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.

SH7780 Group

Example of Using the SCIF in Asynchronous Mode (Serial Data Transfer)

Introduction

This document gives an example of the settings for using the SCIF of the SH7780 Group in asynchronous mode and describes a sample application.

Target Device

SH7780 (MS7780SE03 Solution Engine by Hitachi ULSI Systems)

Contents

1. Preface

1.1 Specification

SCIF channel 0 is used in asynchronous mode to transfer data. Data received by the connected device are sent back after reception is completed.

An overview of the communication is given below and a connection example is shown in figure 1.

- Bit rate: 115200 bps
- Format: 8-bit data length, parity bit added, 1 stop bit

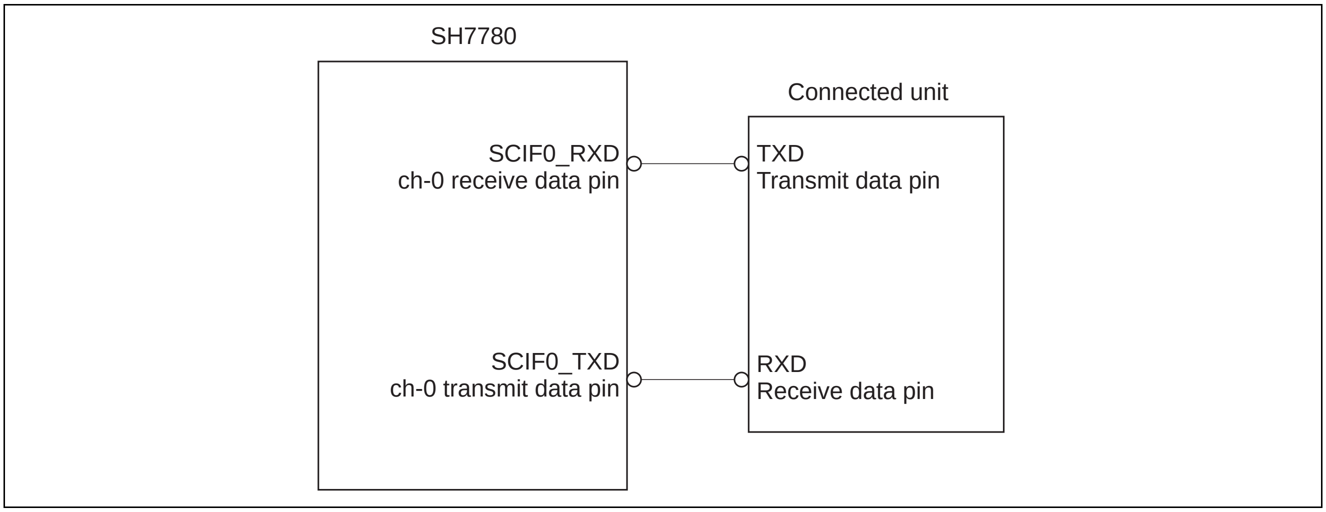


Figure 1 Example of Connection for SCIF (Asynchronous Mode) Operation

1.2 Module Used

- SCIF channel 0

1.3 Applicable Conditions

• MCU	SH7780	
• Operating frequency	Internal clock	: 400 MHz
	SuperHyway clock	: 200 MHz
	Peripheral clock	: 33 MHz
	DDR clock	: 160 MHz
	External clock	: 33 MHz
	PCI bus clock	: 33 MHz
• Clock operating mode	Mode 3 (MODE7 = low, MODE2 = low, MODE1 = high, MODE0 = high)	
• Data alignment	Little endian	
• Addressing mode	29-bit	
• C compiler	SuperHRISC Engine Family C/C++ Compiler Package Ver.9.1.0 (manufactured by Renesas Technology)	

1.4 Related Application Note

The operation of the reference program for this document was confirmed with the setting conditions described in the following application note: *SH7780 Initialization Example* (REJ06B0712-0100). Please refer to the application note in combination with this one.

2. Description of the Sample Application

The sample program uses channel 0 of the SCIF in asynchronous mode to transmit and receive data. Raw received data are sent back by the connected device as transmitted data.

In processing for reception, received data are read out from within interrupt handling for the receive data full interrupt. Data are transmitted without using interrupts during the interrupt handling process.

2.1 Description of the Sample Program

This sample program consists of the following four source files.

(1) scif.c
(2) main.c
(3) intprg.c
(4) vecttbl.src

(1) scif.c describes the function used in this program to set up SCIF operations, the receive data full interrupt function to which a jump is made from the interrupt function of intprg.c that was included *SH7780 Initialization Example*, the receive error interrupt function, and the break interrupt function. This program code is not included in the application for *SH7780 Initialization Example*, which is used as its basis.

(2) main.c sets the status register (SR) and calls the function to set up the SCIF operation.
Change main.c that was included with *SH7780 Initialization Example* as required to match main.c for this sample program.

(3) intprg.c describes the interrupt program called from the exception/interrupt handler.
Change intprg.c that was included with *SH7780 Initialization Example* as required to match intprg.c for this sample program.

(4) vecttbl.src describes the exception/interrupt vector table (including vector-table entries), and interrupt mask table.
To ensure that interrupts that have been accepted are not accepted again while they are being processed, the interrupt mask levels to be set in the IMASK bits of the status register should be described in the interrupt mask table.
Change vecttbl.src that was included with *SH7780 Initialization Example* as required to match vecttbl.src for this sample program.

2.2 Operational Overview of Module Used

The SCIF can perform serial communications in two modes: an asynchronous mode in which synchronization is achieved character by character, and a clock synchronous mode in which communications are synchronized with clock pulses.

64-stage FIFO buffers are provided for both transmission and reception, enabling fast, efficient, and continuous communications. Only channel 0 has modem control functions ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$).

The serial mode register (SCSMR) is used to set the transfer format. The clock source for the SCIF is determined by the combination of settings of the C/A bit in SCSMR and the CKE1 and CKE0 bits in the serial control register (SCSCR).

(1) Asynchronous mode

- Data length: 7 or 8 bits
- LSB first for data transmission/reception
- Choice of appending parity bit and 1 or 2 stop bits
- Receive error detection: Framing, parity, and overrun errors
- Break signal detection: A break is detected when a framing error lasts for more than 1 frame length at Space 0 (low level).
- Sending a break signal: The input/output condition and level of the SCIF_TXD pin are determined by bits SPB2IO and SPB2DT in SCSPTR. This feature can be used to send a break signal. To send a break signal during serial transmission, clear the SPB2DT bit to 0 (designating low level), and then clear the TE bit to 0 (halting transmission). When the TE bit is cleared to 0, the transmitter is initialized, regardless of the current transmission state, and 0 is output from the SCIF_TXD pin.
- Indication of the number of data bytes stored in the transmit and receive FIFO registers (SCTFDR, SCRFDR)
- SCIF clock source: Choice of peripheral clock or SCIF_SCK input clock
  When peripheral clock (Pck) is selected: The SCIF operates on the baud rate generator clock.
  When SCIF_SCK input clock is selected: A clock with a frequency of 16 times the bit rate must be input.

(2) Clocked synchronous mode

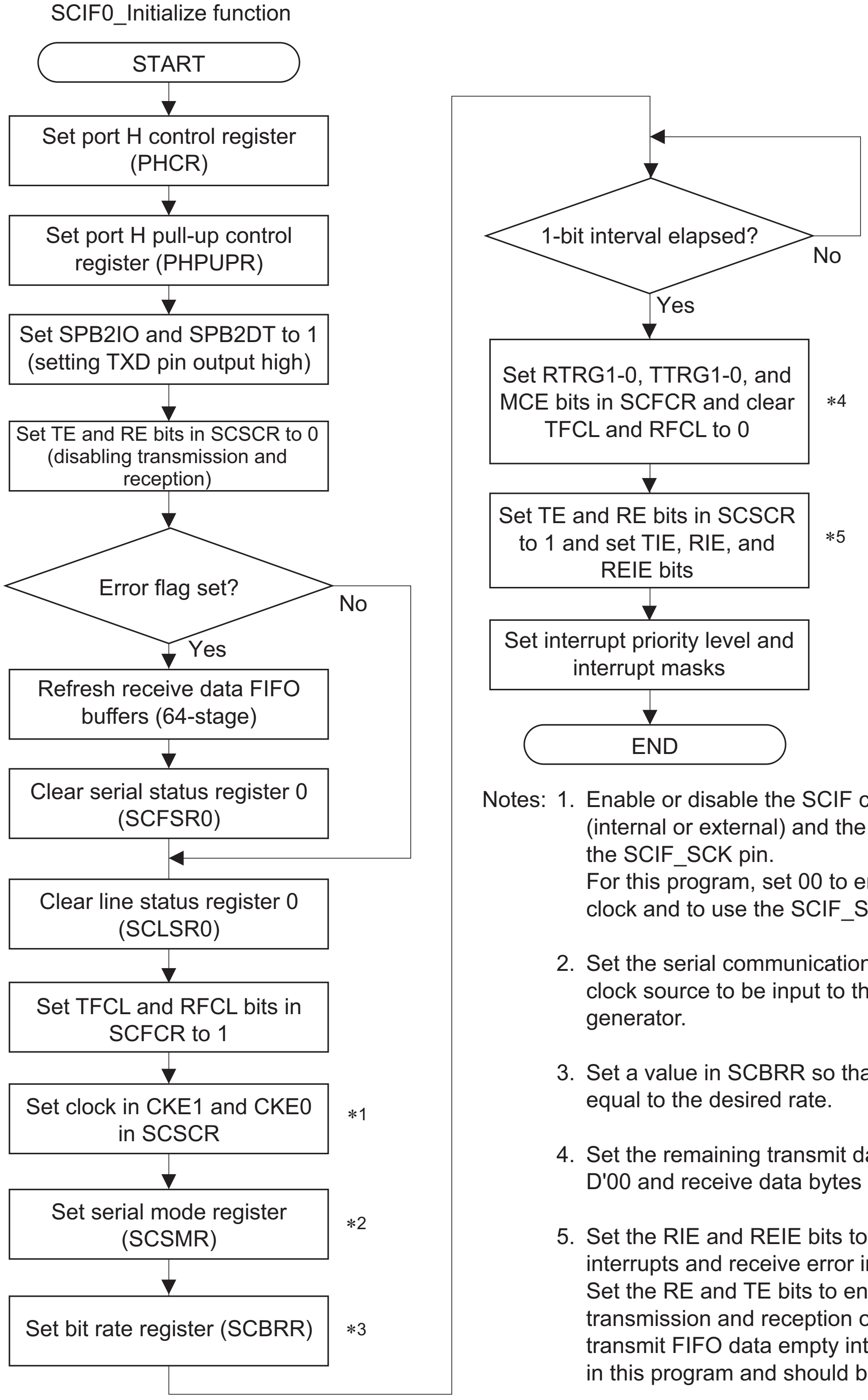
- Data length: 8 bits
- LSB first for data transmission/reception
- Detection of overrun errors during reception
- SCIF clock source: Choice of peripheral clock or SCIF_SCK input clock
  When peripheral clock (Pck) is selected: The SCIF operates on the baud rate generator clock.
  When SCIF_SCK input clock is selected: A clock with a frequency of 16 times the bit rate must be input.

2.3 Procedure for Setting Module Used

This section describes the procedure for setting up SCIF channel 0 for asynchronous mode operation.

In this program, the following initial settings are made at the beginning of the main function on the assumption that this is based on the program for *SH7780 Initialization Example*. Since operation in privileged mode is a precondition for the program doing this, take care with regard to the processing mode when you adapt this code for use with other programs etc.

Figure 2 in the next page shows an example of the flow for setting operations of the SCIF channel 0 in asynchronous mode. For details on the settings of individual registers, see the *SH7780 Group Hardware Manual*.


SCIF0_Initialize function
START
Set port H control register (PHCR)
Set port H pull-up control register (PHPUPR)
Set SPB2IO and SPB2DT to 1 (setting TXD pin output high)
Set TE and RE bits in SCSCR to 0 (disabling transmission and reception)
Error flag set?
No
Yes
Refresh receive data FIFO buffers (64-stage)
Clear serial status register 0 (SCFSR0)
Clear line status register 0 (SCLSR0)
Set TFCL and RFCL bits in SCFCR to 1
Set clock in CKE1 and CKE0 in SCSCR
*1
Set serial mode register (SCSMR)
*2
Set bit rate register (SCBRR)
*3
1-bit interval elapsed?
No
Yes
Set RTRG1-0, TTRG1-0, and MCE bits in SCFCR and clear TFCL and RFCL to 0
*4
Set TE and RE bits in SCSCR to 1 and set TIE, RIE, and REIE bits
*5
Set interrupt priority level and interrupt masks
END

Notes: 1. Enable or disable the SCIF clock source (internal or external) and the clock output from the SCIF_SCK pin.
For this program, set 00 to enable internal clock and to use the SCIF_SCK pin as a port.

2. Set the serial communication format and the clock source to be input to the baud rate generator.

3. Set a value in SCBRR so that the bit rate is equal to the desired rate.

4. Set the remaining transmit data bytes trigger to D'00 and receive data bytes trigger to D'16.

5. Set the RIE and REIE bits to enable receive interrupts and receive error interrupts.
Set the RE and TE bits to enable serial transmission and reception operations. The transmit FIFO data empty interrupt is not used in this program and should be disabled.

Figure 2 Flow for Setting up SCIF Asynchronous Mode

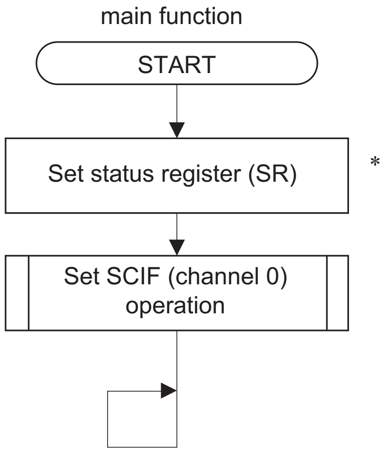
2.4 Processing Sequence of Sample Program

Table 1 gives examples for setting SCIF-related registers.

Figures 3 to 6 provide sample flowcharts of the main function and interrupt handling of the sample program.

Table 1 SCIF Setting

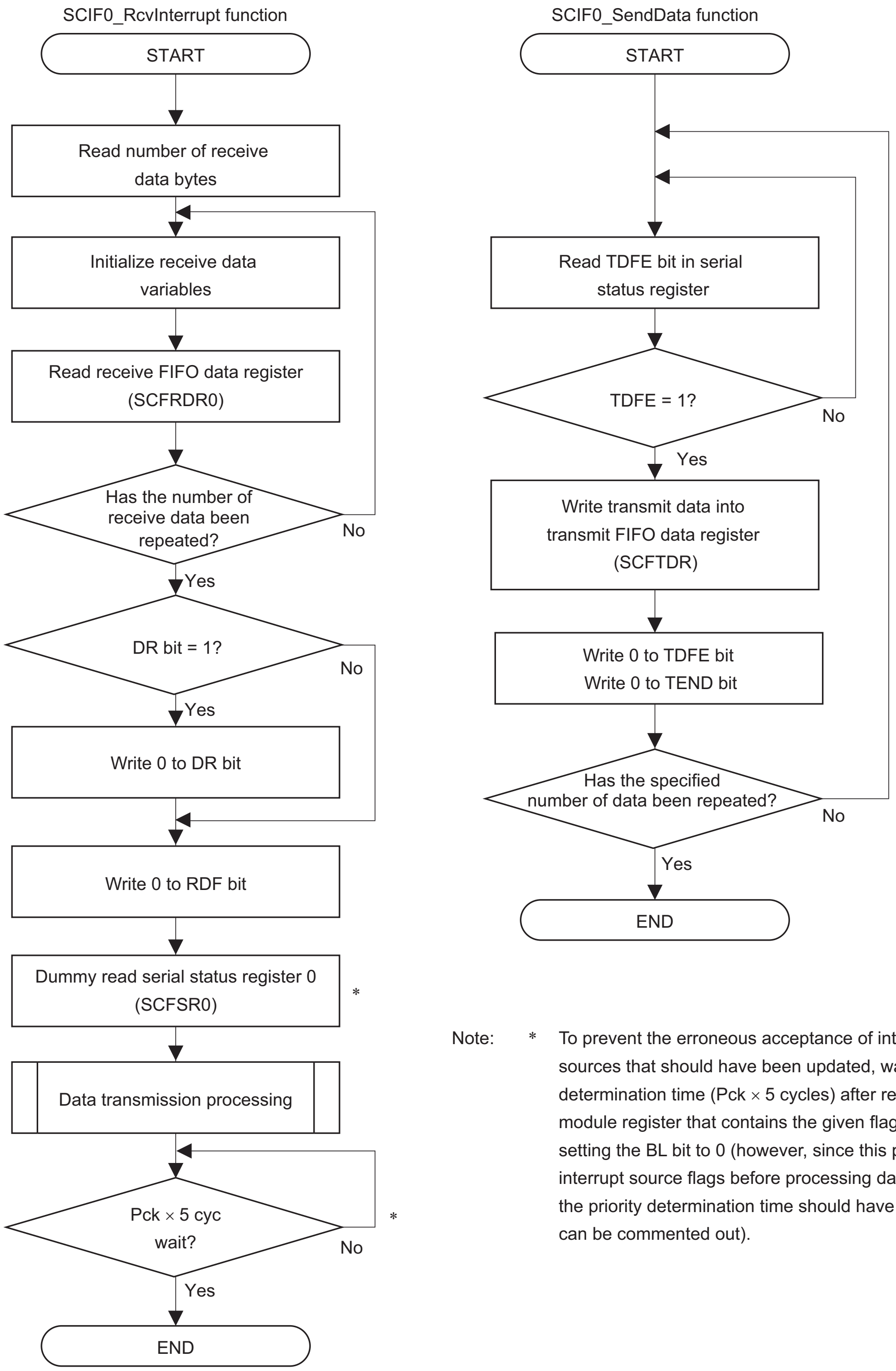
Register Name	Address	Setting	Function
Interrupt priority register 2 (INT2PRI2)	H'FFD4 0008	H'1F00 0000	SCIF-ch0 interrupt priority level: 31
Interrupt mask clear register (INT2MSKCR)	H'FFD4 003C	H'0000 0008	SCIF-ch0 interrupt mask clear
Port H control register (PHCR)	H'FFEA 000E	H' FC80	PH3 to PH0 SCIF0 module functions
Port H pull-up control register (PHPUPR)	H'FFEA 004E	H' 0000	PH0 to PH7 pull-up off
Serial mode register (SCSMR)	H'FFD8 0004	H' 0020	Asynchronous mode 8-bit data Parity enabled (even parity) 1 stop bit Clock source: Pck
Serial control register (SCSCR)	H'FFD8 0008	H' 0078	Operating on internal clock RXI, ERI, and BRI interrupts enabled SCK pin used as a port
Bit rate register (SCBBR)	H'FFD8 000C	H' 08	
FIFO control register (SCFCR)	H'FFD8 0010	H' 0070	RDF flag trigger number: 16



MD: "1", privileged mode
BL: "0", exeptions and interrupts are not blocked.
IMASK: Interrupt mask level = 14

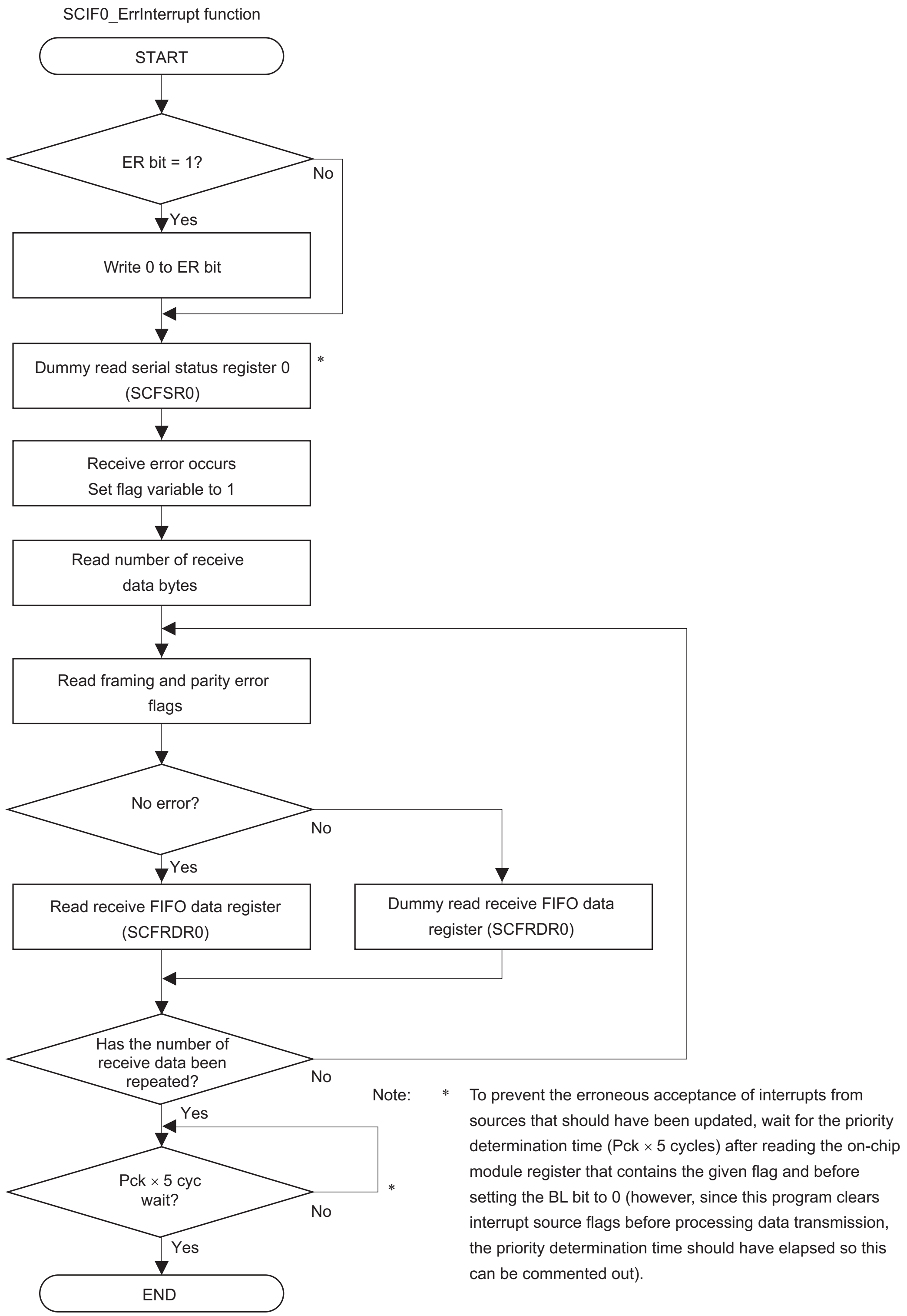
Note: * Since this program directly changes registers for peripheral functions, the MD bit (processing mode bit) is set to 1 (privileged mode). If this bit is changed to "0" (user mode) and the function for setting operations is called, access to the registers for peripheral functions leads to an address error.

Figure 3 Flow of Processing by the Main Function of the Sample Program (Asynchronous Serial Data Transmission)


SCIF0_RcvInterrupt function
START
Read number of receive data bytes
Initialize receive data variables
Read receive FIFO data register (SCFRDR0)
Has the number of receive data been repeated?
No
Yes
DR bit = 1?
No
Yes
Write 0 to DR bit
Write 0 to RDF bit
Dummy read serial status register 0 (SCFSR0)
*
Data transmission processing
Pck × 5 cyc wait?
*
No
Yes
END
SCIF0_SendData function
START
Read TDFE bit in serial status register
TDFE = 1?
No
Yes
Write transmit data into transmit FIFO data register (SCFTDR)
Write 0 to TDFE bit
Write 0 to TEND bit
Has the specified number of data been repeated?
No
Yes
END

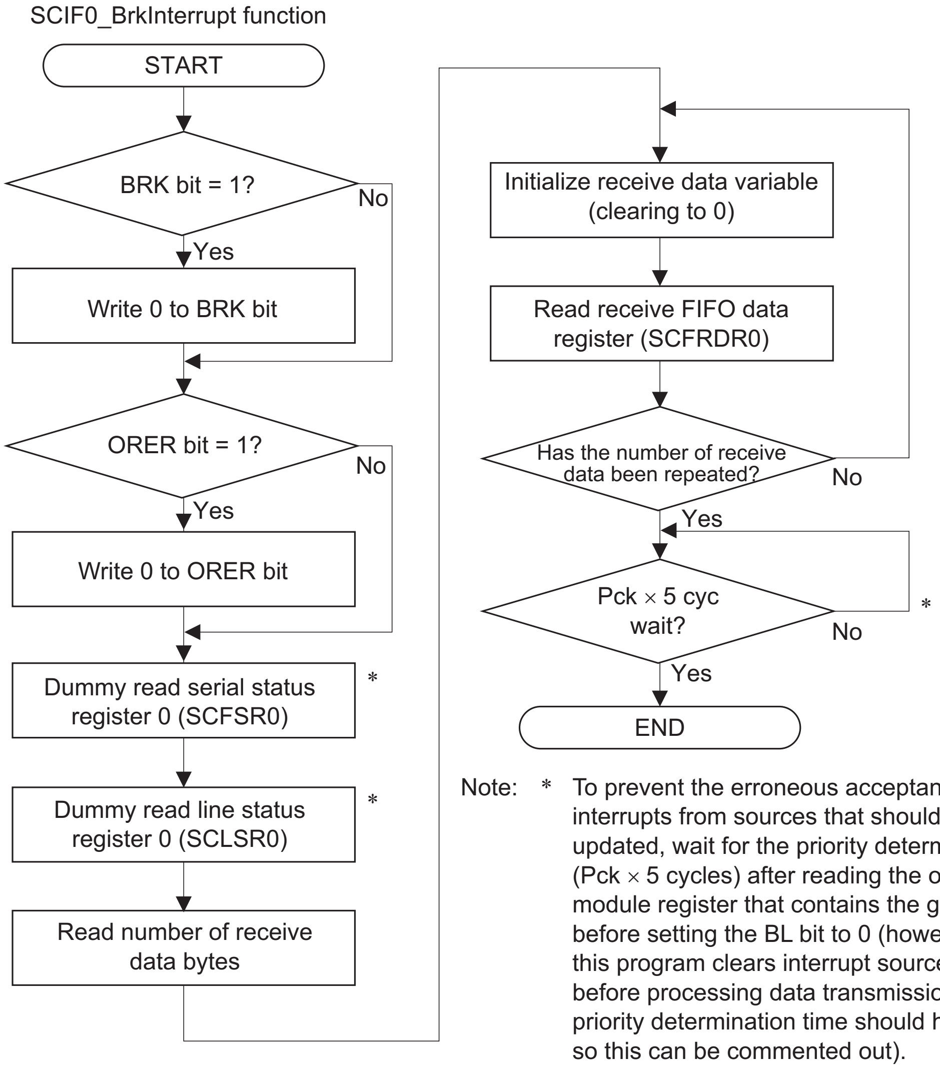
Note: * To prevent the erroneous acceptance of interrupts from sources that should have been updated, wait for the priority determination time (Pck × 5 cycles) after reading the on-chip module register that contains the given flag and before setting the BL bit to 0 (however, since this program clears interrupt source flags before processing data transmission, the priority determination time should have elapsed so this can be commented out).

Figure 4 Flow of RXI (Receive Data Full) Interrupt Handling and Data Transmission


SCIF0_ErrInterrupt function
START
ER bit = 1?
No
Yes
Write 0 to ER bit
Dummy read serial status register 0 (SCFSR0)
*
Receive error occurs
Set flag variable to 1
Read number of receive data bytes
Read framing and parity error flags
No error?
No
Yes
Read receive FIFO data register (SCFRDR0)
Dummy read receive FIFO data register (SCFRDR0)
Has the number of receive data been repeated?
No
Yes
Pck × 5 cyc wait?
No
*
Yes
END

Note: * To prevent the erroneous acceptance of interrupts from sources that should have been updated, wait for the priority determination time (Pck × 5 cycles) after reading the on-chip module register that contains the given flag and before setting the BL bit to 0 (however, since this program clears interrupt source flags before processing data transmission, the priority determination time should have elapsed so this can be commented out).

Figure 5 Flow of ERI (Receive Error) Interrupt Handling


SCIF0_BrkInterrupt function
START
BRK bit = 1?
No
Yes
Write 0 to BRK bit
ORER bit = 1?
No
Yes
Write 0 to ORER bit
Dummy read serial status register 0 (SCFSR0)
*
Dummy read line status register 0 (SCLSR0)
*
Read number of receive data bytes
Initialize receive data variable (clearing to 0)
Read receive FIFO data register (SCFRDR0)
Has the number of receive data been repeated?
No
Yes
Pck × 5 cyc wait?
*
No
Yes
END

Note: * To prevent the erroneous acceptance of interrupts from sources that should have been updated, wait for the priority determination time (Pck × 5 cycles) after reading the on-chip module register that contains the given flag and before setting the BL bit to 0 (however, since this program clears interrupt source flags before processing data transmission, the priority determination time should have elapsed so this can be commented out).

Figure 6 Flow of BRK (Break) Interrupt Handling

3. Listing of Sample Program

1. Sample Program Listing: "main.c"

```
/*""FILE COMMENT""********************************************************************
*    System Name :   SH7780 Sample Program
*    File Name   :   main.c
*    Version     :   1.00.00
*    Contents    :   SH7780 Initialize Program
*    Model       :   Hitachi_ULSI_Systems SolutionEngine MS7780SE03
*    CPU         :   SH7780
*    Compiler    :   SHC.9.1.00
*    OS          :   none
*
*    note        :   < Caution >
*                    This sample program is provided simply as a reference and
*                    its operation is not guaranteed.
*                    Use this sample program as a technical reference when
*                    developing software.
*
* Copyright (C) 2007 Renesas Technology Corp. All Rights Reserved
*
*    History     :   2007/12/26  ver 1.00.00
*
*************************************************************************************/
#include <machine.h>

/* --- Function Definition(internal) --- */

/* --- Symbol Definition --- */
#define  SR_Init        0x400000e0         /* Privileged mode, RB,BL=0, IMASK level 14 */

/***** Function Comment ***************************************
* Outline    : main
*--------------------------------------------------------------
* Declaration    : void main(void)
*--------------------------------------------------------------
* Functional description:
*              main function
*--------------------------------------------------------------
* Return Value   : -
* Argument       : -
*--------------------------------------------------------------
* Input          : -
* Output     : -
*--------------------------------------------------------------
* Notes          : -
****** Function Comment End ***********************************/
void main(void)
{
  set_cr(SR_Init);          /* Set SR "Privileged mode, RB,BL=0, IMASK level 14" */

  SCIF0_Initialize();       /* SCIF0 Initialize  (additional part from Initialize program)*/

  while(1)
  {
  }
}
```

2. Sample Program Listing: "scif.c" (1)

```
/*""FILE COMMENT""**********************************************************************
 *     System Name :   SH7780 Sample Program
 *     File Name   :   scif.c
 *     Version     :   1.00.00
 *     Contents    :   SH7780 SCIF0 transmit Program
 *     Model       :   Hitachi_ULSI_Systems SolutionEngine MS7780SE03
 *     CPU         :   SH7780
 *     Compiler    :   SHC.9.1.00
 *     OS          :   none
 *
 *     note        :   < Caution >
 *                     This sample program is provided simply as a reference and
 *                     its operation is not guaranteed.
 *                     Use this sample program as a technical reference when
 *                     developing software.
 *
 * Copyright (C) 2007 Renesas Technology Corp. All Rights Reserved
 *
 *     History     :   2007/12/26  ver 1.00.00
 *
***************************************************************************************/
#include <machine.h>

/* --- Function Definition(internal) --- */
void SCIF0_Initialize(void);
void SCIF0_SendData(void);
void SCIF0_RcvInterrupt(void);
void SCIF0_ErrInterrupt(void);
void SCIF0_BrkInterrupt(void);
static void int_responstime_wait(unsigned int wait_time);

/* --- Symbol Definition --- */
union st_scsmr{                             /* struct SCSMR*/
  unsigned short     WORD;              /* Word Access */
  struct{                               /* Bit Access  */
      unsigned short     :8;            /*             */
      unsigned short CA  :1;            /*      CA     */
      unsigned short CHR :1;            /*      CHR    */
      unsigned short PE  :1;            /*      PE     */
      unsigned short OE  :1;            /*      OE     */
      unsigned short STOP :1;           /*      STOP   */
      unsigned short     :1;            /*             */
      unsigned short CKS :2;            /*      CKS    */
  }BIT;
};

union st_scscr{                             /* struct SCSCR*/
  unsigned short     WORD;              /* Word Access */
  struct{                               /* Bit Access  */
      unsigned short     :8;            /*             */
      unsigned short TIE :1;            /*      TIE    */
      unsigned short RIE :1;            /*      RIE    */
      unsigned short TE  :1;            /*      TE     */
      unsigned short RE  :1;            /*      RE     */
      unsigned short REIE :1;           /*      REIE   */
      unsigned short     :1;            /*             */
```

3. Sample Program Listing: "scif.c" (2)

```
      unsigned short CKE :2;        /*     CKE    */
  }BIT;
};

union st_scfsr{                         /* struct SCFSR*/
  unsigned short     WORD;          /* Word Access */
  struct{                           /* Bit Access  */
      unsigned short     :8;        /*             */
      unsigned short ER  :1;        /*     ER      */
      unsigned short TEND :1;       /*     TEND    */
      unsigned short TDFE :1;       /*     TDFE    */
      unsigned short BRK :1;        /*     BRK     */
      unsigned short FER :1;        /*     FER     */
      unsigned short PER :1;        /*     PER     */
      unsigned short RDF :1;        /*     RDF     */
      unsigned short DR  :1;        /*     DR      */
  }BIT;
};

union st_scfcr{                         /* struct SCFCR*/
  unsigned short     WORD;          /* Word Access */
  struct{                           /* Bit Access  */
      unsigned short      :5;       /*             */
      unsigned short RSTRG :3;      /*     RSTRG   */
      unsigned short RTRG :2;       /*     RTRG    */
      unsigned short TTRG :2;       /*     TTRG    */
      unsigned short MCE  :1;       /*     MCE     */
      unsigned short TFCL :1;       /*     TFCL    */
      unsigned short RFCL :1;       /*     RFCL    */
      unsigned short LOOP :1;       /*     LOOP    */
  }BIT;
};

union st_scsptr{                    /* struct SCSPTR */
  unsigned short     WORD;          /* Word Access */
  struct{                           /* Bit Access  */
      unsigned short         :8;    /*             */
      unsigned short RTSIO   :1;    /*     RTSIO   */
      unsigned short RTSDT   :1;    /*     RTSDT   */
      unsigned short CTSIO   :1;    /*     CTSIO   */
      unsigned short CTSDT   :1;    /*     CTSDT   */
      unsigned short SCKIO   :1;    /*     SCKIO   */
      unsigned short SCKDT   :1;    /*     SCKDT   */
      unsigned short SPB2IO  :1;    /*     SPB2IO  */
      unsigned short SPB2DT  :1;    /*     SPB2DT  */
  }BIT;
};

union st_sclsr{                         /* struct SCLSR*/
  unsigned short     WORD;          /* Word Access */
  struct{                           /* Bit Access  */
      unsigned short         :15;   /*             */
      unsigned short ORER    :1;    /*     ORER    */
  }BIT;
};

```

4. Sample Program Listing: "scif.c" (3)

```
union st_screr{                       /* struct SCRER*/
  unsigned short     WORD;            /* Word Access */
  struct{                             /* Bit Access  */
      unsigned short         :2;      /*             */
      unsigned short PER     :6;      /*     PER     */
      unsigned short         :2;      /*             */
      unsigned short FER     :6;      /*     FER     */
  }BIT;
};

#define SCSMR0      (*(volatile union st_scsmr *)0xFFE00000)   /* SCSMR0 Address */
#define SCBRR0      (*(volatile unsigned char  *)0xFFE00004)   /* SCBRR0 Address */
#define SCSCR0      (*(volatile union st_scscr *)0xFFE00008)   /* SCSCR0 Address */
#define SCFTDR0     (*(volatile unsigned char  *)0xFFE0000C)   /* SCFTDR0 Address */
#define SCFSR0      (*(volatile union st_scfsr *)0xFFE00010)   /* SCFSR0 Address */
#define SCFRDR0     (*(volatile unsigned char  *)0xFFE00014)   /* SCFRDR0 Address */
#define SCFCR0      (*(volatile union st_scfcr *)0xFFE00018)   /* SCFCR0 Address */
#define SCRFDR0     (*(volatile unsigned short *)0xFFE00020)   /* SCRFDR0 Address */
#define SCLSR0      (*(volatile union st_sclsr *)0xFFE00028)   /* SCLSR0 Address */
#define SCSPTR0     (*(volatile union st_scsptr *)0xFFE00024) /* SCSPTR0 Address */

#define PHCR    (*(volatile unsigned short *)0xFFEA000E)       /* PHCR Address */
#define PHPUPR      (*(volatile unsigned char  *)0xFFEA004E)   /* PHPUPR Address */
#define INT2PRI2 (*(volatile unsigned int   *)0xFFD40008)      /* INT2PRI2 Address */
#define INT2MSKCR   (*(volatile unsigned int   *)0xFFD4003C)   /* INT2MSKCR Address */

#define NUM_RCV_DATABUF     (0x40)

#define INTC_RESPONSEWAIT   (0x00000014)       /* INT response wait Pck 5cycle
                                              H'14 = (1/Pck*5cyc) / (1/Ick*3cyc) */

/* --- RAM allocation variable declaration --- */
volatile unsigned char u1SCIF0Rcvdata[16];
volatile unsigned char u1FlgSCIF0BrkInt;
volatile unsigned char u1FlgSCIF0RcvErrInt;
volatile unsigned short u2NumRcvData;

/***** Function Comment ****************************************
* Outline    : SCIF0_Initialize
*----------------------------------------------------------------
* Declaration    : void SCIF0_Initialize(void)
*----------------------------------------------------------------
* Functional description:
*               SCIF0 Initialize
*----------------------------------------------------------------
* Return Value    : -
* Argument        : -
*----------------------------------------------------------------
* Input           : -
* Output     : -
*----------------------------------------------------------------
* Notes           : -
****** Function Comment End ***********************************/
void SCIF0_Initialize(void)
{
  volatile unsigned short   u2loop;
```

5. Sample Program Listing: "scif.c" (4)

```
    volatile unsigned char u1dummy;
    volatile unsigned char u1ErrFlg;

    PHCR = 0xFC80;            /* PH4-PH0 mode select "SCIF0" */
    PHPUPR = 0x00;            /* set PH7-PH0 pullup off */

    SCSPTR0.BIT.SPB2IO = 1;       /* SPB2DT bit value is output to the SCIF_TXD pin */
    SCSPTR0.BIT.SPB2DT = 1;       /* output data is Hi-level */

    SCSCR0.WORD = 0x0000;     /* TE,RE bit clear */
    u1ErrFlg = ((SCFSR0.BIT.ER)||(SCFSR0.BIT.DR)||(SCFSR0.BIT.BRK)||(SCLSR0.BIT.ORER));/* read error flag */

    if(u1ErrFlg == 0x01)
    {
        for(u2loop = 0; u2loop < NUM_RCV_DATABUF ; u2loop++)
        {
            u1dummy = SCFRDR0;    /* receive data FIFO buffer refresh */
        }
        SCFSR0.WORD &= 0x0000;    /* clear all status bit */
        SCLSR0.BIT.ORER &= 0x00;  /* clear ORER bit */
    }

    SCFCR0.WORD = 0x0006;     /* set TFCL,RFCL bit, loopback disable, modem control disable */
    SCSCR0.WORD = 0x0000;     /* internal clock, SCIF_SCK general port */
    SCSMR0.WORD = 0x0020;     /* UART, 8bit, parity enable, even parity, 1stopbit, Pck */
    SCBRR0 = 0x08;            /* baud rate 115200bps */

    for(u2loop = 0; u2loop < 0x0d90 ; u2loop++)      /* 1-bit interval elapsed? */
    {
        nop();
    }

    SCFCR0.WORD = 0x0070;     /* clear TFCL,RFCL bit,RTRG0,1,TTRG */
    SCSCR0.WORD = 0x0078;     /* transimit enable, receive enable, receive interrupt enable */

    INT2PRI2 |= 0x1F000000;   /* SCIF ch0 interrupt level 31 */
    INT2MSKCR = 0x00000008;   /* SCIF ch0 interrupt mask clear */
}

/***** Function Comment ***************************************
* Outline    : SCIF0_ReceiveInterrupt
*-------------------------------------------------------------
* Declaration    : void SCIF0_RcvInterrupt(void)
*-------------------------------------------------------------
* Functional description:
*              SCIF0 Recieve Data full interrupt
*-------------------------------------------------------------
* Return Value   : -
* Argument       : -
*-------------------------------------------------------------
* Input          : -
* Output     : -
*-------------------------------------------------------------
* Notes          : -
****** Function Comment End ***********************************/
```

6. Sample Program Listing: "scif.c" (5)

```
void SCIF0_RcvInterrupt(void)
{
  volatile unsigned short   u2loop;
  volatile unsigned short u2dummy;

  u2NumRcvData = SCRFDR0;

  for(u2loop = 0; u2loop < u2NumRcvData ; u2loop++)
  {
      u1SCIF0Rcvdata[u2loop] = 0x00;
      u1SCIF0Rcvdata[u2loop] = SCFRDR0; /* read receive data */
  }

  if(SCFSR0.BIT.DR == 0x01)
  {
      SCFSR0.BIT.DR &= 0x00;        /* clear DR bit */
  }
  SCFSR0.BIT.RDF &= 0x00;           /* clear RDF bit */

  u2dummy = SCFSR0.WORD; /* dummy read */

  SCIF0_SendData();

  int_responstime_wait(INTC_RESPONSEWAIT);  /* 5cyc(Pck=33MHz) wait */
}

/***** Function Comment ***************************************
* Outline     : SCIF0_SendData
*--------------------------------------------------------------
* Declaration    : void SCIF0_SendData(void)
*--------------------------------------------------------------
* Functional description:
*              SCIF0 Send data
*--------------------------------------------------------------
* Return Value   : -
* Argument       : -
*--------------------------------------------------------------
* Input          : -
* Output      : -
*--------------------------------------------------------------
* Notes          : -
****** Function Comment End ***********************************/
void SCIF0_SendData(void)
{
  volatile unsigned short   u2loop;

  for(u2loop = 0; u2loop < u2NumRcvData ; u2loop++)
  {
      while(SCFSR0.BIT.TDFE != 1)
      {
          nop();
      }

      SCFTDR0 = u1SCIF0Rcvdata[u2loop];    /* set Send Data */
      SCFSR0.WORD &= 0x9f;                 /* clear TDFE,TEND bit */
  }
```

7. Sample Program Listing: "scif.c" (6)

```
}

/***** Function Comment ***************************************
* Outline     : SCIF0_ErrInterrupt
*--------------------------------------------------------------
* Declaration    : void SCIF0_ErrInterrupt(void)
*--------------------------------------------------------------
* Functional description:
*              SCIF0 Receive error interrupt
*--------------------------------------------------------------
* Return Value   : -
* Argument       : -
*--------------------------------------------------------------
* Input          : -
* Output     : -
*--------------------------------------------------------------
* Notes          : -
****** Function Comment End ***********************************/
void SCIF0_ErrInterrupt(void)
{
  volatile unsigned short   u2loop;
  volatile unsigned char u1ErrFlg;
  volatile unsigned char u1dummy;
  volatile unsigned short   u2dummy;

  if(SCFSR0.BIT.ER == 0x01)
  {
      SCFSR0.BIT.ER &= 0x00;        /* clear ER bit */
  }

  u2dummy = SCFSR0.WORD; /* dummy read */

  u1FlgSCIF0BrkInt = 1;          /* set Recieve Error Interrupt flag */

  u2NumRcvData = SCRFDR0;

  for(u2loop = 0 ; u2loop < u2NumRcvData ; u2loop++)
  {
      u1ErrFlg = ((SCFSR0.BIT.FER)||(SCFSR0.BIT.PER));

      if(u1ErrFlg == 0x00)
      {
          u1SCIF0Rcvdata[u2loop] = SCFRDR0; /* read receive data */
      }
      else
      {
          u1SCIF0Rcvdata[u2loop] = 0x00;
          u1dummy = SCFRDR0;        /* dummy read */
      }
  }

  int_responstime_wait(INTC_RESPONSEWAIT);  /* 5cyc(Pck=33MHz) wait */
}

/***** Function Comment ***************************************
* Outline     : SCIF0_BrkInterrupt
```

8. Sample Program Listing: "scif.c" (7)

```
*--------------------------------------------------------------
* Declaration     : void SCIF0_BrkInterrupt(void)
*--------------------------------------------------------------
* Functional description:
*              SCIF0 Break interrupt
*--------------------------------------------------------------
* Return Value    : -
* Argument        : -
*--------------------------------------------------------------
* Input           : -;
* Output      : -
*--------------------------------------------------------------
* Notes           : -
****** Function Comment End ***********************************/
void SCIF0_BrkInterrupt(void)
{
  volatile unsigned short    u2loop;
  volatile unsigned short    u2dummy;

  if(SCFSR0.BIT.BRK == 0x01)
  {
      SCFSR0.BIT.BRK &= 0x00;            /* clear BRK bit */
  }
  else if(SCLSR0.BIT.ORER == 0x01)
  {
      SCLSR0.BIT.ORER &= 0x00;           /* clear ORER bit */
  }

  u2dummy = SCFSR0.WORD; /* dummy read */
  u2dummy = SCLSR0.WORD; /* dummy read */

  u1FlgSCIF0BrkInt = 1;      /* set BreakInterrupt flag */

  u2NumRcvData = SCRFDR0;
  for(u2loop = 0 ; u2loop < u2NumRcvData ; u2loop++)
  {
      u1SCIF0Rcvdata[u2loop] = 0x00;     /* receive data initial */
      u1SCIF0Rcvdata[u2loop] = SCFRDR0; /* read receive data */
  }

  int_responstime_wait(INTC_RESPONSEWAIT);  /* 5cyc(Pck=33MHz) wait */
}

/***** Function Comment ***************************************
* Outline     : SCIF0_SendBrkSignal
*--------------------------------------------------------------
* Declaration     : void SCIF0_SendBrkSignal(void)
*--------------------------------------------------------------
* Functional description:
*              SCIF0 Send Break Signal
*--------------------------------------------------------------
* Return Value    : -
* Argument        : -
*--------------------------------------------------------------
* Input           : -
* Output          : -
```

9. Sample Program Listing: "scif.c" (8)

```
*---------------------------------------------------------------
* Notes          : -
****** Function Comment End ***********************************/
void SCIF0_SendBrkSignal(void)
{
 volatile unsigned short u2loop;

 SCSPTR0.BIT.SPB2IO = 1;         /* SPB2DT bit value is output to the SCIF_TXD pin */
 SCSPTR0.BIT.SPB2DT = 0;         /* output data is low-level */

 SCSCR0.BIT.TE = 0;              /* clear TE bit */

 for(u2loop = 0 ; u2loop < 0x87a2 ; u2loop++)       /* 10bit(S+1byte+P+1STOP) interval elapsed? */
 {
   nop();
 }

 SCSPTR0.BIT.SPB2IO = 0;         /* SPB2DT bit value is not output to the SCIF_TXD pin */
 SCSCR0.BIT.TE = 1;              /* set TE bit */
}

#pragma inline_asm(int_responstime_wait)
static void int_responstime_wait(unsigned int wait_time)
{
?0001:
         DT          R4
         BF          ?0001
         NOP
}
```

10. Sample Program Listing: "intprg.c"

```
/*""FILE COMMENT""*********************************************************************
*     System Name :   SH7780 Sample Program
*     File Name   :   intprg.c
*     Version     :   1.00.00
*     Contents    :   SH7780 Initialize Program
*     Model       :   Hitachi_ULSI_Systems SolutionEngine MS7780SE03
*     CPU         :   SH7780
*     Compiler    :   SHC.9.1.00
*     OS          :   none
*
*     note        :   < Caution >
*                     This sample program is provided simply as a reference and
*                     its operation is not guaranteed.
*                     Use this sample program as a technical reference when
*                     developing software.
*
* Copyright (C) 2007 Renesas Technology Corp. All Rights Reserved
*
*     History     :   2007/12/26  ver 1.00.00
*
**************************************************************************************/
#include <machine.h>

/* --- Function Definition(internal) --- */
static void int_responstime_wait(unsigned int wait_time);

/* --- Symbol Definition --- */
#define INTC_RESPONSEWAIT    (0x00000014)       /* INT response wait Pck 5cycle
                        H'14 = (1/Pck*5cyc) / (1/Ick*3cyc) */

/* --- RAM allocation variable declaration --- */

#pragma section IntPRG
```

```
/* H'700 SCIF ch-0 receive error interrupt */
void INT_SCIF0_ERI0(void)
{
  SCIF0_ErrInterrupt();           /* (additional part from Initialize program) */
}
/* H'720 SCIF ch-0 receive FIFO data full or receive data ready interrupt */
void INT_SCIF0_RXI0(void)
{
  SCIF0_RcvInterrupt();           /* (additional part from Initialize program) */
}
/* H'740 SCIF ch-0 break or overrun error interrupt */
void INT_SCIF0_BRI0(void)
{
  SCIF0_BrkInterrupt();           /* (additional part from Initialize program) */
}
```

11. Sample Program Listing: "vecttbl.src"

```
;*""FILE COMMENT""*********************************************************************
;     System Name :   SH7780 Sample Program
;     File Name   :   vecttbl.src
;     Version     :   1.00.00
;     Contents    :   SH7780 Initialize Program
;     Model       :   Hitachi_ULSI_Systems SolutionEngine MS7780SE03
;     CPU         :   SH7780
;     Compiler    :   SHC.9.1.00
;     OS          :   none
;
;     note        :   < Caution >
;                     This sample program is provided simply as a reference and
;                     its operation is not guaranteed.
;                     Use this sample program as a technical reference when
;                     developing software.
;
; Copyright (C) 2007 Renesas Technology Corp. All Rights Reserved
;
;     History     :   2007/12/26  ver 1.00.00
;
;*************************************************************************************/

          .include   "vect.inc"

          .section   VECTTBL,data
                 .export     _RESET_VECTORS

;SCIF-ch0
          ;H'700     ERI0
           .data.b    H'F0        /* (change part from Initialize program) */
          ;H'720     RXI0
           .data.b    H'F0        /* (change part from Initialize program) */
          ;H'740     BRI0
           .data.b    H'F0        /* (change part from Initialize program) */
          ;H'760     TXI0
          .data.b    H'00
```

4. Documents for Reference

- Hardware Manual
  SH7780 Hardware Manual
  The most up-to-date version of this document is available on the Renesas Technology Website.

- Software Manual
  SH-4A Software Manual
  The most up-to-date version of this document is available on the Renesas Technology Website.

Website and Support

Renesas Technology Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Mar.21.08	—	First edition issued

All trademarks and registered trademarks are the property of their respective owners.

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.