LAB #8: SCI Serial Network Interface

Lab writeup is due to your TA at the beginning of your next scheduled lab. Don’t put this off to the last minute! There is pre-lab work to complete before the start of the next lab. **NO LATE LAB REPORTS WILL BE ACCEPTED.**

1 Objectives

- Design a low-level communication interface between two or more microcomputers.
- Analyze the synchronization problems that occur when two computers are interfaced.
- Investigate low-level methods for error detection.
- Implement half-duplex serial communication.

2 Reading

- Read Chapter 7 on SCI serial interfacing.

3 Background

Figure 1 shows an open-collector half-duplex asynchronous serial channel network. It is half-duplex because all the TxD and RxD pins are shorted together. Therefore, if two microcomputers attempt to transmit at the same time there is a possibility of a collision. There are many ways to detect such collisions. Since the network is half-duplex, the frame goes to all the RxD serial inputs (including the one transmitting the frame) at the same time. In this lab, you will check the echo of each transmitted byte to detect collisions.

In this lab, you should use the asynchronous serial interface, SCI, with interrupt synchronization. The asynchronous serial protocol will be 8 bits, no parity, 1 stop bits, and 9600 baud.

The software for this lab will be divided into two parts. The low-level “device driver” software will provide support for initialization, transmitting, and receiving individual bytes across the network. Both the receive and transmit I/O threads must be interrupt driven. Two FIFO data structures will link the background threads and foreground thread. Collision detection and reporting is built into this layer. At this level, collision is detected at the transmitter when the echoed data frame received does not match the data frame transmitted. On the receiver end, a collision may result in a NF, or FE error.

The second part is the main program that test the low-level network. The main program will construct frames and send them out on the network. It will also receive frames and check if they are intended for this microcomputer. Each frame will consist of 8-bits. The first four bits will be a network address, and the second four bits will be data. Each microcomputer should have a unique network address. The main loop should construct a frame addressed to some other microcomputer, transmit the frame, and check for collisions. If collisions are detected, it should retry after waiting some backoff delay to avoid repeated collisions. It should then check if a frame has arrived and if so, check the frame’s address against its own. If there is a match, it should output the data onto 4 LEDs connected to an output port. It should then repeat by constructing a new frame. You may hard code a series of several frames to loop through. Be creative on setting up your test. Be sure to put some delays into your loop to help minimize collisions.
4 Prelab

1. Write assembly code for your low-level device driver software.

2. Write assembly code for your main program.

5 Lab Tasks

1. Test your code using a serial port connection to the PC and hyperterminal.

2. Connect your microcomputer to another groups microcomputer. Be sure to select unique network addresses for each microcomputer. Create a set of frames in each code that would demonstrate that the microcomputers are communicating (for example, one could send frames counting up while the other sends frames counting down).

3. For more fun, connect 3 or more microcomputers together and create a real network!

6 Writeup

Include the following items. In this lab, only one writeup per team is required.

1. A hardware schematic.

2. A printout of all your assembly code.