EXPERIMENT #10
UART DATA TRANSMISSION

INTRODUCTION:

Most digital systems process and communicate information more than one bit at a time; for example, many computer systems transfer data a byte (8 bits) at a time. However, the methods of converting digital signals for analog transmission (such as FSK) only deal with one bit at a time. You can see that it would be highly impractical to require eight telephone lines to transfer 8-bit parallel data! There *is* a way to get parallel information to "fit" onto a single-bit data path; the process of converting parallel information into serial (one bit at a time) information is called *serialization*, and this is one of the purposes of the UART chip. The transmit section of the UART in essence converts *parallel* information to *serial* information.

At the receiver, a serial data stream will be received. The receiver's UART will convert the serial data back into parallel data. The receiving system will then be ready to process the information.

CIRCUIT ANALYSIS:

Figure 1 shows the circuit used in this lab. As you can see, the UART chip does most of the work! Everything else is either an input, an output, or a clock signal.

This circuit essentially talks to itself. Parallel data is supplied by DIP switch S3, with the help of the 4.7K pullup resistors in R3. This parallel data is applied to the UART on the *TBR1-TBR8* pins, which is where the UART transmitter expects to get its parallel data from a computer or other device.

Data transmission is initiated with switch S2, which drives the *TBRL* input of the UART. On the rising edge of *TBRL* (switch is released), the UART converts the provided parallel data on *TBR1-TBR8* into serial data, and outputs the serial data through its *TRO* pin.

The serial data signal on *TRO* is digitally looped back into the *RRI* (receiver register input). The UART then converts the received serial data signal on *RRI* back into a parallel output, which is then provided at the *RBR1-RBR8* output pins. The parallel data pattern is made visible by driver IC U2 and LED pack U3. U2 is used to "buffer" the LEDs, since they would draw excessive current from the UART output pins.

The *status* signals (DR, PE, FE, OE) are also displayed through LED pack U5, with U4 providing the necessary signal buffering (current amplification).

The UART requires a clock signal with a frequency equal to 16 times the desired data rate in bits per second (bps). The *TTL OUTPUT* of the benchtop signal generator will be used to provide a proper clock. Note that the same clock signal will supply both receiver and transmitter sections.

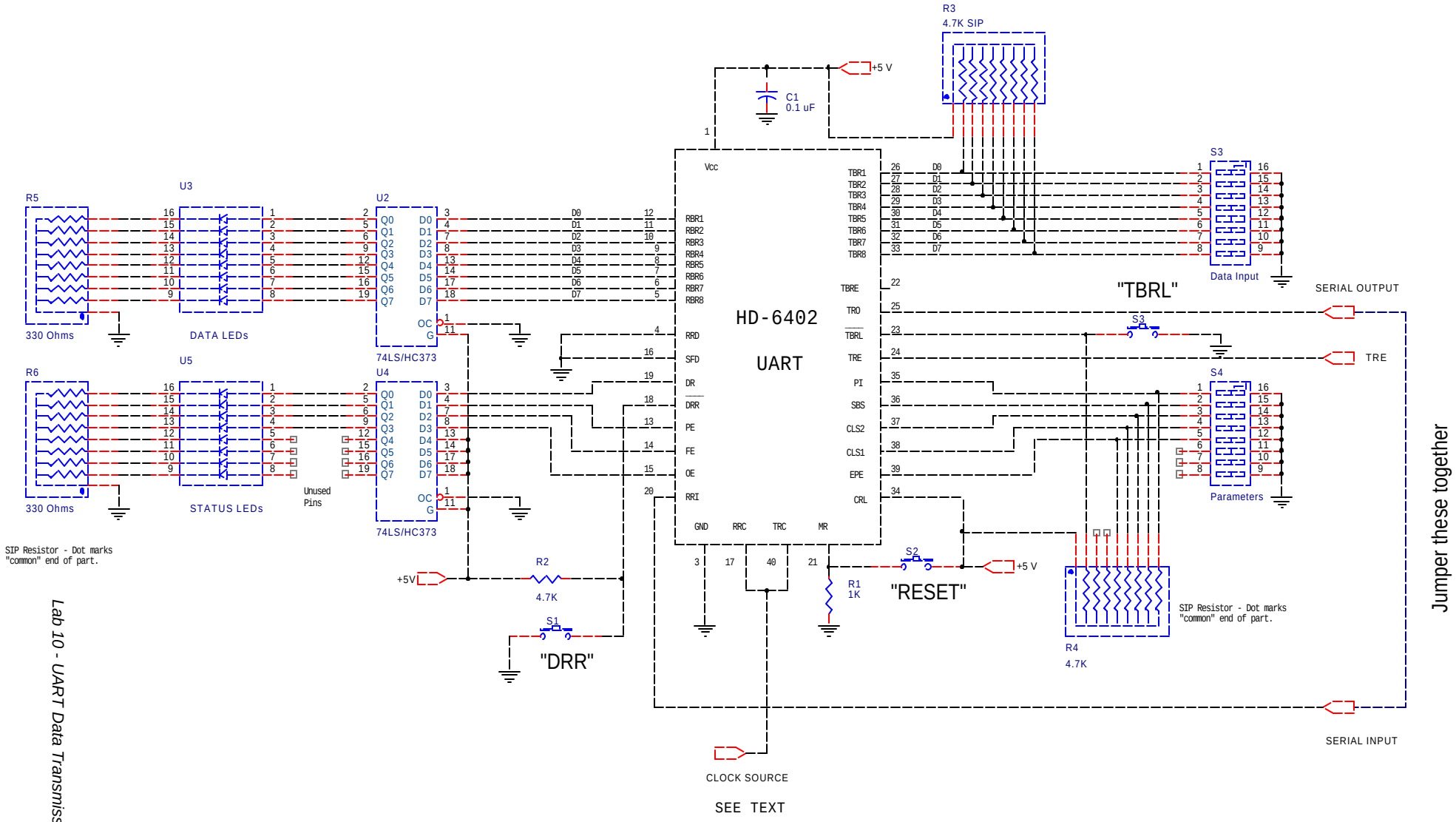

Figure 1: UART Data Transmitter with Digital Loopback

LABORATORY PROCEDURE:

Name _____ Sign-off _____

PART I: General Checkout and Receiver Signal Observation

1. Build the circuit of Figure 1.

2. Set the clock frequency for a data rate of 300 bps. (Use a frequency counter for best accuracy). Remember that the UART needs a clock frequency 16 times the data rate.

   IMPORTANT: The clock needs to be CMOS compatible to properly drive the UART clock inputs on pins 17 and 40. The logic output on some signal generators doesn't swing all the way up to +5V, which might cause the UART to ignore the clock. To fix this problem, add a 4.7k pull-up resistor to Vcc on the clock input.

3. Set the parameter switches for: 8 data bits, no parity, 1 stop bit. You'll need to refer to the data sheet for the UART, which is in the back of this manual.

> Note: If you have trouble getting your circuit to work properly, don't panic! Troubleshooting the UART circuit isn't as bad as it looks. Just remember to check:
>
> a) *Visually* - look for obvious problems;
>
> b) *Power Supply* - Check Vcc and GND potentials;
>
> c) *Inputs and Outputs* - The inputs are the clock signal (*RRC* and *TRC*), the control word (*PI,SBS,CLS1,CLS2,* and *EPE*), and the data input word (*TBR1-TBR8*). The outputs are the serial output *TRO*, as well as the received data output word and status word.

4. Let's observe whether or not we can now send 8-bit data to ourselves:

a) Apply power, and press the RESET switch to put the UART into a known state. All status LEDs (U5) should turn off. The *data* LEDs may remain lit in any pattern.

b) Set the data input to FFH. Then press *TBRL*. Does the data appear on the RECEIVED DATA LEDs? Do any other indicator lights come on? Which one(s)? What does the data sheet say about them?

c) Send the data 00H. Is it working? Did any new lights come on since step (b)? If everything went well, your UART circuit is working!

5. Let's take a look at how the receiver DR, RDR, and OE signals work. As you recall, the DR signal tells the receiver that a byte has been received, and it stays on until the receiver resets it with the RDR input. In addition, the OE signal is turned on when the UART receiver receives a new data byte before the old one could be read by the receiving device. Devise your own sequence for testing the DR, RDR, and OE signals; make sure to REPORT ON YOUR SEQUENCE AND RESULTS!

PART II: Transmit Signal Observation

6. Now we want to use the oscilloscope to look at the various timing relationships of the UART transmit section. To view a waveform on an oscilloscope, it must be periodic (repeating). The addition of the jumper between pins 23 and 24 of the UART will cause it to repeatedly send the same character. To begin the transmission, the *TBRL* switch is activated. The only way to halt the transmission is to remove power or press RESET.

   Add a jumper wire between pin 23 (TBRL) and pin 24 (TRE).

7. The TRE signal goes high after each complete character (byte) has been transmitted, and goes low while a character is being transmitted. It will be used as the reference for all other waveforms, because it will synchronize the scope to the start of each character.

   Connect scope channel #1 to the TRE signal, triggering off this channel.

8. Make sure the parameters are still at 8,N,1 , and start sending the byte A5H.

9. Record one complete cycle of the TRE signal in your graph section.

10. Use channel #2 of the scope to view the serial data output (*TRO*). (Leave triggering and TRE display of channel #1 on the screen.) You should be able to measure the bit-time. Record the complete A5H byte transmission in your graph section, WITH RESPECT TO TRE. Label each bit cell with its contents (i.e., start bit, stop bit, D0-D7, etc.)

11. Change the transmit parameters to 8 data, 1 stop, even parity; again, record the data output on a graph with respect to TRE. It can share the same page as the graph from step 10.

12. Change the transmit parameters to 8 data, 1 stop, odd parity; record the result.

13. The UART supposedly uses 16 clock cycles for each bit-cell. See if you can verify this. Record the graph of ONE BIT CELL VERSUS THE TRANSMIT CLOCK.

Questions

1. What is the function of a UART?

_____

_____

2. What is controlled by DIP switch S4?

_____

_____

3. Why are ICs U2 and U4 needed in this circuit?

_____

_____

4. Did you have any troubleshooting to do? Explain.

_____

_____

_____

_____

_____

_____