Lab 8: Asynchronous Serial Communications

Introduction

In this exercise, a "dumb" terminal communicates with the EVB using the Serial Communications Interface (SCI) of the 68HC11. You will design and run a driver program for the terminal that can send and receive characters that are typed on the keyboard. The objective is to understand how serial data can be sent back and forth between devices. Read sections 10.1.1, 10.2, and 10.3 in the textbook, and sections 8.4 and 8.6. Also useful are the register and control bit assignments in Table B.1A (textbook pages 635-636).

Basic tasks necessary to send and receive ASCII information.

1. Determine the details of the communication protocol (baud rate, parity, stop and start bits, etc.). Will interrupts be used? Is there a transmit and/or receive routine?
2. Configure the SCI system according to the protocol (the transmit and receive registers).
3. Design a transmit routine and code it.
4. Design a receive routine; code it.
5. Structure the transmit and receive routines within the context of the driver program.
6. Run and debug the driver program.

There are number of registers associated with the SCI. Keep in mind that many of the bits in the registers are either "read only" or "write only". And, unlike memory, when you write to a register it may not be possible to read that same data back from the given location. The important registers for this lab are the SCCR1, SCCR2, Baud register, DATA register, and SCI status register (SCSR).

The programs you write for this lab should use *polling* (i.e., do not use interrupts); also use a baud rate of 9.6 kbps (kilobits per second).

The teaching assistants will provide you with the code to initialize the SCI system.

Write and run the following two programs:

Program 1

Write a program that echoes a single character each time it is typed on the dumb terminal when connected to the EVB at P3. Use your own routines and <u>do not</u> rely on subprograms that might be available in the BUFFALO monitor.

Program 2

Extend Program 1 to accept characters until an ASCII period is typed by the user. Then display the entire line on the dumb terminal screen.

Hints:

1. Set the SCCR1 register to all zeros for one start bit, 8 data bits, and 1 stop bit; set the WAKE bit to zero. Set the SCCR2 register so that the TE and RE bits are high, and the high four bits are set to all zero (to disable interrupts). Set the BAUD register to $30 for 9.6 kbps.

   You must also write $FF to location $4000 as part of your initialization. This will enable the P3 connector on the EVB.

2. The data register (SCDR, for Serial Communication Data Register) is where the data to be transmitted is written, and where the received data comes from upon a read.

3. The big problem in all of this is that the serial transmission times are very long compared to the execution times of a typical instruction. Therefore it is necessary to check for each character: was it sent? Or is a new character ready to be read? These can be checked by reading the status register (SCSR) and examining the TDRE (Transmit Data Register Empty) bit to see if a character has been transmitted, or the RDRF (Receive Data Register Full) bit to see if a new character has arrived. If not, just keep checking these bits as appropriate. This *polling mode* keeps the processor busy, but it is easier to set up and debug than the interrupt mode.

5. We are receiving and transmitting data using the ASCII code. Read pages in 354-359 in Chapter 10, and the ASCII Chart in Appendix B (p. 641) in the textbook.