 hardware sets CCR (I) to 1 to disable interrupts other than NMI              ## */
          // ## and address breaks.                                                                                          ## */
          // ## This routine handles the mask register to enable a timer interrupt during interrupt processing.             ## */
          // ################################################################################################################## */

     /*************************************************************************************************************************/
     /*   Receives data from the serial interface.                                                                            */
     /*************************************************************************************************************************/
     ret = com_read_sireal_data () ;                                       /* Receives data and stores it in the common buffer.*/
          // ##(program note)################################################################################################## */
          // ##  Receives only the first 1 byte with an interrupt and receives the remaining character string by polling in  ## */
          // ##  com_read_serial_data until a carriage return or a return code is received. Stores up to 40 characters       ## */
          // ##  of receive data in com_sci3_rcv_buf.                                                                        ## */
          // ################################################################################################################## */

     if (ret != 0) {goto exit ;}

     /*************************************************************************************************************************/
     /*   Reads the command contents.                                                                                         */
     /*************************************************************************************************************************/
     cmd_ptr = strtok(com_sci3_rcv_buf , " ") ;
```

```
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Executes H8 register processing                                                                                         */
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Reads H8 register                                                                                                       */
/*****************************************************************************************************************************/
if (strncmp(cmd_ptr,"hr",2) == 0){                                      /* command="hr"                                      */
          // ##(program note)############################################################################################ */
          // ##  Evaluates the first two characters.  If it is "hr", H8 register is read.                              ## */
          // ##  To specify "hr" as "abc", write if (strncmp(cmd_ptr,"abc",3) == 0 to evaluate three characters.       ## */
          // ############################################################################################################ */

     com_cnsl_msg("\n\r");
          // ##(program note)############################################################################################ */
          // ##  Enters a carriage return to the console display.                                                      ## */
          // ############################################################################################################ */

     addr_ptr = strtok(NULL , " ") ;                                    /* Reads ADDRESS                                     */
          // ##(program note)############################################################################################ */
          // ##  Checks the next character string using a space ( ) as a separator.                                    ## */
          // ##  To use a comma (,) as a separator, write addr_ptr = strtok(NULL , ",").                               ## */
          // ############################################################################################################ */

     if (addr_ptr == NULL) {                                            /* At addr space                                     */
          // ##(program note)############################################################################################ */
          // ##  Performs all register read processing without address specification if no data is read               ## */
          // ##  from addr_ptr(=NULL).                                                                                 ## */
          // ############################################################################################################ */

          /* Displays all registers.                                                                                 */

          h8_addr = 0xF700 ;                                            /* Displays F700-F77F.                               */
               com_cnsl_msg("            0       4        8      C\n\r");
               //        addr:YYYY = xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
          for (i=0; i<8; i++){
               // ##(program note)####################################################################################### */
               // ##  isplays the contents of registers from F700 to F77F in 16 bytes per line.                         ## */
               // ####################################################################################################### */
               com_cnsl_msg("  addr:%04X = ", h8_addr + i*16);

               for (j=0; j<4; j++){
                    com_cnsl_msg("%02X%02X%02X%02X "
                         , *((unsigned char *)h8_addr+i*16 + j*4)
                         , *((unsigned char *)h8_addr+i*16 + j*4 +1)
                         , *((unsigned char *)h8_addr+i*16 + j*4 +2)
                         , *((unsigned char *)h8_addr+i*16 + j*4 +3)) ;
               }
               com_cnsl_msg("\n\r") ;                                   /* A carriage return                                 */
          }


          h8_addr = 0xFF80 ;                                            /* Displays FF80-FFFF.                               */
               com_cnsl_msg("           0        4       8      C\n\r");
               //        addr:YYYY = xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
          for (i=0; i<8; i++){
```

```
            // ##(program note)######################################################################################## */
            // ##  Displays the contents of registers from FF80 to FFFF in 16 bytes per line.                          ## */
            // ######################################################################################################## */
            com_cnsl_msg("  addr:%04X = ", h8_addr + i*16);

            for (j=0; j<4; j++){
                com_cnsl_msg("%02X%02X%02X%02X "
                    , *((unsigned char *)h8_addr+i*16 + j*4)
                    , *((unsigned char *)h8_addr+i*16 + j*4 +1)
                    , *((unsigned char *)h8_addr+i*16 + j*4 +2)
                    , *((unsigned char *)h8_addr+i*16 + j*4 +3)) ;
            }
            com_cnsl_msg("\n\r") ;                          /* A carriage return                            */
        }
    }
    else{
        sscanf(addr_ptr , "%4hX" , &h8_addr) ;              /* Converts hexadecimal character string to integer.*/
            // ##(program note)######################################################################################## */
            // ##  If the next character string is read from addr_ptr, converts it into hexadecimal data            ## */
            // ##  and stores it in h8_addr to use it as the H8 register address.                                   ## */
            // ######################################################################################################## */

        goto h8_read_op ;
    }
}

/*******************************************************************************************************************/
/*   Writes to H8 register                                                                                         */
/*******************************************************************************************************************/
if (strncmp(cmd_ptr,"hw",2) == 0){                          /* command = "hw"                               */
        // ##(program note)############################################################################################ */
        // ##  Evaluates the first two characters.  If it is "hw", H8 register is written.                           ## */
        // ##  To specify "hr" as "abc", write if (strncmp(cmd_ptr,"abc",3) == 0 to evaluate three characters.       ## */
        // ############################################################################################################ */

    com_cnsl_msg("\n\r");
        // ##(program note)############################################################################################ */
        // ##  Enters a carriage return to the console display.                                                     ## */
        // ############################################################################################################ */

    addr_ptr = strtok(NULL , " ") ;                         /* Reads ADDRESS                                */
        // ##(program note)############################################################################################ */
        // ##  Checks the next character string using a space ( ) as a separator.                                   ## */
        // ##  To use a comma (,) as a separator, write addr_ptr = strtok(NULL , ",").                               ## */
        // ############################################################################################################ */
    if (addr_ptr == NULL) {
            // ##(program note)######################################################################################## */
            // ##  Performs all register read processing without address specification if no data is read          ## */
            // ##    from addr_ptr (=NULL).                                                                         ## */
            // ######################################################################################################## */

        ret = SCI3_INVALID_PARM ;
        goto exit ;
    }
    else{
        sscanf(addr_ptr , "%4hX" , &h8_addr) ;              /* Converts hexadecimal character string to integer.*/
            // ##(program note)######################################################################################## */
            // ##  If the next character string is read from addr_ptr, converts it into hexadecimal data            ## */
            // ##  and stores it in h8_addr to use it as the H8 register address.                                   ## */
            // ######################################################################################################## */
    }
    data_ptr = strtok(NULL , " ") ;                         /* Reads data                                   */
        // ##(program note)############################################################################################ */
        // ##  Checks the next character string using a space ( ) as a separator.                                   ## */
        // ############################################################################################################ */
```

```
        if (addr_ptr == NULL) {
                // ##(program note)######################################################################################## */
                // ##  Performs all register read processing without address specification if no data is read           ## */
                // ##  from addr_ptr (=NULL).                                                                             ## */
                // ####################################################################################################### */

            ret = SCI3_INVALID_PARM ;
            goto exit ;
        }
        else{
            sscanf(data_ptr , "%2X" , &buf.d_int[0]) ;                   /* Converts hexadecimal character string to integer.*/
                // ##(program note)######################################################################################## */
                // ##  If the next character string is read from data_ptr, converts it into hexadecimal data             ## */
                // ##  and stores it in buf.d_int[0] to use it as data to be written to the H8 register.                 ## */
                // ####################################################################################################### */
        }

        *((unsigned char *)h8_addr) = buf.d_byte[1] ;
            // ##(program note)############################################################################################ */
            // ##  Writes the specified data to the H8 register address specified by h8_addr.                            ## */
            // ##  Data to be written in the H8 register is 1 byte. Since the char type cannot be specified for sscanf,  ## */
            // ##  buf.d_int[0] is specified in int type and buf/d_byte[1] is specified at write.                        ## */
            // ############################################################################################################ */

        com_cnsl_msg("  (H8 REG WRITE %04X <- %02X) \n\r",h8_addr,buf.d_byte[1]);
            // ##(program note)############################################################################################ */
            // ##  Displays the write data on the console.                                                               ## */
            // ############################################################################################################ */

h8_read_op :
        buf.d_byte[0] = *((unsigned char *)h8_addr) ;

                                                                 /* Displays the read data on the console.            */
        com_cnsl_msg("  addr : %04X = %02X \n\r",h8_addr,buf.d_byte[0]);
            // ##(program note)############################################################################################ */
            // ##  Reads the specified register and displays it on the console.                                          ## */
            // ##  During a register write, reads the contents of the specified register and displays it on the console  ## */
            // ##  to verify the register write.                                                                          ## */
            // ############################################################################################################ */
    }

exit :
    /******************************************************************************************************************/
    /*   Displays the console command menu                                                                            */
    /******************************************************************************************************************/
    com_menu_disp() ;

    /******************************************************************************************************************/
    /*   Opens the SCI rcvint mask.                                                                                   */
    /******************************************************************************************************************/
    com_int_ctl(1) ;

}
```

```
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 4.5 Message output example source codes ----------------------------------------------------------------------------------   */
/* ---------------------------------------------------------------------------------------------------------------------------- */
     /*#########################################################################################################################*/
     /*   Displays debugging messages.                                                                                          */
     /*#########################################################################################################################*/
     com_cnsl_msg("H8 application sample program start!! \n\r") ;

     com_cnsl_msg("Ver Rev MRev : %02hX %02hX %02hX \n\r",MSC_VER,MSC_MAJOR_REV,MSC_MINOR_REV) ;



/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 5.      Sample program 10-E  TimerZ processing ------------------------------------------------------------------------------ */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */

/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 5.1    Addition of reset vectors ---------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
/*   Specify the jump destination to h8_timerz                                                                                   */

/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 5.2    Common variable definitions for TimerZ ------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
     struct    {
          int counter;                                                  /* 100 ms counter                                 */
          int wait_10ms;                                                /* Sets a wait time of 10 ms                      */
          int wait_100ms;                                               /* Sets the wait time in 100 ms units (common)    */
          int wait_100ms_scan;                                          /* Sets the wait time in 100 ms units (for I2C)   */
     }com_timer;
```

```
/* ---------------------------------------------------------------------------------------------------------------------------- */
/* 5.3 TimerZ initial settings     ------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------------- */
     /* ######################################################################################################################## */
     /* ######################################################################################################################## */
     /*                                                                                                                          */
     /*        Sets  TimerZ                                                                                                      */
     /*                                                                                                                          */
     /* ######################################################################################################################## */
     /* ######################################################################################################################## */
     /**************************************************************************************************************************/
     /*   Sets TimerZ initial settings                                                                                           */
     /**************************************************************************************************************************/
     TZ.TSTR.BYTE = 0x00 ;
     TZ.TMDR.BYTE = 0x00 ;
     TZ.TPMR.BYTE = 0x00 ;
     TZ.TFCR.BYTE = 0x00 ;
     TZ.TOER.BYTE = 0xFF ;
     TZ.TOCR.BYTE = 0x00 ;

     TZ0.TCR.BYTE   = 0x23 ;
                                                                      /* CCLR[2:0] = 001 Clears the counter              */
                                                                      /* when a GRA occurs compare/matching               */
                                                                      /* CKEG[1:0] = 00 Counts up at the rising edge      */
                                                                      /* TPSC[2:0] = 011 Counts using internal clock ϕ/8  */
     TZ0.TIORA.BYTE = 0x00 ;
                                                                      /* IOA[2:0] = 000                                   */
                                                                      /* GRA is used as an output compare register.       */
     TZ0.TIER.BYTE  = 0x01 ;
                                                                      /* IMIEA = 1 Enables IMFA                           */

     TZ0.GRA        = 20000 ;                                         /* Issues an interrupt every 10 msec                */
          /* ##(program note)################################################################################################### */
          /* ## The set values differ depending on the operating frequency of the microcomputer.                          ## */
          /* ## Please refer to the H8/3687 Hardware Manual.                                                               ## */
          /* ################################################################################################################### */

     TZ0.TCNT       = 0 ;                                             /* Clears the timer counter                         */

     /**************************************************************************************************************************/
     /*   Starts TimerZ                                                                                                          */
     /**************************************************************************************************************************/
     TZ.TSTR.BYTE   = 0x01 ;                                          /* timer start                                      */
                                                                      /* STR0 = 1 Start counting by TCNT_0                */
```

```
/* ------------------------------------------------------------------------------------------------------------------------- */
/* 5.4 TimerZ interrupt processing ---------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------- */
/*****************************************************************************************************************************/
/*   1. Module name: h8_timerz                                                                                               */
/*   2. Function overview: Interval timer processing every 10 msec*                                                          */
/*   3. History of revisions: REV  Date created/revised    Created/revised by     Revision contents                          */
/*                            000           2002.02.11            Ueda                   New                                 */
/*****************************************************************************************************************************/
#pragma interrupt( h8_timerz )
void h8_timerz( void )
{

     /************************************************************************************************************************/
     /*   Clears the source                                                                                                  */
     /************************************************************************************************************************/
     com_global.dummy = TZ0.TSR.BYTE;                                        /* dummy read                                   */

     TZ0.TSR.BIT.IMFA = 0;                                                   /* IMFA clear                                   */

     /************************************************************************************************************************/
     /*   -1 in units of 10 msec                                                                                             */
     /************************************************************************************************************************/
     if( com_timer.wait_10ms>0 )
          com_timer.wait_10ms --;

     /************************************************************************************************************************/
     /*   Increments the counter                                                                                             */
     /************************************************************************************************************************/
     com_timer.counter++;
     if( com_timer.counter >= 10 ){
          /*******************************************************************************************************************/
          /*   -1 in units of 100 msec                                                                                       */
          /*******************************************************************************************************************/
          if( com_timer.wait_100ms>0 )
               com_timer.wait_100ms --;
          if( com_timer.wait_100ms_scan>0 )
               com_timer.wait_100ms_scan --;

          com_timer.counter = 0;
     }

}
```

4. Reference Documents

- H8/3687 Group Hardware Manual (published by Renesas Technology Corp.)

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sep.29.03	—	First edition issued