LAB #8: SCI Serial Network Interface

The Lab write-up 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 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 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 batch the data frame transmitted. On the receiver end, a collision may result in a NF (noise flag), or FE (framing error) error code.

The second part is the main program that tests 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 high order four bits (i.e., b7-b4) will be a network address, and the low order four bits (i.e., b3-b0) 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 retransmit after waiting some back-off 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 or the LCD. 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.

To setup the network you will need to short the TxD and RxD pinx coming out of the DB9 connector on the project board. For proper operation you should ensure that the COM_SEL jumper on the project board is set to COM. The MCU_COM and COM_EN jumpers on the project board should be enabled as well. You will also need to use a pull-up resister. To ensure that the value of ground is the same between them you should connect an alligator clip between the two ground pins on the project board.
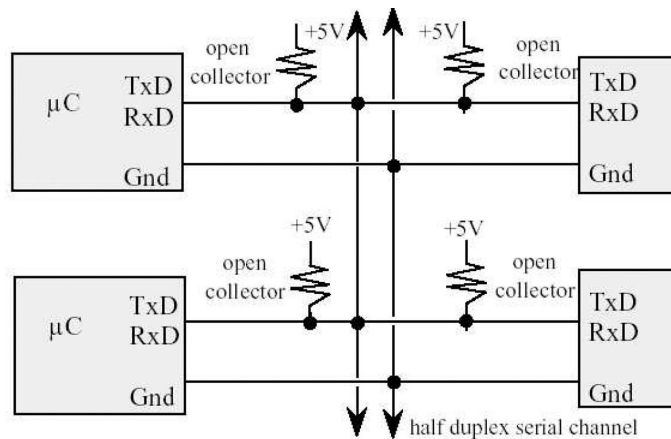

Figure 1: Block diagram of an open-collector half-duplex asynchronous serial channel network.

3 Pre-lab

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

2. Write C code for your main program.

4 Tasks

1. Test your code by connecting a serial cable between the DB9 connector on your module to the serial port connection on a PC without shorting TxD and RxD. Use hyperterminal to test your transmit and receive code. Demonstrate this to the TA.

2. Connect your microcomputer to another groups microcomputer (or the TAs). 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). Demonstrate this to the TA.

5 Writeup

1. A printout of your commented code.