HC08 SCI Operation with Various Input Clocks

By Rick Cramer
CSIC MCU Product Engineering
Austin, Texas

INTRODUCTION

This application note describes the operation of the serial communications interface (SCI) as it relates to the MC68HC708XL36 (XL36) with clock generation module A (CGMA). Specifically, the information here provides an analysis of the effects of the input clock on the SCI baud rate. SCI communication in various hardware applications is also examined, as well as code segments and listings.

GENERAL SCI INFORMATION

What is SCI?

SCI is a defined standard for transmitting and receiving data. Serial refers to the method used to transfer data. This method uses one wire to transfer a data byte serially, that is, one bit at a time. Data also can be transferred by a parallel method. A parallel data transfer uses eight wires to send eight bits of a byte of data simultaneously. **Figure 1** shows the serial hardware configuration, and **Figure 2** shows the parallel.

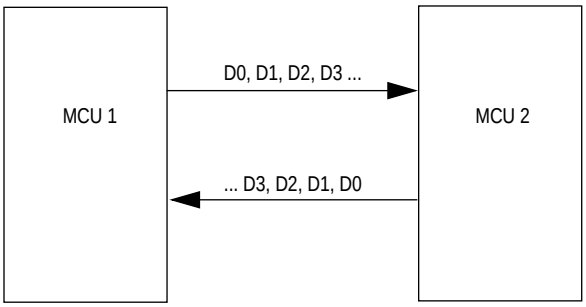

Figure 1. Serial Hardware Configuration

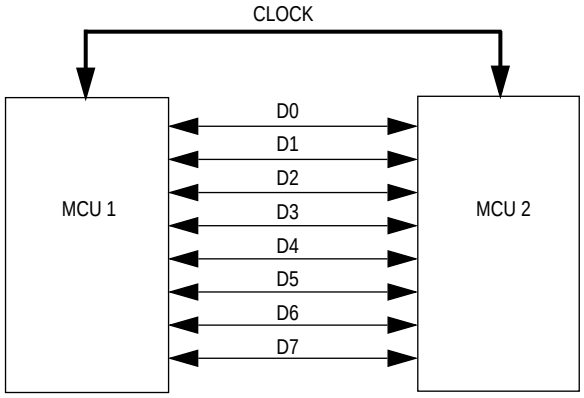

Figure 2. Parallel Hardware Configuration

Serial communication has an advantage over parallel data transfer because serial uses fewer connections which results in a cheaper manufacturing cost. The drawback of using serial, however, is that this type of communication generally makes a slower transfer than parallel.

The SCI is used in most applications to communicate with other microcontroller units (MCU) over a 2-wire system. Using two wires allows full duplex operation. Essentially, both MCUs can transmit information to each other simultaneously. **Figure 1** shows full duplex operation.

Baud Rate

Baud rate is the rate in which the data is transferred. Baud rate also conveys the capacity of a data channel and is expressed most commonly as bits per second (BPS). Although not exactly a correct definition, BPS is often used synonymously with baud, and, for the purpose of this document, they are considered the same. For example, if an SCI system is set to run at 1200 baud, 1200 bits of data can be transferred in one second. This is a measure of speed and capacity. The bit period, which is the length of time required to transfer one bit of data, is calculated by dividing one second by the baud rate. Therefore, the bit rate for a 1200-baud system is 8.334×10^{-4}s, or 834 µs.

Input Clock

The input clock is a square wave that determines the chip speed. Ideally, the input clock could be set to any frequency, and the chip could run as fast as it is driven. Realistically, however, the MCU can run only within specific limits. Consult the appropriate technical data sheet for the device in use to determine its maximum and minimum input clock frequencies. Typically for the XL36, the input clock frequencies range from 4 MHz to 8 MHz. The input clock in the XL36 using CGMA is referred to as CGMXCLK.

How does the input clock affect the baud rate?

The input clock is related directly to the baud rate. This relationship is defined by the baud rate equation in **Figure 3**.

$$\text{baud rate} = \frac{\text{CGMXCLK}}{64 \times \text{PD} \times \text{BD}}$$

Figure 3. Baud Rate Equation

In this equation, PD is the prescaler divisor and BD is the baud rate divisor. PD and BD are variables that allow the user to change the baud rate; however, the baud rate continues to be directly related to GCMXCLK. So when the CGMXCLK frequency is increased, the baud rate also increases.

Why are two communicating devices required to have the same baud rate?

When using the SCI system, no clock signal is sent with the data. A clock signal normally is used to tell the receiving device when the bits of data should be read. Therefore, since no clock is sent with the SCI data, the receiver must determine when each data bit is valid to read. This is done with critical timing. The receiver marks the time when the first bit starts and then reads each following bit at the time they are supposed to be valid at the receiver. If the receiver and transmitter are attempting to communicate at different baud rates, the receiver will misread the incoming data, and errors will occur.

BASIC SCI OPERATION WITH EQUAL INPUT CLOCK FREQUENCIES

The SCI operates easily under ideal conditions, such as these:

- Using two identical HC08 microcontrollers
- Operating both microcontrollers at the same input clock frequency
- Placing both microcontrollers adjacent to each other
- Allowing enough time in the application for a complete SCI transfer
- Accepting the default values of SCI parameters
- Operating the circuit at room temperature

An important point to realize is that the simplest way to use HC08 microcontrollers in unison is to leave most of the registers at their default values. Since all XL36 microcontrollers have the same default value, assuming that they will operate together is a fair assumption. Seven registers are associated with the SCI operation on the XL36, but only three are needed. **Figure 4** illustrates the process of using the SCI to transmit data.

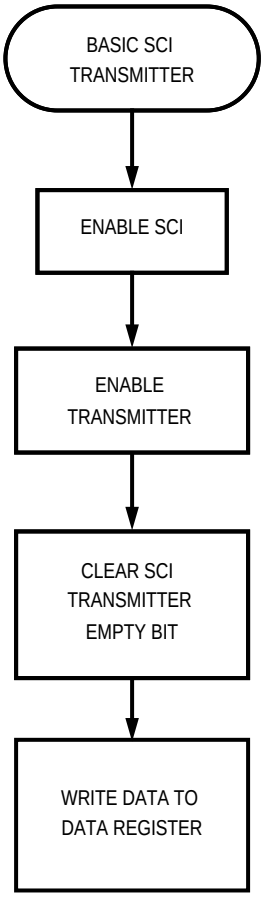

Figure 4. SCI Transmitter Operation

The flowchart in **Figure 4** illustrates the basic operation of how the SCI transmits data. However, some further explanations and a software code example are necessary. **Appendix A: Basic SCI Transmitter Code** shows the implementation of the basic SCI operation. This code is mostly academic. For the code to be of any actual value, the transmitted code must be received. Several variables are defined at the beginning of the program. Notice that most variables are not used in the program code but are included to provide a complete header of SCI code for later use.

Connecting Two HC08s

Figure 5 shows how to cross-couple two MMDS08 emulators. Note that these signals are cross-coupled. Essentially, TxD is connected to RxD on MMDS08 2.

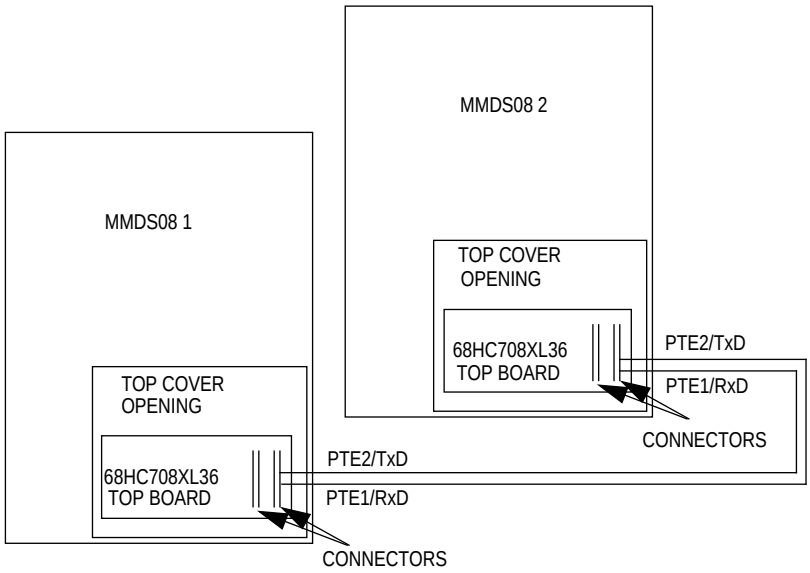

Figure 5. Cross-Coupled MMDS08 Diagram

Once TxD and RxD are cross-coupled, software then can be written on the other MMDS08 to receive the data being transmitted.

The flowchart in **Figure 6** illustrates the basic operation of the SCI as it receives data. However, further explanation and a software code example are necessary. **Appendix B: Basic SCI Receiver Code** shows the implementation of the basic SCI receiver. This code is also mostly academic, since it is seldom that only one character will be received. Additional code is necessary to demonstrate the flow of data between two HC08 microcontrollers. Several variables are defined at the beginning of the program. Most of them are not actually used in the program code but are included to provide a complete header of SCI code for later use.

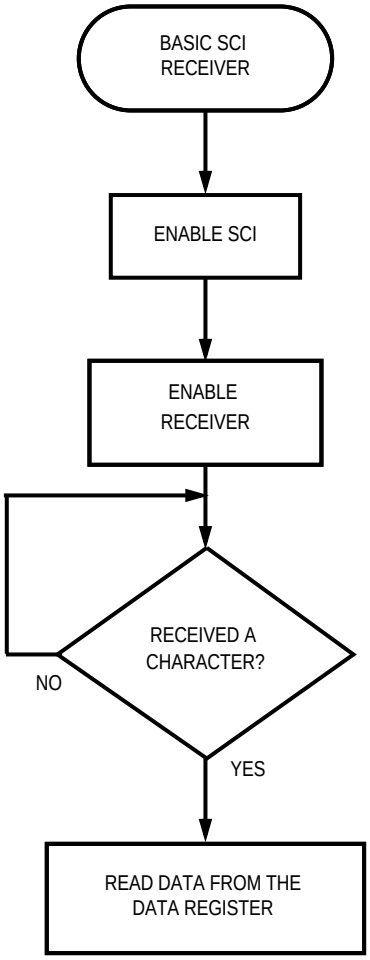

Figure 6. Basic SCI Receiver

Sending packets of information from one device to another is often necessary. For the purpose of this application note, a packet is defined as a string of characters of data, which is variable in length and is terminated with, but does not contain, the carriage return character (decimal value 13). For example, the phrase, 'This is a test,' as shown here, is a packet:

```
546869732069      DATA:      fcb     'This is a test'
0D                           fcb     13
```

To better understand the SCI transmitter and receiver, see **Figure 7** for an example of how to send packets of data between two microcontrollers. By using the circuit diagram in **Figure 5**, new software is developed to provide a smoother flow between the two chips.The Ping-Pong got its name from the back and forth nature of its operation. One device sends a packet to the other. The receiving device then verifies that it has received the correct information and sends a packet of information back to the original sender. This creates a Ping-Pong effect such that when the data is viewed with an oscilloscope, the data actually bounces from one device to the other and then back again. **Appendix C: SCI Ping-Pong Code** is the software used to illustrate the Ping-Pong algorithm.

Notice that there is much more flow control in Ping-Pong than in the previous two program examples. This software can be classified as the first program described in this application note that can be used in practice. Notice, too, the many similarities between the transmitting and receiving MCUs. Also notice that the variable baud located in **Appendix C: SCI Ping-Pong Code** is set to $00. This value is updated in later program examples to explain how to change that baud rate. The default reset value of the SCI baud rate register is $00, which means that setting the value to $00 is the same as not setting it at all after a reset.

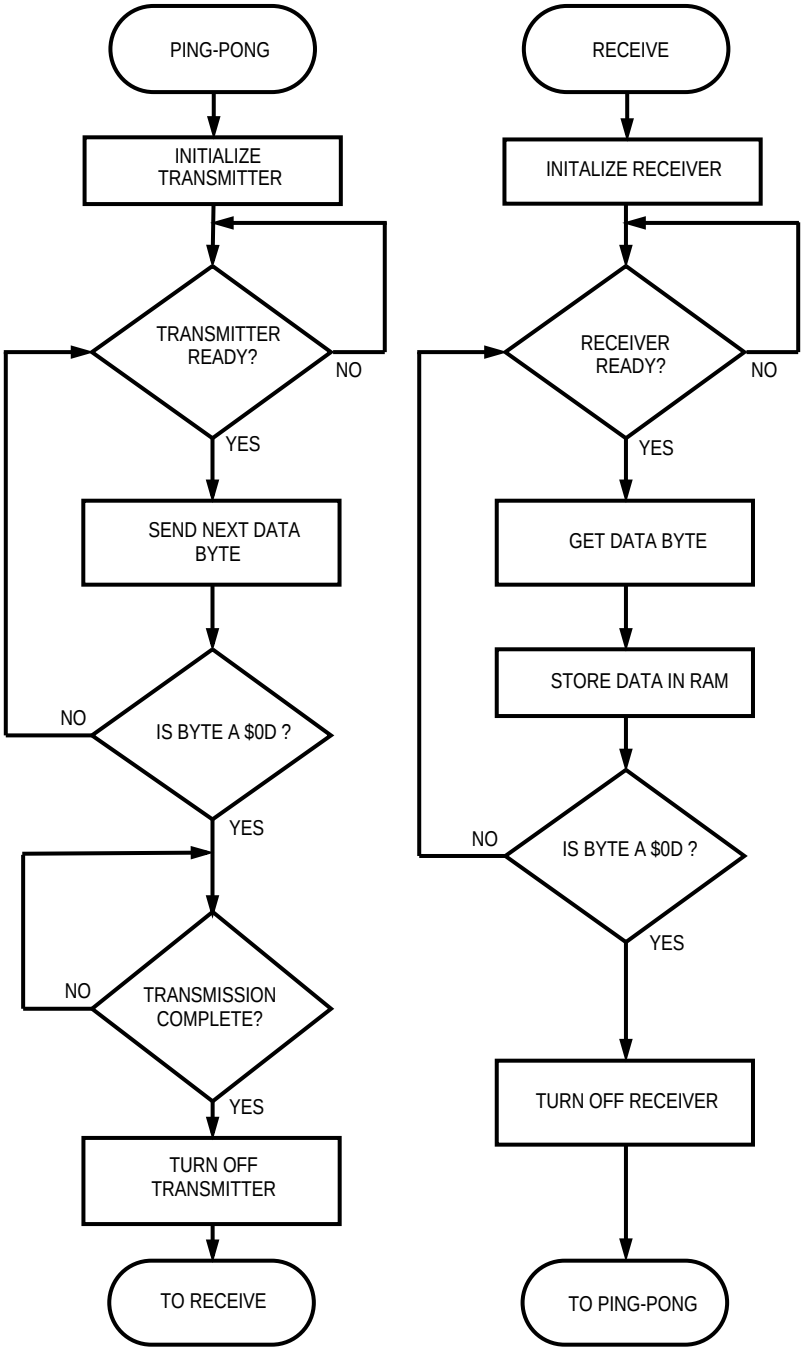

Figure 7. Ping-Pong Flowchart

SCI OPERATION WITH UNEQUAL INPUT CLOCK FREQUENCIES

When using unequal input clock frequencies, adjust the baud rate so that both devices operate at the same rate. As shown in **Figure 3,** there is a direct relationship between the input clock and the baud rate. The two divisors, PD and BD, are variables that allow the user to change the baud rate. The baud rate remains directly related to GCMXCLK so that when the input clock frequency increases, the baud rate also increases.

NOTE

When using the XL36 with CGMA, the phase lock loop (PLL) settings have no effect on the baud rate, even though changing the settings within the PLL cause the MCU internal clock to operate at a different frequency. This happens because the clock for the SCI is taken from the input clock and not the PLL output.

Figure 8 shows how to calculate the baud rate using the equation shown in **Figure 3**.

$$\text{baud rate} = \frac{\text{CGMXCLK}}{64 \times \text{PD} \times \text{BD}}$$

If using a 4.9152-MHz clock source and PD and BD are set to one, the equation becomes:

$$\text{baud rate} = \frac{4915200\ \text{Hz}}{64 \times 1 \times 1}$$

$$\text{baud rate} = 76{,}800$$

Figure 8. Baud Rate Calculation Example 1

Figure 9 shows that using the same settings but a different clock frequency produces a different baud rate.

$$\text{baud rate} = \frac{\text{CGMXCLK}}{64 \times \text{PD} \times \text{BD}}$$

If using a 8.000-MHz clock source and PD and BD are set to one, the equation becomes:

$$\text{baud rate} = \frac{8000000\ \text{Hz}}{64 \times 1 \times 1}$$

$$\text{baud rate} = 125{,}000$$

Figure 9. Baud Rate Calculation Example 2

If these two previous example MCUs were configured similarly to those in **Figure 5,** communication between them would not occur, because they operate at different baud rates. **Appendix D: SCI Baud Rate Chart** shows these two input clock frequencies. To allow these two example MCUs to communicate, pick a baud rate that is common between both sets. For example, choose 600 baud, which is common to both sets. For an 8-MHz input clock, use PD = 13 and BD = 16. For a 4.9152-MHz input clock, use PD = 4 and BD = 32. See **Figure 10** for more information.

$$\text{baud rate} = \frac{\text{CGMXCLK}}{64 \times \text{PD} \times \text{BD}}$$

If using an 8.000-MHz clock source, PD = 13, and BD = 16, the equation becomes:

$$\text{baud rate} = \frac{8000000\ \text{Hz}}{64 \times 13 \times 16}$$

$$\text{baud rate} = 600.96$$

$$\text{baud rate} = \frac{\text{CGMXCLK}}{64 \times \text{PD} \times \text{BD}}$$

If using a 4.915-MHz clock source, PD = 4, and BD = 32, the equation becomes:

$$\text{baud rate} = \frac{4915200\ \text{Hz}}{64 \times 4 \times 32}$$

$$\text{baud rate} = 600.00$$

Figure 10. Baud Rate Calculation Example 3

Although the baud rates are not exactly equal, they are close enough to work properly. Baud rates should be within 3.4% of each other to operate properly. See **Figure 11** for more information.

$$\text{baud rate deviation} = 600 \times 0.034 = 2.04$$

$$\begin{aligned}\text{baud rate} &= \text{baud rate} + \text{baud rate deviation} \\ &= 600 + 2.04 \\ &= 602.04\end{aligned}$$

Figure 11. Baud Rate Deviation Equation

If one MCU operates at 600 baud, the other may operate at as much as 2.04 baud out of frequency, which is 602.04 baud. So as shown in **Figure 10**, the baud rate of 600.96 is well within the 3.4% deviation limit and, therefore, will work properly in this implementation.

SCI OPERATION WITH COMMONLY USED BAUD RATES

Baud rates are used widely when communicating with modems. Modems are devices that convert binary data into audible frequencies that can be sent across a normal telephone line. Modems also can receive these frequencies and convert them back into digital data. The SCI format is in exactly the same format as that used by most modems. The only difference is the voltage levels. The SCI operates at the voltage supplied to the V_{DD} pin and the voltage supplied at the V_{SS} pin (commonly +5 V and 0 V, respectively). Since standard modems (or the RS-232 standard) require the voltage be +12 V and –12 V, circuitry is necessary to convert the levels of the MCU to these standard RS-232 levels. The MC1488 and MC1489 shift voltage levels to meet the RS-232 standard. **Figure 12** shows how to connect these chips to a modem.

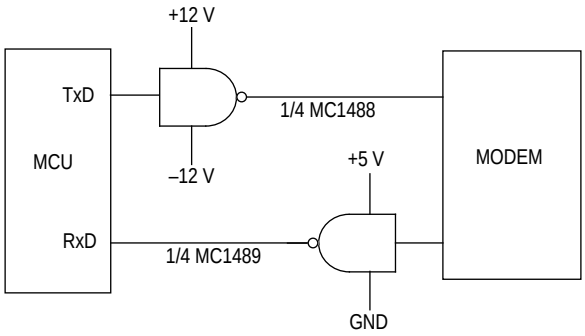

Figure 12. Modem Circuit

Typically, the baud rate between the modem and the MCU is higher than the baud rate between the modem and another modem. This is because modern modems have become so advanced that they automatically can sense the maximum baud rate of the second modem. As long as the link between the MCU and modem is as high as possible, the modem will, in most cases, find the highest baud rate. Over the years of modem development, several baud rates have been chosen. These are 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, and 57600.

To communicate with a modem, choose one of these baud rates. For instance, to communicate with a 1200-baud modem using the MC1488 and MC1489 chips as shown in **Figure 12,** choose 1200 from the baud rate chart. See **Appendix D: SCI Baud Rate Chart**. For a clock input frequency of 8 MHz, choose PD = 13 and BD = 8. This gives a baud rate of 1201.92. Using the baud rate deviation equation in **Figure 13**, calculate that the maximum deviation of the baud rate at 1200 baud is 4.08 as shown.

$$\text{baud rate deviation} = 1200 \times 0.034 = 4.08$$

$$\begin{aligned}\text{baud rate} &= \text{baud rate} + \text{baud rate deviation}\\ &= 1200 + 4.08\\ &= 1204.08\end{aligned}$$

Figure 13. Baud Rate Deviation Equation

This allows a baud rate between 1204.08 and 1195.92. Since the chosen baud rate is 1201.92, it is within tolerable limits.

Appendix A: Basic SCI Transmitter Code

```
*****************************************************************
** Basic SCI Transmitter program                               **
** By Rick Cramer                                              **
** 25 November 1995                                            **
**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**
**                                                             **
**  This program illustrates the basic operation of transmitting   **
**  data using the SCI on the 68HC708XL36 using CGMA.          **
*****************************************************************
*---------------------------------------------------------------*
DATA        EQU         $55     ; Data to transmit
*---------------------------------------------------------------*
SCC1        EQU         $13     ; SCI Control Register 1
LOOPS       EQU         7       ; Bit 7: Loop Mode Select Bit
ENSCI       EQU         6       ; Bit 6: ENable SCI Bit
TXINV       EQU         5       ; Bit 5: Transmit Inversion Bit
M           EQU         4       ; Bit 4: Mode (Character Length) Bit
WAKE        EQU         3       ; Bit 3: Wake-up Condition Bit
ILTY        EQU         2       ; Bit 2: Idle Line Type Bit
PEN         EQU         1       ; Bit 1: Parity Enable Bit
PTY         EQU         0       ; Bit 0: Parity Bit
*---------------------------------------------------------------*
SCC2        EQU         $14     ; SCI Control Register 2
SCTIE       EQU         7       ; Bit 7: SCI transmit Interrupt Enable Bit
TCIE        EQU         6       ; Bit 6: Transmission Complete Interrupt Enable
                                ;        Bit
SCRIE       EQU         5       ; Bit 5: SCI Receive Interrupt Enable Bit
ILIE        EQU         4       ; Bit 4: Idle Line Interrupt Enable Bit
TE          EQU         3       ; Bit 3: Transmitter Enable Bit
RE          EQU         2       ; Bit 2: Receiver Enable Bit
RWU         EQU         1       ; Bit 1: Receiver Wake Up Bit
SBK         EQU         0       ; Bit 0: Send Break Bit
*---------------------------------------------------------------*
SCC3        EQU         $15     ; SCI Control Register 3
R8          EQU         7       ; Bit 7: Received Bit 8
T8          EQU         6       ; Bit 6: Transmitted Bit 8
DMARE       EQU         5       ; Bit 5: DMA Receive Enable Bit
DMATE       EQU         4       ; Bit 4: DMA Transfer Enable Bit
ORIE        EQU         3       ; Bit 3: Receiver Overrun Interrupt Enable Bit
NEIE        EQU         2       ; Bit 2: Receiver Noise Error Interrupt Enable
                                ;        Bit
FEIF        EQU         1       ; Bit 1: Receiver Framing Error Interrupt Enable
                                ;        Bit
PEIE        EQU         0       ; Bit 0: Receiver Parity Error Interrupt Enable
                                ;        Bit
*---------------------------------------------------------------*
SCS1        EQU         $16     ; SCI Status Register 1
SCTE        EQU         7       ; Bit 7: SCI Transmitter Empty Bit
TC          EQU         6       ; Bit 6: Transmission Complete Bit
SCRF        EQU         5       ; Bit 5: SCI Receiver Full Bit
IDLE        EQU         4       ; Bit 4: Receiver Idle Bit
OR          EQU         3       ; Bit 3: Receiver Overrun Bit
```

```
NF          EQU         2       ; Bit 2: Receiver Noise Bit
FE          EQU         1       ; Bit 1: Receiver Framing Error Bit
PE          EQU         0       ; Bit 0: Receiver Parity Error Bit
*------------------------------------------------------------------*
SCS2        EQU         $17     ; SCI Status Register 2
BKF         EQU         1       ; Bit 1: Break Flag Bit
RPF         EQU         0       ; Bit 0: Reception in Process Flag Bit
*------------------------------------------------------------------*
SCDR        EQU         $18     ; SCI Data Register
*------------------------------------------------------------------*
SCBR        EQU         $19     ; SCI Baud Rate Register
*------------------------------------------------------------------*

            org $6E00           ; EPROM Space in the Memory map is $6E00 on the
                                ; MC68HC708XL36
            bset ENSCI,SCC1     ; Enable the SCI by writing a logic one to the
                                ; enable
                                ; SCI bit (ENSCI) in the SCI Control Register 1
                                ; (SCC1)
            bset TE,SCC2        ; Enable the transmitter by writing a logic one
                                ; to the
                                ; transmitter enable bit (TE) in the SCI Control
                                ; Register 2
                                ; (SCC2)
            ldx SCS1            ; Clear the SCI transmitter empty bit (SCTE) by
                                ; reading
                                ; SCI Status Register 1(SCS1)
            lda #DATA           ; Load the Accumulator with the DATA that you
                                ; want to transmit
            sta SCDR            ; Write the data to transmit into the SCI Data
                                ; Register (SCDR)
            end
```

Appendix B: Basic SCI Receiver Code

```
*****************************************************************
** Basic SCI Receiver program                                  **
** By Rick Cramer                                              **
** 25 November 1995                                            **
**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**
**                                                             **
**  This program illustrates the basic operation of receiving  **
**  data using the SCI on the 68HC708XL36 using CGMA.          **
*****************************************************************
*---------------------------------------------------------------*
DATA        EQU         $55     ; Data to transmit
*---------------------------------------------------------------*
SCC1        EQU         $13     ; SCI Control Register 1
LOOPS       EQU         7       ; Bit 7: Loop Mode Select Bit
ENSCI       EQU         6       ; Bit 6: ENable SCI Bit
TXINV       EQU         5       ; Bit 5: Transmit Inversion Bit
M           EQU         4       ; Bit 4: Mode (Character Length) Bit
WAKE        EQU         3       ; Bit 3: Wake-up Condition Bit
ILTY        EQU         2       ; Bit 2: Idle Line Type Bit
PEN         EQU         1       ; Bit 1: Parity Enable Bit
PTY         EQU         0       ; Bit 0: Parity Bit
*---------------------------------------------------------------*
SCC2        EQU         $14     ; SCI Control Register 2
SCTIE       EQU         7       ; Bit 7: SCI transmit Interrupt Enable Bit
TCIE        EQU         6       ; Bit 6: Transmission Complete Interrupt Enable
                                ;        Bit
SCRIE       EQU         5       ; Bit 5: SCI Receive Interrupt Enable Bit
ILIE        EQU         4       ; Bit 4: Idle Line Interrupt Enable Bit
TE          EQU         3       ; Bit 3: Transmitter Enable Bit
RE          EQU         2       ; Bit 2: Receiver Enable Bit
RWU         EQU         1       ; Bit 1: Receiver Wake Up Bit
SBK         EQU         0       ; Bit 0: Send Break Bit
*---------------------------------------------------------------*
SCC3        EQU         $15     ; SCI Control Register 3
R8          EQU         7       ; Bit 7: Received Bit 8
T8          EQU         6       ; Bit 6: Transmitted Bit 8
DMARE       EQU         5       ; Bit 5: DMA Receive Enable Bit
DMATE       EQU         4       ; Bit 4: DMA Transfer Enable Bit
ORIE        EQU         3       ; Bit 3: Receiver Overrun Interrupt Enable Bit
NEIE        EQU         2       ; Bit 2: Receiver Noise Error Interrupt Enable
                                ;        Bit
FEIF        EQU         1       ; Bit 1: Receiver Framing Error Interrupt Enable
                                ;        Bit
PEIE        EQU         0       ; Bit 0: Receiver Parity Error Interrupt Enable
                                ;        Bit
*---------------------------------------------------------------*
SCS1        EQU         $16     ; SCI Status Register 1
SCTE        EQU         7       ; Bit 7: SCI Transmitter Empty Bit
TC          EQU         6       ; Bit 6: Transmission Complete Bit
SCRF        EQU         5       ; Bit 5: SCI Receiver Full Bit
IDLE        EQU         4       ; Bit 4: Receiver Idle Bit
```

```
OR          EQU         3       ; Bit 3: Receiver Overrun Bit
NF          EQU         2       ; Bit 2: Receiver Noise Bit
FE          EQU         1       ; Bit 1: Receiver Framing Error Bit
PE          EQU         0       ; Bit 0: Receiver Parity Error Bit
*------------------------------------------------------------------*
SCS2        EQU         $17     ; SCI Status Register 2
BKF         EQU         1       ; Bit 1: Break Flag Bit
RPF         EQU         0       ; Bit 0: Reception in Process Flag Bit
*------------------------------------------------------------------*
SCDR        EQU         $18     ; SCI Data Register
*------------------------------------------------------------------*
SCBR        EQU         $19     ; SCI Baud Rate Register
*------------------------------------------------------------------*
            org $6E00           ; EPROM Space in the Memory map is $6E00 on the
                                ; MC68HC708XL36
            bset ENSCI,SCC1     ; Enable the SCI by writing a logic one to the
                                ; enable
                                ; SCI bit (ENSCI) in the SCI Control Register 1
                                ; (SCC1)
            bset RE,SCC2        ; Enable the Receiver by writing a logic one to
                                ; the
                                ; Receiver enable bit (RE) in the SCI Control
                                ; Register 2
                                ; (SCC2)
            brclr SCRF,SCS1,*   ; Wait here until a data byte comes by Polling
                                ; the SCI Receiver
                                ; Full Bit in the SCI Status Register 1.
            lda SCDR            ; Get the Data from the SCI Data Register
            end
```

Appendix C: SCI Ping-Pong Code

```
*****************************************************************
** SCI Ping Pong                                               **
** By Rick Cramer                                              **
** 25 November 1995                                            **
**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**
**                                                             **
**  This program illustrates the operation of transmitting and **
**  receiving data packets using the SCI on the 68HC708XL36 using **
**  CGMA.                                                      **
*****************************************************************
BAUD        equ         $00     ; Baud Rate set to Default Value
*---------------------------------------------------------------*
SCC1        EQU         $13     ; SCI Control Register 1
LOOPS       EQU         7       ; Bit 7: Loop Mode Select Bit
ENSCI       EQU         6       ; Bit 6: ENable SCI Bit
TXINV       EQU         5       ; Bit 5: Transmit Inversion Bit
M           EQU         4       ; Bit 4: Mode (Character Length) Bit
WAKE        EQU         3       ; Bit 3: Wake-up Condition Bit
ILTY        EQU         2       ; Bit 2: Idle Line Type Bit
PEN         EQU         1       ; Bit 1: Parity Enable Bit
PTY         EQU         0       ; Bit 0: Parity Bit
*---------------------------------------------------------------*
SCC2        EQU         $14     ; SCI Control Register 2
SCTIE       EQU         7       ; Bit 7: SCI transmit Interrupt Enable Bit
TCIE        EQU         6       ; Bit 6: Transmission Complete Interrupt Enable
                                ;        Bit
SCRIE       EQU         5       ; Bit 5: SCI Receive Interrupt Enable Bit
ILIE        EQU         4       ; Bit 4: Idle Line Interrupt Enable Bit
TE          EQU         3       ; Bit 3: Transmitter Enable Bit
RE          EQU         2       ; Bit 2: Receiver Enable Bit
RWU         EQU         1       ; Bit 1: Receiver Wake Up Bit
SBK         EQU         0       ; Bit 0: Send Break Bit
*---------------------------------------------------------------*
SCC3        EQU         $15     ; SCI Control Register 3
R8          EQU         7       ; Bit 7: Received Bit 8
T8          EQU         6       ; Bit 6: Transmitted Bit 8
DMARE       EQU         5       ; Bit 5: DMA Receive Enable Bit
DMATE       EQU         4       ; Bit 4: DMA Transfer Enable Bit
ORIE        EQU         3       ; Bit 3: Receiver Overrun Interrupt Enable Bit
NEIE        EQU         2       ; Bit 2: Receiver Noise Error Interrupt Enable
                                ;        Bit
FEIF        EQU         1       ; Bit 1: Receiver Framing Error Interrupt Enable
                                ;        Bit
PEIE        EQU         0       ; Bit 0: Receiver Parity Error Interrupt Enable
                                ;        Bit
*---------------------------------------------------------------*
SCS1        EQU         $16     ; SCI Status Register 1
SCTE        EQU         7       ; Bit 7: SCI Transmitter Empty Bit
TC          EQU         6       ; Bit 6: Transmission Complete Bit
SCRF        EQU         5       ; Bit 5: SCI Receiver Full Bit
IDLE        EQU         4       ; Bit 4: Receiver Idle Bit
OR          EQU         3       ; Bit 3: Receiver Overrun Bit
```

```
NF          EQU         2       ; Bit 2: Receiver Noise Bit
FE          EQU         1       ; Bit 1: Receiver Framing Error Bit
PE          EQU         0       ; Bit 0: Receiver Parity Error Bit
*------------------------------------------------------------------*
SCS2        EQU         $17     ; SCI Status Register 2
BKF         EQU         1       ; Bit 1: Break Flag Bit
RPF         EQU         0       ; Bit 0: Reception in Process Flag Bit
*------------------------------------------------------------------*
SCDR        EQU         $18     ; SCI Data Register
*------------------------------------------------------------------*
SCBR        EQU         $19     ; SCI Baud Rate Register
*------------------------------------------------------------------*

            org $6E00           ; EPROM Space in the Memory map is $6E00 on the
                                ; MC68HC708XL36

DATA:       fcb     'This is a test'
            fcb     13

initscitx:  lda #$01
            sta $1f             ; Disable COP in MMDS08
            lda #BAUD
            sta SCBR            ; Setup BAUD Rate
            bset ENSCI,SCC1     ; Set the Enable SCI Bit
            bset TE,SCC2        ; Set Transmit Enable Bit
            lda SCS1            ; Clear SCI Transmitter Empty Bit
            ldx #$00            ; Index Counter
getbyte:    lda DATA,x          ; Get the packet from “DATA”
            incx                ; Increment the counter to get next data byte
            brclr SCTE,SCS1,*   ; Wait until transmitter is ready
            sta SCDR            ; Store Data in SCI Data Register
            cmp #$0D            ; Is this the end of the packet?
            bne getbyte         ; If not, then get another byte to send
            brclr TC,SCS1,*     ; Wait for transmitter to finish
            bclr TE,SCC2        ; Stop the transmitter
            lda #$00
            sta SCDR            ; Clear the SCI Data Register


initscirx:  bset ENSCI,SCC1     ; Set the Enable SCI Bit
            bset RE,SCC2        ; Set Receive Enable
            ldx #$00            ; Setup the index counter

****Check the received character ****
next:       brclr SCRF,SCS1,*   ; Wait here until a chr comes
            lda SCDR            ; Get the Data from the SCI Data Register
            sta $50,x           ; Store the data in RAM
            cmp DATA,x          ; Error Checking
            bne bad             ; Run “bad” routine if data is bad
back        incx                ; Increment index register to receive next byte
            cmp #$0D            ; Is the data received the “End of Packet”?
            bne next            ; If Not, then get the NEXT byte of the packet
            bclr RE,SCC2        ; Turn Off the receiver
            jmp initscitx       ; Go back to the beginning.
```

```
*****************************************************************
** bad routine                                                 **
**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**
**                                                             **
**  This subroutine turns on an LED if an error has occurred.  **
*****************************************************************

bad:        stx $90             ; Save index register for later use
            ldx #$ff            ; %1111 1111
            stx $00             ; Port A Data Direction Register = output
            stx $04             ; Port A Data Register = HIGH
            ldx $90             ; Restore Index Register
            bra back            ; Always go back to main program.

*****************************************************************
** Reset Vectors                                               **
**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**
**                                                             **
**  These Vectors tell the HC08 what line to execute upon reset.  **
*****************************************************************

            org $fffe
            fdb initscitx

            END
```

Appendix D: SCI Baud Rate Chart

Table 1. SCI Baud Rates

PD	BD	SCBR Binary	SCBR Hex	CGMXCLK 8.00 MHz	CGMXCLK 4.9152 MHz
1	1	%0000 0000	$00	125000.00	76800.00
1	2	%0000 0001	$01	62500.00	38400.00
1	4	%0000 0010	$02	31250.00	19200.00
1	8	%0000 0011	$03	15625.00	9600.00
1	16	%0000 0100	$04	7812.50	4800.00
1	32	%0000 0101	$05	3906.25	2400.00
1	64	%0000 0110	$06	1953.13	1200.00
1	128	%0000 0111	$07	976.56	600.00
3	1	%0001 0000	$10	41666.67	25600.00
3	2	%0001 0001	$11	20833.33	12800.00
3	4	%0001 0010	$12	10416.67	6400.00
3	8	%0001 0011	$13	5208.33	3200.00
3	16	%0001 0100	$14	2604.17	1600.00
3	32	%0001 0101	$15	1302.08	800.00
3	64	%0001 0110	$16	651.04	400.00
3	128	%0001 0111	$17	325.52	200.00
4	1	%0010 0000	$20	31250.00	19200.00
4	2	%0010 0001	$21	15625.00	9600.00
4	4	%0010 0010	$22	7812.50	4800.00
4	8	%0010 0011	$23	3906.25	2400.00
4	16	%0010 0100	$24	1953.13	1200.00
4	32	%0010 0101	$25	976.56	600.00
4	64	%0010 0110	$26	488.28	300.00
4	128	%0010 0111	$27	244.14	150.00
13	1	%0011 0000	$30	9615.38	5907.69
13	2	%0011 0001	$31	4807.69	2953.85
13	4	%0011 0010	$32	2403.85	1476.92
13	8	%0011 0011	$33	1201.92	738.46
13	16	%0011 0100	$34	600.96	369.23
13	32	%0011 0101	$35	300.48	184.62
13	64	%0011 0110	$36	150.24	92.31
13	128	%0011 0111	$37	75.12	46.15

Home Page:
www.freescale.com
email:
support@freescale.com
USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
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
+81 2666 8080
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
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com