EXPERIMENT #11
UART DATA RECEPTION

INTRODUCTION:

Under normal conditions, a receiving device (i.e., computer) must wait for each character or byte from the UART. This is because the serial data transmission process is a lot slower than the operation of the receiving computer. To coordinate the reception process, a process of *handshaking* is used.

A *handshake* is a procedure for coordinating (controlling) the transfer of data. The HD-6402 UART receiver section has two signal pins that are used for handshaking, *DR* (Data Ready) and *DRR* (Data Ready Reset). The operation of these signals will be studied in this experiment.

UART Error Conditions

For a UART to correctly receive data, several important things about the overall system setup must be true:

- The transmitter and receiver bit-rate clocks must be very close in frequency;
- The transmitter and receiver must agree on the number of data bits to be sent;

  and,

- The transmitter and receiver must agree on whether or not parity will be used, and if so, what type of parity (even or odd) will be used.

If any of these conditions are not met, then the system is very likely to pass erroneous data; and in many cases, the system will not work at all. In this experiment, these settings are controlled by DIP switch inputs and the benchtop signal generator. In a real-world application, the settings will be under software control - meaning that the parameter settings for communication will likely be contained within a menu of an application program.

CIRCUIT ANALYSIS:

The experiment uses two complete UART circuits with independent clock sources, both of which are initially set the proper frequency to produce a 1200 bps data rate. Data flows in just one direction, so the mode of transmission is simplex.

The transmitting UART is set to transmit a character pattern repeatedly, as in Experiment 10. This data travels directly to the receiving UART, where it will be displayed as a complete data byte.

© 2001-2010 Tom A. Wheeler

A receiver UART is normally connected to a computer or other processing system. When a UART receives a completed data byte, it activates its *DR* signal. In turn, the computer recognizes this as a signal that data is present. The computer reads the data, then activates the *DRR* signal to reset the flag. This procedure is called a *handshake.* The process conversationally might look like this:

```
UART: Good day. I have data waiting for you.                ("DR=TRUE")

COMPUTER: Very good. I now have the data. Thank you.        ("DRR=TRUE")

UART: You're welcome; I see you have the data now.          ("DR=FALSE")
      I'll let you know if anything else comes in for you.

COMPUTER: OK, I appreciate that.  I'll wait until you       ("DRR=FALSE")
           alert me again.
```

There is no computer in this system, but the necessary logic to drive the *DRR* signal can easily be simulated. Logically, when *DR* is high, data is available. To satisfy the UART that we received the data from it, we need to activate the *DRR* signal (it is active low). Activating *DRR* will reset *DR*, completing the cycle. Therefore, an *inverter* circuit could possibly satisfy the handshaking requirement, except for one complication.

Imagine an inverter placed between *DR* and *DRR*. Data arrives and *DR* goes high. The output of the inverter naturally goes low. This activates *DRR*, resetting *DR* to a zero again. The cycle would then be completed (*DR* would be returned to its original state, a logic 0, indicating that UART's receiver register is empty).

Unfortunately, the time required for this to happen is merely the sum of the propagation delay of the gates in the circuit (including the internal logic of the UART), which is too fast to easily see on the oscilloscope. We need to slow it down a bit! That is the function of the *pulse stretcher*. The pulse stretcher adds a small delay to the handshaking process, which simulates the response time of a computer receiving data from the UART.

The pulse stretcher circuit of Figure 1 is essentially a "delayed inverter." An RC time-constant is formed by R1 and C1. R2 and R3 are a voltage divider. They increase the turn-on voltage of the transistor (at the capacitor terminals) to about 1.4 volts, thus increasing the time delay. The diode rapidly discharges the capacitor when *DR* goes low, in order to prepare the circuit for the next cycle.

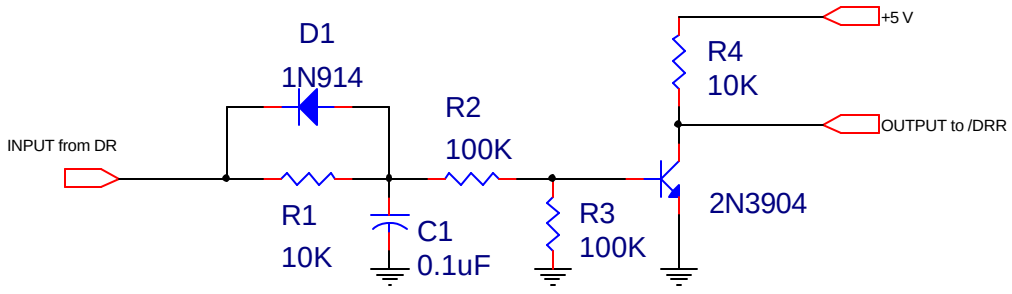

Figure 1: Receiver Pulse Stretcher Circuit

© 2001-2010 Tom A. Wheeler

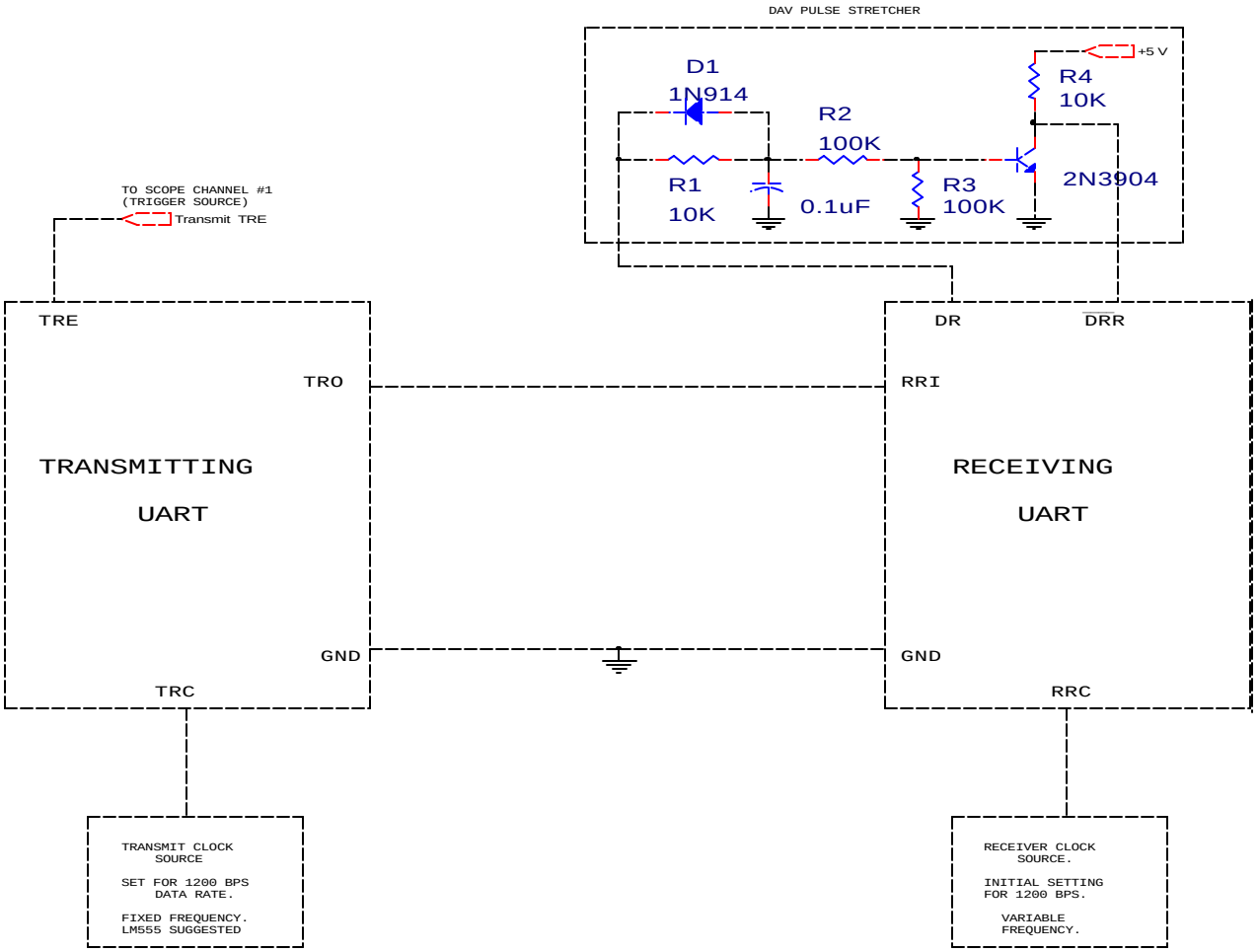

Figure 2: Experimental Circuit Layout

© 2001-2010 Tom A. Wheeler

LABORATORY PROCEDURE:

Name _____ Sign-off _____

1. A laboratory partner is required for this experiment. One of you will be the transmitter, and the other will be the receiver. Arrange the circuits as shown in Figure 2.

   The transmitter must use a fixed-frequency oscillator built from an LM555 timer IC. (A typical circuit is given at the end of the experiment.)

   The receiver circuit must have the pulse stretcher circuit, and will use the benchtop signal generator as a variable frequency clock source.

   > ***Don't forget to connect the grounds of both circuits together!***
   >
   > ***Make sure that the RRI and TRO pins of the receiver are not connected together, to prevent contention on the TRO signal.***

2. Let's observe the entire transmission and reception process, including the receiver "handshake." *During this entire step, channel 1 of the oscilloscope will be connected to the TRE signal of the transmitting UART. The transmitter's TRE signal will be the timing reference for all the other measurements.*

   In this step you will produce a graph with four signals: Transmit TRE, transmit TRO, receiver DR, and receiver DRR. Use a separate sheet of graph paper to record these waveforms.

   a) Reset both UARTs.

   NOTE: Resetting the receiver UART while the transmitter is running may produce invalid data at the receiver.

   b) Start transmitting A5H using parameters 8,N,1

   c) Using the transmitter *TRE* as a trigger reference, record on your graphs the transmitter's *TRE* signal (1 cycle, using channel 1), and the transmitted data byte from *TRO (channel 2)*. (This is also the same as the *RRI* signal at the receiver.)

   d) Exactly when does *DR* turn on in the reception cycle? Find out by moving the "free" scope channel (channel 2) to the receiver's *DR*. Record this in reference to the *TRE* signal of the transmitter.

   e) How does the *DRR* signal look? Record it in reference to the transmitter *TRE*. (Again, move channel 2 to *DRR* to see it in relation to the transmitted *TRE*).

© 2001-2010 Tom A. Wheeler

3. How close must the receiver clock frequency be to that of the transmitter? Let's find out:

a) Set the transmitter and receiver for 8 data bits, Even parity, 1 stop bit.

b) Make sure the transmit rate is very close to 1200 bps. Use a frequency counter to verify. (Don't forget that the clock must be 16 times the data rate!)

c) Start the transmission-reception process as in step 2 above.

d) Set the receiver clock frequency as close to the transmitter clock frequency as possible. Then, very slowly increase the receiver clock frequency until an error occurs. An error could be either:

-An outright invalid data display

-Or an error LED flickering or lit steadily.

Record the maximum clock frequency at the receiver that could be used without generating a receiver error.

e) Repeat step (d), except *decrease* the receiver clock frequency. Again, record the minimum clock frequency that could be used without causing a reception error.

f) Express the maximum *percent difference* for the frequencies obtained in steps (d) and (e). A figure reported as a percentage is often more applicable when it is considered that not all UART systems run at the same rate. The percentage of frequency deviation is calculated by:

$$\%\ \text{difference} = \frac{F_{actual} - F_{nominal}}{F_{nominal}} \times 100\%$$

Where F_{actual} = the variable clock frequency (that of the receiver),
$F_{nominal}$ = the nominal or "named" clock frequency (that of the transmitter).

This formula is easy to remember if you think of the following example:

A car has the cruise control set for 100 MPH. It actually travels at 105 MPH. What is the percentage difference between the setpoint and actual speed?

$$\%\ \text{difference} = \frac{F_{actual} - F_{nominal}}{F_{nominal}} \times 100\% = \frac{105MPH - 100MPH}{100MPH} \times 100\% = \underline{\underline{+5\%}}$$

Maximum error-free percent difference of clocks: _____ %

© 2001-2010 Tom A. Wheeler

Typical LM555 / NE555 Application Circuit (Astable Multivibrator)

The circuit of Figure 3 can be used to build a fixed-frequency clock oscillator using the LM555 timer IC.

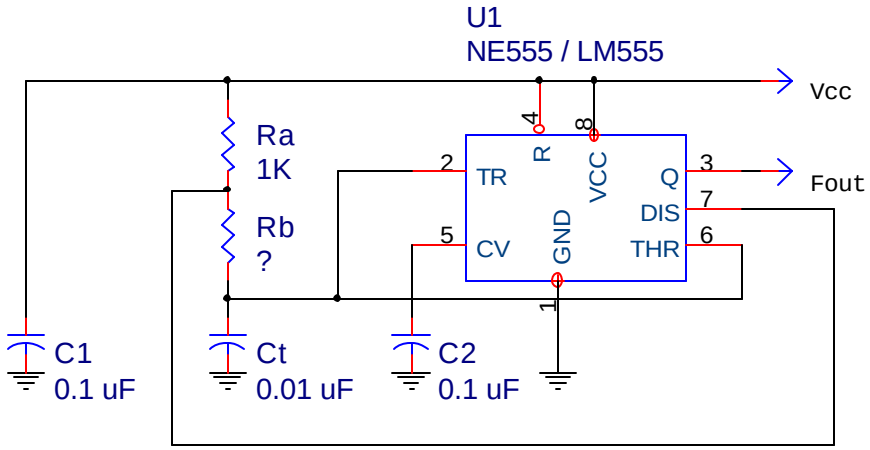

Figure 3: LM555 in Astable Mode

In this circuit, C1 and C2 are used as power supply filter (bypass capacitors). Components Ra, Rb, and Ct set the output frequency as follows:

$$f_{out} = \frac{1}{0.693(R_a + 2R_b)C_t}$$

Starting values for Ra and Ct are given in the circuit above. To find Rb, substitute the known values for Ct and F_{out} into the equation above and solve. Note that the duty cycle of the output will not be 50%, but this makes no difference to the UART, since the UART internally divides the *TRC* and *RRC* clocks by 16. You may wish to use a variable resistor for Rb so that the frequency can be adjusted precisely.

© 2001-2010 Tom A. Wheeler

Questions

1. In order for a UART to successfully receive data, what three things must be true?

   1) _____

   2) _____

   3) _____

2. Explain what a *handshake* is, and why it is needed in communications circuits.

   _____

   _____

   _____

3. Did you have to perform any troubleshooting during this experiment? Explain.

   _____

   _____

   _____

© 2001-2010 Tom A. Wheeler