Tutorial Introduction

PURPOSE
- To explain how to configure and use the Serial Communications Interface (SCI) Module in common applications

OBJECTIVES:
- Describe the uses and features of the Serial Communications Interface Module.
- Identify the steps to configure the SCI transmitter.
- Identify the steps to configure the SCI receiver.
- Identify the steps to set up serial communications from the MC68HC08 to the host PC.

CONTENTS:
- 22 pages
- 2 questions

LEARNING TIME:
- 40 minutes

PREREQUESITE:
- 68HC08 CPU training module

Welcome to this tutorial on the 68HC08 Serial Communications Interface (SCI) Module. This tutorial describes the features and configuration of the SCI Module. Please note that on subsequent pages, you will find reference buttons in the upper right of the content frame that access additional content.

Upon completion of this tutorial, you'll be able to describe the uses and features of the SCI Module. You'll also be able to configure the SCI for data transmission and reception and to set up serial communications from the MC68HC08 to the host PC.

The recommended prerequisite for this tutorial is the 68HC08 CPU training module. Click the Forward arrow when you're ready to begin the tutorial.

SCI Module Features

- **Full-duplex high speed asynchronous operation**
- **Standard mark/space NRZ format**
- **32 programmable baud rates**
- **Programmable 8-bit or 9-bit character length**
- **Two receiver wakeup methods:**
  - **Address mark wakeup**
  - **Idle line wakeup**
- **Separately enabled, double-buffered transmitter and receiver**
- **Advanced data sampling and re-synchronization logic**
- **Programmable transmitter output polarity**
- **Eight interrupt sources**

Let's begin this tutorial with a discussion of the SCI Module features.

The SCI Module, also referred to as a Universal Asynchronous Receiver Transmiter (UART), provides full-duplex, high-speed, asynchronous communications with peripheral devices or other MCUs. The SCI uses standard nonreturn-to-zero (NRZ) format and includes an on-chip baud rate generator. The SCI baud rate generator doesn't require the 68HC08 timer system. Instead, it derives one of 32 standard baud-rate frequencies directly from the MCU oscillator.

You can select a character length of eight or nine bits. The most common configuration uses one start bit, 8-bits of character data, and one stop bit. A nine bit configuration uses one start bit, 9-bits of character data, and one stop bit. The ninth bit of character data can be used for an extra stop bit, the SCI receiver wake-up function, or a parity bit.

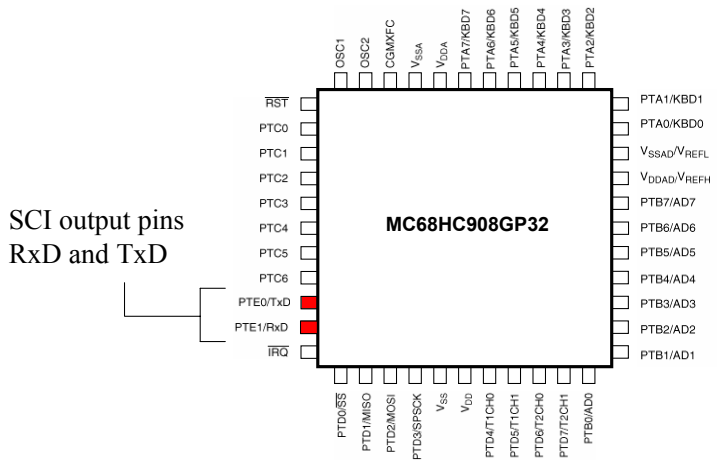
Two receiver wakeup methods are provided: address mark wakeup and idle line wakeup. In multiple-receiver systems, this feature allows the SCI Module to ignore transmissions intended for other receivers by entering a standby mode.

The transmitter and receiver are enabled separately and are functionally independent, though they use the same data format and baud rate. Both the transmitter and receiver are double-buffered. This means that back-to-back characters are handled easily, even if the CPU is delayed in responding to the completion of a character. In addition, the SCI receiver includes a number of advanced features to assure highly reliable data reception. These features assist in the development of efficient communication networks.

You can program the transmitter output polarity by setting a bit in one of the SCI control registers. This feature allows you to transmit in plain TTL levels. Some devices translate TTL logic levels into standard RS-232 data levels. Such devices require a plain signal polarity on which a logic 0 represents 0V and a logic one represents a voltage above 3V.

The SCI can be interrupt driven with eight flags and interrupt requests. The SCI provides separate interrupt requests and vectors for the transmitter, the receiver, and for error conditions. This provides highly efficient interrupt processing of normal trasmit/receive functions without polling or interrupt checking. Any error conditions can be handled by a separate interrupt service routine.

SCI Module Pins



The figure shows the pin assignment of the 68HC908GP32 MCU 44-pin quad flat pack (QFP).

The SCI Module uses two I/O pins. The RxD pin is an input pin used to receive data and the TxD pin is an output pin used to transmit data. These two pins are shared with the MCU parallel I/O port pins. When the SCI is disabled, the two pins can be used for general purpose I/O.

If your application requires RS-232 or RS-422 levels, you need to provide external logic level shifter buffers to translate from the MCU logic levels to the required logic levels. A typical translation is from 0 to 3 or 5V logic levels to +/- 12V logic levels.

SCI Register Summary

- SCI data register (SCDR)
- SCI baud rate register (SCBR)
- SCI control register 1 (SCC1)
- SCI control register 2 (SCC2)
- SCI control register 3 (SCC3)
- SCI status register 1 (SCS1)
- SCI status register 2 (SCS2)

The SCI Module register set includes several registers to control and monitor SCI operations, including a data register, a baud rate register, three control registers, and two status registers.

Let's take a look at each register, beginning with a discussion of how the SCI transmits and receives data using the SCI data register. Then we'll look at how the baud rate is calculated using the information in the SCI baud rate register. We'll finish the discussion with a detailed review of the SCI status and control registers.

SCI Data Register (SCDR)

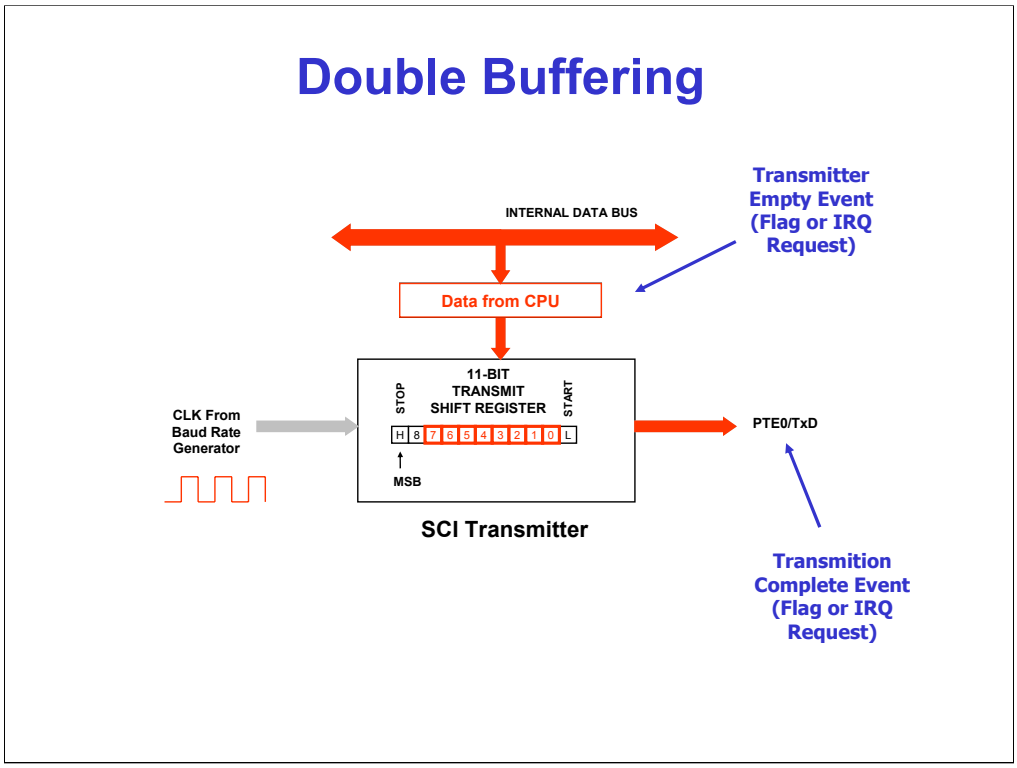
Address: $0018

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
Reset:	Unaffected by reset							

Reading SCDR accesses R7:R0.

Writing SCDR accesses T7:T0.

The SCI data register, SCDR, is actually two registers that you access with one address. The two registers are the receive data register and the transmit data register. You access the individual registers using the read/write commands. Reading the SCI data register reads data from the receive data register. Writing to the SCI data register writes data to the transmit data register.



The SCI Module includes a double-buffered transmitter and receiver. Let's look at how this feature is implemented in the SCI transmitter.

To transmit data, the CPU writes data to the SCI data register. In addition to the data register, the SCI transmitter includes an 11-bit shift register for storing the data during transmission. This allows the CPU to write a character of data while the SCI is transmitting the previous character of data. In order to transmit data, the CPU only needs to check that the SCI data register is empty before writing to it.

Let's look at an example of a typical data transmission.

Let's assume that the SCI data register is empty and the CPU wants to transmit a character through the SCI transmitter. The CPU writes the character to the SCI data register.

When the shift register is ready to accept new data, the SCI hardware automatically moves the data from the SCI data register to the shift register.

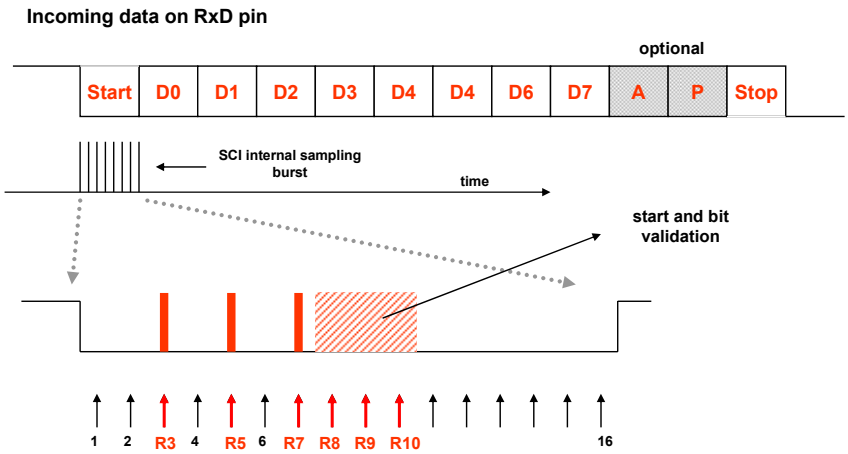
Once the data is moved to the shift register, the SCI sets the transmitter empty flag to indicate to the CPU that another character can be sent. You can use software to poll the flag or you can configure the SCI to generate an optional interrupt request.

The shift register is clocked by the auto baud generator, shifting the 11-bits out to the TxD pin.

Once the last bit is shifted, the SCI sets the transmission complete flag and optionally generates an interrupt request for CPU processing.

The SCI receiver also uses a double-buffer to receive data.

Data Receiver Recovery



The SCI receiver includes a data recovery system that samples and verifies the data. These features assure highly reliable data reception. Let's take a detailed look at how the receiver handles an asynchronous bit stream coming into the RxD pin.

The data recovery system samples the incoming bit stream at a fixed clock rate, called the RT clock rate, that is 16 times the baud rate. The RT clock rate is synchronized after every start bit and any data bit that changes from logic 1 to logic 0. The additional synchronization makes the SCI module less sensitive to differences in the baud rates of the sending device and the MCU.

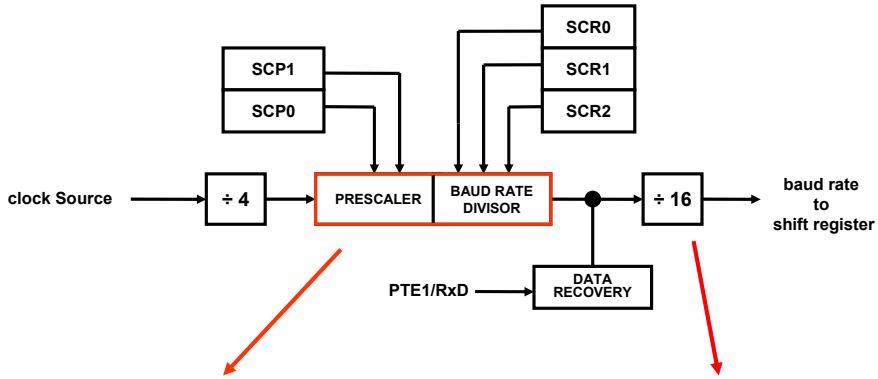
To detect a start bit, the SCI receiver hardware is continually executing an asynchronous search for a logic 0 preceded by three logic 1 values. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16 and the sampling takes place.

This sampling technique allows the SCI to detect possible interference or noise overlapping with the incoming data stream. To verify the integrity of the start bit and to detect noise, the data recovery logic takes samples RT3, RT5 and RT7.

The start bit verification fails if any two of the three verification samples have a value of logic 1. When the verification fails, the RT clock is reset and the search begins again. To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10.

A similar situation occurs when the SCI receiver expects a stop bit. In this case, the recovery logic takes samples RT8, RT9 and RT10 to verify the bit integrity and to detect noise.

Baud Rate Generator



SCI baud rate register (SCBR)

$$\text{baud rate} = \frac{\text{clock source}}{\text{prescaler} \bullet \text{divisor} \bullet 4 \bullet 16}$$

Next, let's look at how the SCI baud rate is calculated.

The figure shows the blocks and signals for the SCI baud rate generator. The baud rate generator includes a prescaler and a baud rate divisor that you can program using the SCI baud rate register.

The SCI baud rate generator clock source varies based on the device. The 68HC08 SCI clock source can be the CGMXCLK crystal frequency divided by four or it could be the internal bus frequency in devices with a PLL or internal clock generator. Check the device technical data book for the specific SCI implementation.

Notice the by-4 divider and the by-16 divider. The by-4 divider at the clock input is for fetch cycle purposes. The by-16 divider that goes to the shift register allows the receiver to perform data sampling at 16 times the baud rate speed.

To calculate the SCI baud rate, divide the clock source by the product of the prescaler, the baud rate divisor, the by-4 divider, and the by-16 divider.

SCI Baud Rate Register (SCBR)

Address: $0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:			SCP1	SCP0	R	SCR2	SCR1	SCR0
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

SCP1 - SCP0	prescaler
00	1
01	3
10	4
11	13

SCR2 - SCR0	baud rate divisor
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

You can program the prescaler and the baud divider using the SCI baud rate register, SCBR. You should select the values appropriate for your application.

The SCI baud rate prescaler bits, SCP1 and SCP0, select one of four prescaler values as shown in the table.

The SCI baud rate select bits, SCR2-SCR0, select one of eight baud rate divisors as shown in the table.

Next, let's look at an example of how to use these tables to select a specific baud rate.

Selecting the Baud Rate

Generate a baud rate of 19200 baud:

$$\text{baud rate} = \frac{\text{Clock Source}}{\text{prescaler} \bullet \text{divisor} \bullet 4 \bullet 16}$$

$$= \frac{4.9152\ \text{MHz}}{1 \bullet 4 \bullet 256} = 19200\ \text{baud}$$

SCP1 - SCP0	prescaler
00	1
01	3
10	4
11	13

SCR2 - SCR0	baud rate divisor
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Let's generate a baud rate of 19200 baud assuming a clock source frequency of 4.9152 MHz.

The baud rate is the clock source divided by the product of the prescaler, the baud rate divisor, the by-4 divider, and the by-16 divider. To program the baud rate generator, select the prescaler and baud rate divisor values that produce 19200 baud. In this case, we'll select a prescaler value of 1 and a baud rate divisor of 4.

SCI Control Registers Overview

		Bit 7	6	5	4	3	2	1	Bit 0
SCC1	Read: / Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
	Reset:	0	0	0	0	0	0	0	0
SCC2	Read: / Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Reset:	0	0	0	0	0	0	0	0
SCC3	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
	Write:								
	Reset:	U	U	0	0	0	0	0	0

= Unimplemented U = Unaffected

Let's continue our discussion of the SCI register set, beginning with the three SCI control registers SCC1, SCC2, and SCC3.

You can configure standard SCI operations using bits 7-0 in SCC1, bits 3-0 in SCC2, and bits 7-4 in SCC3.

You can also configure the SCI to generate eight different interrupts using bits 7-4 of SCC2 and bits 3-0 in SCC3.

We'll first look at how to configure the SCI for standard operations and then follow-up with a discussion of the eight SCI interrupt sources.

Configuring SCI Operations - SCC1

Address: $0013

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
Reset:	0	0	0	0	0	0	0	0

LOOPS — Loop Mode Select Bit

1 = Loop mode enabled
0 = Normal operation enabled

The SCC1 register contains eight read/write control bits that are cleared on reset. All eight bits configure standard SCI operations.

The loop mode select bit, LOOPS, enables loop mode operation. This mode is used primarily for testing and diagnostics. In loop mode, the RxD pin is disconnected from the SCI and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode.

The enable SCI bit, ENSCI, enables the SCI and the SCI baud rate generator.

Setting the transmit inversion bit, TXINV, to 1 reverses the polarity of transmitted data. Note that setting the TXINV bit inverts all transmitted values, including idle, break, and start/stop bits.

The character length mode bit, M, selects the SCI character length. You can select an eight-bit or nine-bit configuration. In the nine-bit configuration, the ninth bit can serve as an extra stop bit, a receiver wakeup signal, or as a parity bit.

The wakeup condition bit, WAKE, selects either address mark or idle line wake-up. The SCI receiver can enter a temporary stand-by wakeup mode to ignore messages intended for a different receiver. If address mark is selected, a logic 1 in the most significant bit position of a received character wakes up the SCI in time to see the first character of the next message. This is typically used to greatly reduce CPU overhead in multi-drop SCI networks. In idle line mode, the receiver detects when no other SCI is transmitting by monitoring the RxD pin.

The idle line type bit, ILTY, selects the point in the bit stream that the SCI begins counting logic 1 bits as idle character bits. With the ILTY bit to 1, the SCI begins counting after the stop bit. With the ILTY bit to 0, the SCI begins counting after the start bit.

The parity enable bit, PEN, enables the SCI parity function. When enabled, the parity function inserts a parity bit in the most significant bit position.

The parity bit, PTY, selects the type of parity, odd or even, that the SCI generates and checks for. Setting the bit to 1 selects odd parity and setting the bit to 0 selects even parity.

Configuring SCI Operations - SCC2

Address: $0014

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0

TE - Transmitter Enable Bit

1 = Transmitter enabled
0 = Transmitter disabled

The SCC2 register contains eight read/write control bits that are cleared on reset. Bits 3-0 configure standard SCI operations.

Setting the transmitter enable bit, TE, begins the transmission by sending a preamble of 10 or 11 logic 1 bits from the transmit shift register to the TxD pin. Clearing the TE bit will finish any current transmission before returning to the idle logic 1 condition.

Setting the receiver enable bit, RE, enables the receiver. Clearing the RE bit disables the receiver but does not affect the receiver interrupt flag bits.

When the receiver wakeup bit, RWU, is set to 1, the receiver is put in stand-by mode. When the receiver is in stand-by mode, receiver interrupts are disabled. The WAKE bit in SCC1 determines whether an address mark or an idle input brings the receiver out of stand-by mode. When the receiver is brought out of stand-by mode, the RWU bit is cleared.

When the send break bit, SBK, is set to 1, the SCI transmits a break character followed by a logic 1 bit. This guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no logic 1 bits between them.

Configuring SCI Operations - SCC3

Address: $0015

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
Write:								
Reset:	U	U	0	0	0	0	0	0

= Unimplemented U = Unaffected

R8 — Received Bit 8

When receiving 9-bit characters, R8 is the ninth bit (bit 8) of the character.

When receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7) of the character.

The SCC3 register contains eight read/write control bits that are cleared on reset. Bits 7-4 configure standard SCI operations.

The R8 bit stores the received bit 8 when the SCI is receiving 9-bit characters (MSB 8-LSB 0). R8 is received at the same time that SCDR receives the other 8 bits.

The T8 bit is very similar to the R8 bit. T8 is used when the SCI is transmitting 9-bit characters.

DMARE and DMATE are used on MCUs with DMA to enable DMA transfers and DMA receiver services. Note that the DMA module is not included in the 68HC08 MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

Configuring SCI Interrupts

		Bit 7	6	5	4	3	2	1	Bit 0
SCC2	Read: / Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Reset:	0	0	0	0	0	0	0	0
SCC3	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
	Write:								
	Reset:	U	U	0	0	0	0	0	0

= Unimplemented U = Unaffected

The remaining bits in SCC2 and SCC3 configure the SCI to generate eight different CPU interrupt requests when the corresponding status bits are set in the SCI status register SCS1. We'll refer to these control bits when we discuss the status registers. Note that you can view a detailed description of these bits using the Register Maps button in the upper right of the content frame.

SCI Status Register 1 (SCS1)

Address: $0016

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
Write:								
Reset:	1	1	0	0	0	0	0	0

= Unimplemented

SCTE — SCI Transmitter Empty Bit

1 = SCDR data transferred to transmit shift register
0 = SCDR data not transferred to transmit shift register

The SCI status register, SCS1, is a read-only register that contains eight status flags for monitoring SCI operations.0

The SCI transmitter empty bit, SCTE, is set by the transmitter when the SCDR transfers a character to the transmit shift register. Setting the SCTE bit will generate an interrupt if the SCTIE bit in SCC2 is set.

The transmission complete bit, TC, is set when the SCTE bit is set and no data, preamble, or break character is being transmitted. Setting the TC bit will generate an interrupt if the TCIE bit in SCC2 is set.

The SCI receiver full bit, SCRF, is set when the data in the receive shift register transfers to the SCDR. Setting the SCRF bit will generate an interrupt if the SCRIE in SCC2 is set.

The receiver idle bit, IDLE, is set when 10 or 11 consecutive logic 1 values appear on the receiver input. Setting the IDLE bit will generate an interrupt if the ILIE bit in SCC2 is set.

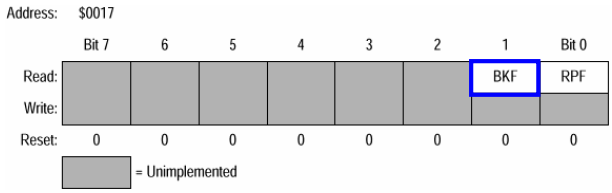
The receiver overrun bit, OR, is set when the software fails to read the SCDR prior to the receive shift register receiving the next character. When this happens, the data in the shift register is lost, but the data already in the SCDR is not affected. Setting the OR bit will generate an interrupt if the ORIE in SCC3 is set.

The receiver noise flag bit, NF, is set when the SCI detects noise on the receiver input pin (RxD pin). Setting the NF bit will generate an interrupt if the NEIE bit in SCC3 is set.

The receiver framing error bit, FE, is set when a logic 0 is accepted as the stop bit. To clear the the FE bit, read SCS1 with FE set and then read SCDR. Setting the FE bit will generate an interrupt if the FEIE bit in SCC3 is set.

The receiver parity error bit, PE, is set when the SCI detects a parity error in the incoming data. To Clear the PE bit, read SCS1 with PE set and then read SCDR. Setting the PE bit will generate an interrupt if the PEIE bit in SCC3 is set.

SCI Status Register 2 (SCS2)



BKF — Break Flag Bit

1 = Break character detected
0 = No break character detected

The SCI status register, SCS2, is a read-only register that contains two status flags for monitoring SCI operations.

The break flag, BKF, is set when the SCI detects a break character on the receiver input pin (PTE1/RxD). The BKF bit doesn't generate an interrupt request.

The reception in progress flag, RPF, is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. The RPF bit doesn't generate an interrupt request.

Now that we've discussed how to configure the SCI, let's look at an example of how to set up serial communications from the MC68HC08 to the host PC. We'll look at two example programs. The first program configures the SCI for transmission, and the second program configures the SCI for reception.

Example: Transmission Program

Transmit "Hello World" at 9600,n,8,1 to a terminal:

```
$include 'iogp20.inc'

            ORG     $FFFE
            FDB     Application             ;Reset Vector

            ORG     $B000
Application:
            MOV     #$0B,CONFIG1 ;No COP, 5V Operation, No Low voltage detection
            MOV     #$02,CONFIG2 ;Use Internal data bus clock as clock source for SCI

            MOV     #%00000011,SCBR         ;Select 9600 Baud rate over a clock = 4.9152 MHz
            MOV     #%01000000,SCC1         ;Enable SCI
            MOV     #%00001000,SCC2         ;Transmit enable, Disable receiver
            LDA     SCS1                    ;Condition to clear SCT Tx empty bit
Main:
            MOV     #'H',SCDR               ;Send "H"
            BRCLR   7,SCS1,*                ;Wait for TX
            MOV     #'e',SCDR               ;Send "e"
            BRCLR   7,SCS1,*                ;Wait for TX
            MOV     #'l',SCDR               ;Send "l"
            BRCLR   7,SCS1,*                ;Wait for TX
            MOV     #'l',SCDR               ;Send "l"
            BRCLR   7,SCS1,*                ;Wait for TX
            MOV     #'o',SCDR               ;Send "o"
            BRCLR   7,SCS1,*                ;Wait for TX
            MOV     #$0013,SCDR             ;Send <CR>
            BRCLR   7,SCS1,*                ;Wait for TX
            JMP     MAIN
```

This program configures the SCI for data transmission. The SCI transmitter is enabled and the SCI receiver is disabled. The baud rate is set to 9600 baud. The program simply sends the word "Hello" to a terminal or any other device capable of understanding asynchronous communications.

The main program contains a series of MOV instructions for sending the characters through the serial port. After each MOV instruction, the program must wait until the transmit shift register (Tx) moves the next byte to be transmitted from the SCDR. Recall that the SCDR is a one-byte buffer between the SCI and the CPU. When the SCDR transfers a character to the transmit shift register, the SCTE flag in SCS1 is set. The BRCLR instruction that follows each MOV instruction waits for this to happen.

Example: Reception Program

Receive a "4" to turn off/on a LED:

```
$include 'iogp20.inc'

            ORG     $FFFE
            FDB     Application             ;Reset Vector

            ORG     EPROM
Application:
            MOV     #$0B,CONFIG1            ;No COP, 5V Operation, No Low voltage detection
            MOV     #$02,CONFIG2            ;Use Internal data bus clock as clock source for SCI

            MOV     #%00000011,SCBR         ;Select 9600 Baud rate over a clock = 4.9152 MHz
            MOV     #%01000000,SCC1         ;Enable SCI
            MOV     #%00100100,SCC2         ;Receiver enable, Disable transmitter
            LDA     SCS1                    ;Condition to clear SCT Tx empty bit

            CLR     PORTC                   ;Set PORTC = 0x00
            BSET    4,DDRC                  ;Set PORTC  Bit 4 as Output.
            CLI
Main:       JMP     MAIN                    ;Application Loop

RxEvent:
            LDA     SCS1                    ;Clear Rx flag
            LDA     SCDR                    ;Read incoming char
            CMP     #'4'                    ;IF Acc = ASCII(4) Then Complement PORTC output contents
            BNE     EndIRQ                  ;ELSE Exit IRQ
            COM     PORTC                   ;Complement PORTC output contents
EndIRQ      RTI                             ;End this IRQ

            ORG     $FFE4                   ;Set SCI Receiver full Event  (a New character has arrive)
            DW      RxEvent                 ;Define Interrupt Service Routine in IRQ Vector Table
```

This program configures the SCI for data reception. The SCI receiver is enabled and the SCI transmitter is disabled. The program waits for a character to be received. When this happens, the main application loop is notified by the interrupt service routine RxEvent.

The interrupt service routine instructs the CPU to read the input buffer, SCDR, and compare it with the ascii code of 4. If it matches, the CPU complements the contents of PORTC. This toggles the LED connected to PORTC to 4.

Question

What is the SCI baud rate when the clock frequency is 4.9152 MHz, the prescaler select bits are set to 01, and the baud rate select bits are set to 001? Click on your choice.

a) 3200 baud

b) 12,800 baud

c) 19,200 baud

d) 25,600 baud

Let's complete this tutorial with a few questions to check your understanding of the material.

What is the SCI baud rate when the clock frequency is 4.9152 MHz, the prescaler select bits are set to 01, and the baud rate select bits are set to 001? Click on your choice.

Answer: To calculate the SCI baud rate, divide the clock source by the product of the prescaler, the baud rate divisor, the by-4 divider, and the by-16 divider. The prescaler is set to a value of 3 and the baud rate divisor is set to a value of 2. Therefore, the baud rate is 4.9152 MHz divided by the product of 3, 2, 4, and 16, or 384. This yields a baud rate of 12,800 baud.

Question

Which of the following instructions configures the SCI to generate both a noise error interrupt and a parity error interrupt? Click on your choice.

a) MOV #%01010000, SCC3
b) MOV #%00000101, SCS1
c) MOV #%10000001, SCC2
d) MOV #%00000101, SCC3

Which of the following instructions configures the SCI to generate both a noise error interrupt and a parity error interrupt? Click on your choice.

Answer: These two interrupts are configured in SCC3. The noise error interrupt enable bit, NEIE, is bit 2 of SCC3. The parity error interrupt enable bit, PEIE, is bit 0 of SCC3. Therefore, the correct instruction is choice d.

Tutorial Completion

- SCI Uses and Features
- SCI Configuration
- SCI Transmission
- SCI Reception

In this tutorial, you've learned about the features of the Serial Communications Interface Module. You've also learned how to configure the SCI for data transmission and reception, including an example to set up serial communications between the MC68HC08 and the host PC.