Freescale Semiconductor
Application Note

AN2883
Rev. 1.0, 05/2005

Serial Communication Interface as UART on HCS12 MCUs

By Amin Morales
RTAC Americas
Mexico 2005

Introduction

This document is intended to serve as a quick reference for an embedded engineer to get the serial communication interface (SCI) module up and running for any HCS12 MCU. Basic knowledge about the functional description and configuration options will give the user a better understanding on how the SCI module works. This application note provides examples which demonstrate one use of the SCI module for the HCS12 Family of microcontrollers. The examples mentioned are intended to be modified to suit the specific needs for any application.

The example CodeWarrior project files are available as AN2883SW.zip from http://freescale.com.

HCS12 SCI Features

The SCI allows asynchronous serial communications with peripheral devices and other CPUs. Freescale Semiconductor's HCS12 SCI included the following features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit baud rate selection

freescale™ semiconductor

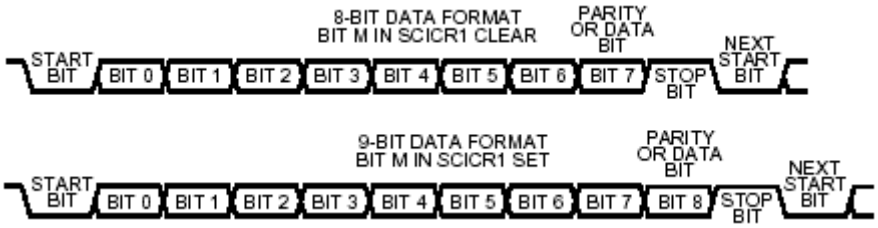
- Programmable 8-bit or 9-bit data format
- Programmable transmitter parity
- Receiver framing error detection
- 1/16 bit-time noise detection
- Interrupt-driven operation
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Receiver overrun
  - Parity error
  - Framing error

Description

The SCI allows full duplex, asynchronous, NRZ serial communication between the CPU and remote devices, including other CPUs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

Data Format

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI control register 1 configures the SCI for 8-bit data characters. Setting the M bit configures the SCI for nine-bit data characters.



Baud Rate Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 0 to 8191 written to the SBR12–SBR0 bits determines the module clock divisor. The baud rate clock is synchronized with the bus clock. The SBR bits are in the SCI baud rate registers (SCIBDH and SCIBDL).

SCI baud rate = SCI module clock / (16 * SCIBR[12:0])

Example

With a module clock = 25 MHz and SBR[12:0] = 41, Baud Rate = 38,109.8

The error of 0.76% because the integer division of the module clock may not give the exact target frequency.

SCI Registers

The memory map for the SCI module is divided into several types of registers: baud rate registers, control registers, and status registers.

Baud Rate Registers (SCIBDH and SCIBDL)

These registers are used to determine the baud rate of the SCI. The baud rate generator is disabled when the baud rate is zero.

Control Registers (SCICR1 and SCICR2)

These registers are used to enable features such as:

- Loop operation
- Low power modes
- Data format — frame of 8 or 9 bits long.
- Parity
- Interrupts enable
- Transmit and receive enable
- Send break character

Status Registers (SCISR1 and SCISR2)

These registers provide inputs to the MCUs for generation of SCI interrupts such as:

- Transmitter empty
- Transmission complete
- Receiver full
- Receiver overrun
- Parity error
- Framing error

Data Registers (SCIDRH and SCIDRL)

Reading these registers accesses SCI the receive data register; writing them accesses the SCI transmit data register. R8 and T8 are the ninth data bit received and transmitted, respectively, when the SCI is configured for 9-bit data format.

In 8-bit data format, only SCIDRL needs to be accessed; however, when transmitting in 9-bit data format, write first to SCIDRH and then to SCIDRL.

For detailed information on SCI registers, refer to the HCS12 SCI block guide, Freescale document number S12SCIV2.

Example Code

The example code explained in this note shows three basic procedures: SCI Configuration, SCI transmission, and SCI reception (SCI transmission and reception are accomplished by the interruption mechanism). The software consists in sending a continuous character string with the title “freescale” through the SCI port. The string can be observed in a PC or other device with a serial terminal. The software can accept data bytes through the reception line on the SCI port. If a character ‘U’ or ‘u’ is received, the character string is uppercased, but if a character ‘L’ or ‘l’ is received, the character string is lowercased.

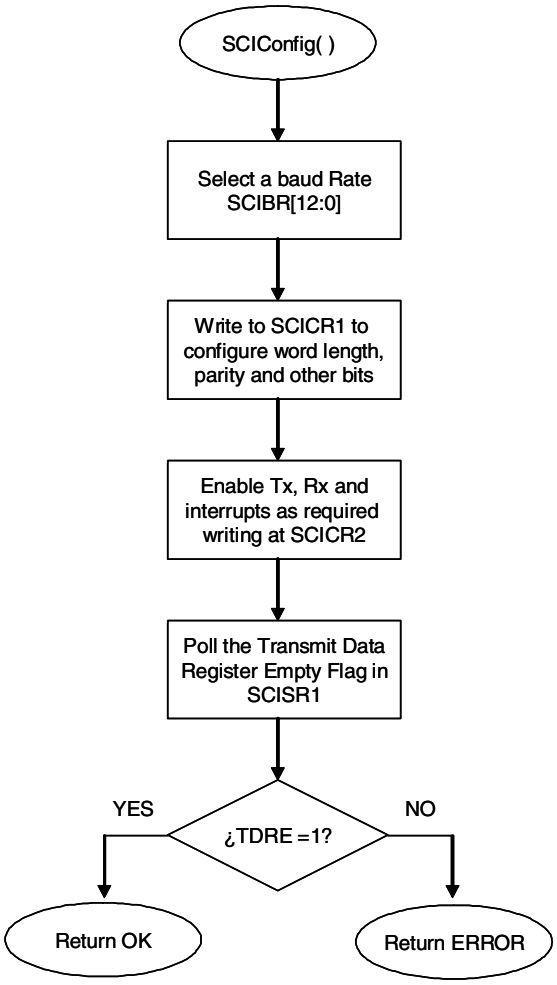


Figure 1. SCI Configuration Flow Diagram

The SCI configuration basically consists of setting the baud rate to 19200 bps with 8-bit data and no parity and then enabling reception and transmission on the SCI port. An 8-MHz external crystal was used to achieve 4 MHz for the SCI module clock.

The following flow diagrams are related to the SCI transmission procedure.

When the first data byte from the SCI data register is received in the transmit shift register, the TDRE flag (transmit data register empty flag) is set, which indicates the SCIDRH:L can receive a new value to transmit and generating an interruption if enabled. The SCI interrupt service routine (ISR) takes over sending the rest of the characters in the string.

A new transmission will not result until the TDRE flag has been cleared. Clear TDRE by reading SCI status register 1 (SCISR1) and then writing to SCI data register low.

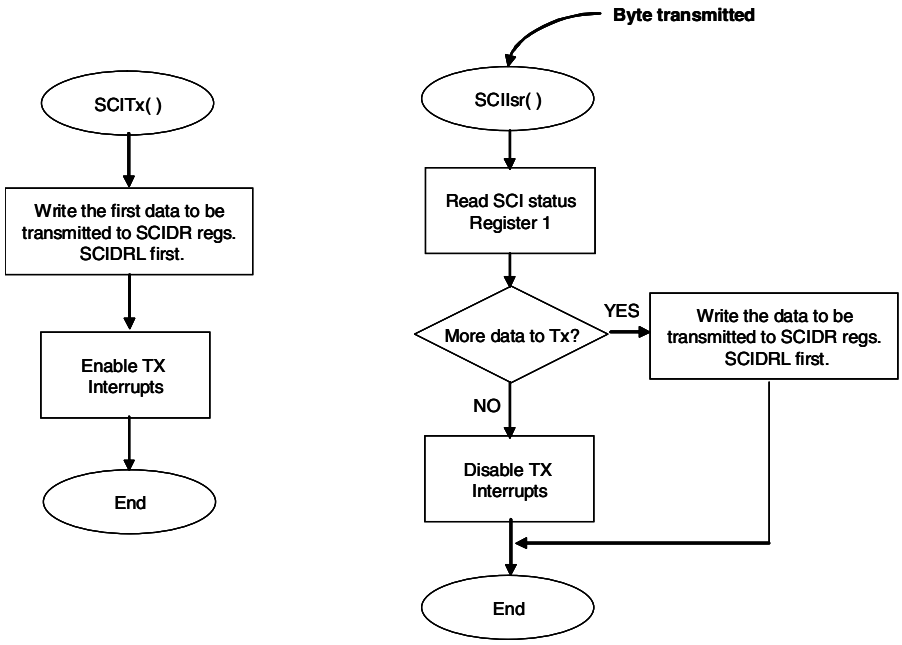


Figure 2. SCI Transmission and Interrupt Service Routine Flow Diagrams

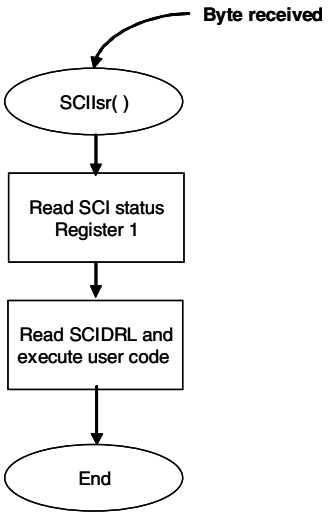


Figure 3. SCI Reception Interrupt Service Routine Flow Diagram

The reception procedure is achieved by clearing the RDRF (receive data register full) flag and executing the user code. In the example code, the conversion from uppercase to lowercase and vice versa is performed in this routine.

An RDRF interrupt indicates that the received data has been transferred to the SCI data register and that byte can now be read by the MCU. The RDRF interrupt is cleared by reading the SCI status register 1 (SCISR1) and then reading SCI data register low.

The screen on the serial terminal will show the string depending upon the selection of upper case or lowercase as it is shown in Figure 4.

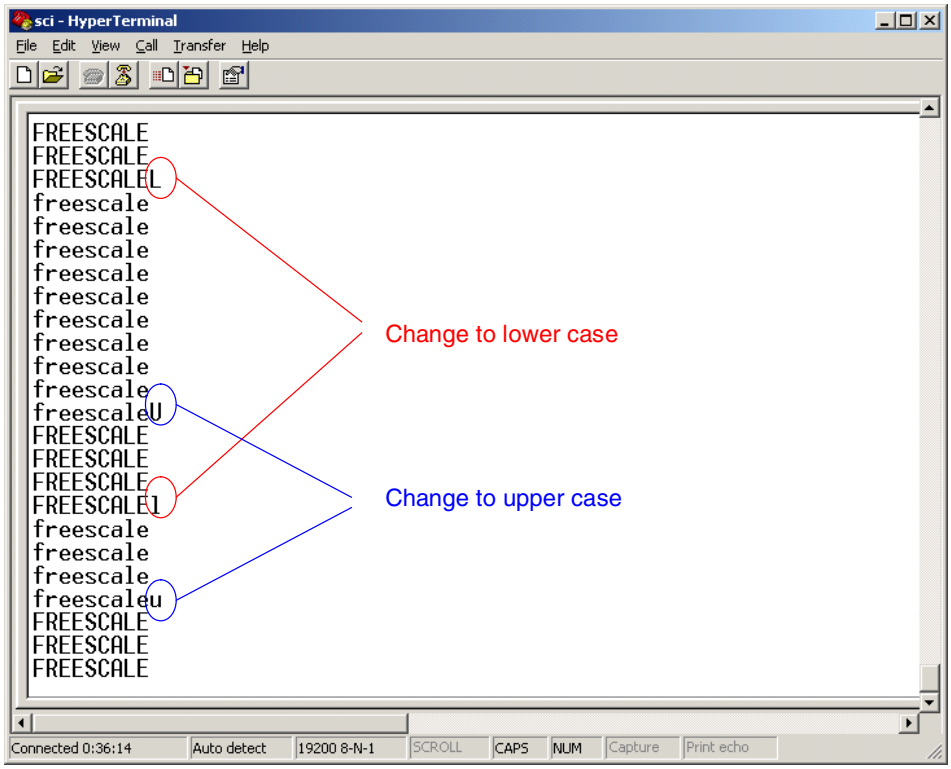


Figure 4. PC Serial Terminal

```
/**
 * Copyright (c) 2004, Freescale Semiconductor
 * Freescale Willy Note
 *
 * File name     : main.c
 * Project name: SCI Demo Software
 *
 * Author          : Amin Morales
 * Department   : RTAC Americas
 *
 * Description    : The SCI Demo Software consists in sending continuously
 *                        a character string with the title "freescale" through the
 *                        SCI port, the string can be visualized in a PC or other device
 *                        with a serial terminal. The software can accept data bytes
 *                        through the reception line on the SCI port, if a character 'U'
 *                        or 'u' is received the character string is uppercased but if
 *                        a character 'L' or 'l' is received the character string is
 *                        lowercased.

 *
 * History       :
 * 08/10/2004 : Release. (A19258)
 */

#include <hidef.h>
```

Example Code

```
/* SCI definitions */
#define SCIBRH   (*((volatile unsigned char*)(0x00C8)))
#define SCIBRL   (*((volatile unsigned char*)(0x00C9)))
#define SCICR1   (*((volatile unsigned char*)(0x00CA)))
#define SCICR2   (*((volatile unsigned char*)(0x00CB)))
#define SCISR1   (*((volatile unsigned char*)(0x00CC)))
#define SCISR2   (*((volatile unsigned char*)(0x00CD)))
#define SCIDRH   (*((volatile unsigned char*)(0x00CE)))
#define SCIDRL   (*((volatile unsigned char*)(0x00CF)))

/* ERROR code and STATUS definitions */
#define ERROR_OK    1
#define ERROR_ERROR 0
#define START_CYCLE 1
#define WAIT_CYCLE  0

/*Global variables*/
unsigned char SCIIniTx;
unsigned char SCIString[12]={'F','R','E','E','S','C','A','L','E',0xa,0xd,'\0'};
unsigned char *SCIStringp;
unsigned char Stringcase;

#pragma CODE_SEG __NEAR_SEG NON_BANKED
/*
 * SCIIsr: Interrupt Service routine for the SCI module
 *  Clear TDRE and RDRF flags
 *  In transmission sends a character string until NULL character
 *  In reception changes from uppercase to lowercase as required
 *
 * Parameters: None
 *
 * Return : None
 */
interrupt void SCIIsr(void) {

    if (SCISR1 & 0x80){    /*If transmission flag is set*/
        SCISR1;
        if(*(SCIStringp++) != '\0'){
            if(*SCIStringp > 0xD){     /*Avoid to change CR and LF characters*/
                SCIDRL=*SCIStringp + Stringcase;
            }
            else{
                SCIDRL=*SCIStringp;
            }
        }
        else{
            SCIIniTx=START_CYCLE;       /*Start new transmission cycle*/
            SCICR2 &= 0x7F;                    /*Disable TDRE interrupt*/

        }
    }

    if(SCISR1 & 0x20){    /*If reception flag is set*/
        SCISR1;
        if(SCIDRL == 'U' || SCIDRL == 'u'){
            Stringcase = 0x00;          /*Uppercase the character string*/
        }
```

```
        else if(SCIDRL == 'L' || SCIDRL == 'l'){
            Stringcase = 0x20;          /*Lowercase the character string*/
        }
    }

    return;
}

#pragma CODE_SEG DEFAULT
/*
 * SCIConfig: Configures SCI port at 19200 bps, 8 data bits, no parity,
 *  enable transmission, reception and RDRF interrupt
 *
 * Parameters: None
 *
 * Return : Error code
 */
unsigned char SCIConfig(void){

    SCIBRL = 0x0D;    /*Configure baud rate at 19200 bps with*/
    SCIBRH = 0x00;    /*an SCI clock modulo of 4MHz*/

    SCICR1 = 0x00;    /*8 data bits, no parity*/
    SCICR2 = 0x2C;    /*Enable Tx, Rx, and RDRF interrupt*/

    if (SCISR1 & 0x80){       /*Poll TDRE flag*/
        return ERROR_OK;   /*TDRE set, return OK*/
    }
    else{
        return ERROR_ERROR; /*TDRE clear, return ERROR*/
    }
}

/*
 * SCITx: Write data byte to SCIDRL register to transmission and
 *  enable TDRE interrrupt.
 *
 * Parameters: SCIByte
 *
 * Return : None
 */

void SCITx(unsigned char SCIByte){

    SCIDRL = SCIByte;                       /*Write data byte to SCIDRL register*/
    SCICR2 |= 0x80;        /*Enable TDRE interrupt*/
}

unsigned char main(void){

    if (SCIConfig())         /*Configure SCI port*/
       ;
    else{
        return ERROR_ERROR;
    }

    EnableInterrupts;
```

```
    SCIIniTx = START_CYCLE;  /*Initialize transmission cycle flag*/

    for (;;){
        if(SCIIniTx == START_CYCLE){
            SCIIniTx = WAIT_CYCLE;

            SCIStringp=SCIString;        /*Set pointer to character string*/
            SCITx(*SCIStringp + Stringcase);/*Send first byte of string*/
        }

    }
    return ERROR_OK;
}
```

Conclusion

The SCI module is available in all HCS12 derivatives available at the time of publish: A, B, C, D, DB, DJ, DG, DP, DT, E, H, KG, KT, NE. The example code above shows an easy way to configure the SCI and the routines to transmission and reception of characters by the interruption mechanism.

Considerations and References

Find these and other useful resources on the Freescale Semiconductor home page: http://www.freescale.com.

- Download the companion software file, AN2880SW.zip, from freescale.com; it may be modified to satisfy the requirements of a specific application.
- To learn about SCI operation in low power modes, consult the specific device data sheet.
- MC9S12DJ256 derivative was used to generate the ***SCI DemoSoftware***.
- The ***SCI DemoSoftware*** code was developed in CodeWarrior 12 version 3.1.
- Refer to HCS12 Serial Communications Interface (SCI) Block Guide, document number **S12SCIV2/D**, for more information on SCI.

This page is intentionally blank.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN2883
Rev. 1.0, 05/2005

freescale™
semiconductor