Lab #9: RS-232 serial communication

Physics 127BL Winter 2024

Lab report due **Thursday, March 14, at 11:55 P.M.**

Please read the lab report and homework guidelines handout on the course web page.

Introduction

In this lab, we will use the venerable RS-232 serial communication protocol to perform single-byte transfers to and from our FPGA boards, and learn in detail how the protocol works. In our next lab we will use RS-232 to download small files into the FPGA memory.

1 Loopback interface

Load the `lab9` project from the course web page and open the top level block diagram `RS232.bdf`. You'll see a simple loopback interface that sends the RS-232 input directly to the output, where it returns to the connected computer. The loopback interface is controlled with a slide switch. You'll also find an RS-232 decoder that has been created for you, to save the input bytes and display them on the HEX displays and green LEDs. "UART" stands for "universal asynchronous receiver-transmitter," the logic device that implements the RS-232 protocol.

1. To test the interface, open a serial terminal window on your computer using any *terminal emulator* software. The open-source programs `minicom` and `PuTTY` should both work. `minicom` is installed on the lab computers, and `PuTTY` can be downloaded here.

   To establish the connection, use the following settings:

   Name = your name
   Protocol = serial
   Port = `/dev/ttyUSB0`

   Baud rate = 115200
   Data bits = 8
   Parity = none

   Stop bits = 1
   Flow control = RTS/CTS.

   Note: You will need to change "Port" to `COM1` or similar if using Windows.

2. Look through the FPGA code to familiarize yourself with the decoder. Load the code into the FPGA and turn on the loopback interface with the slide switch. Take note of what lights up when the RS-232 connection is made. Press keys in the terminal emulator window and you should see them echoed. The terminal is displaying data received from the serial port, not what you are typing. When you flip the loopback switch, you should still see key presses displayed on the FPGA board, but nothing in the terminal.

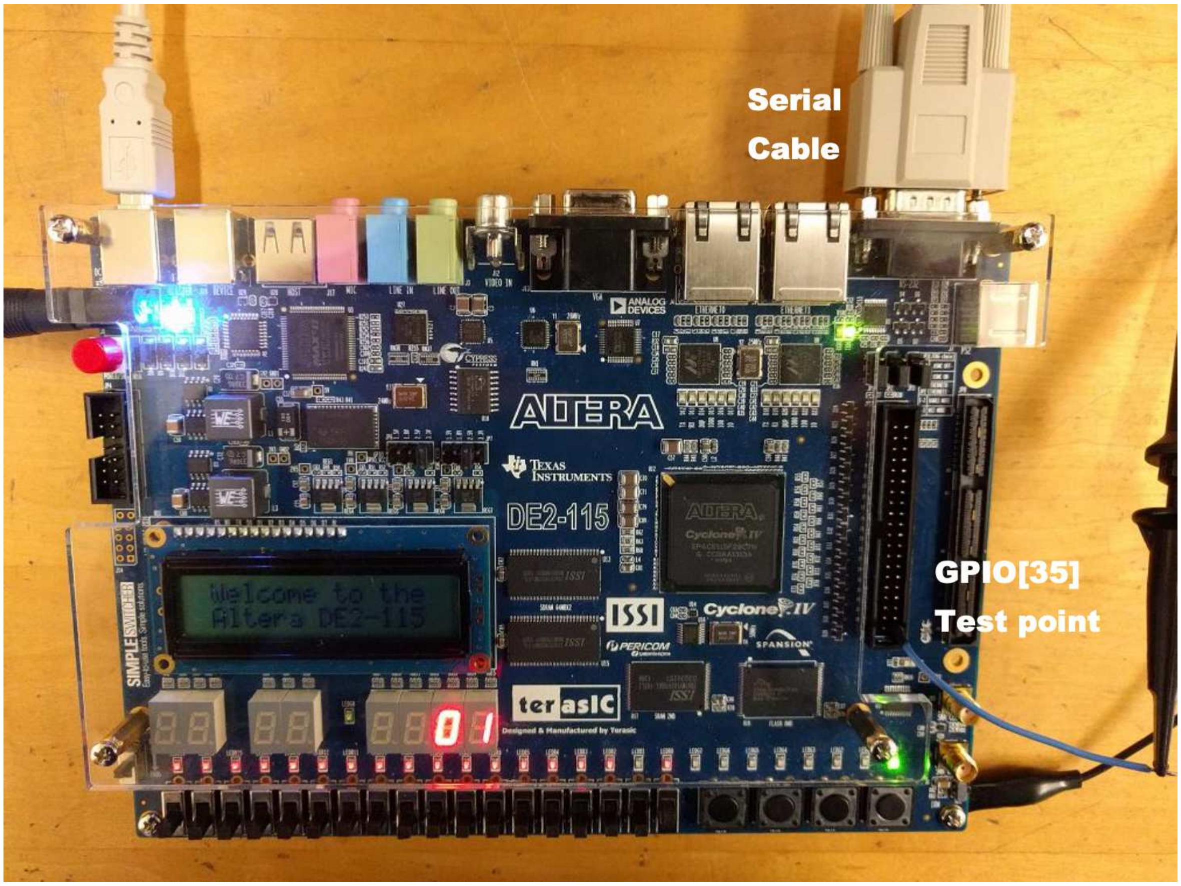
3. Look at the two small LEDs next to the serial port, `RX` and `TX`. Hold down a key. Which of the two lights up? One or both? Flip the loopback switch and try again.

4. Try a few key presses and note how they are decoded in binary. You can check the correspondence here (scroll down for printable characters). Try `<Ctrl+A>` = 00000001_2, `<Ctrl+B>` = 00000010_2, `<Ctrl+D>` = 00000100_2, and `U` = 01010101_2. Why would the characters not be decoded correctly if you set Parity to "ODD"? Try it.

2 Visualizing serial data transfer

We will now view the serial data signal directly with the oscilloscope!

1. Using a jumper wire from your electronic parts kit or the lab, connect an oscilloscope probe to general-purpose input/output pin `GPIO[35]`, as shown in the photo below and in Fig. 4-15 on page 46 of the FPGA board manual on the course web page. Use the metal standoff on the bottom of the board as a ground connection.



2. Set the oscilloscope to trigger on your key presses so it keeps your data packets on the display for easy analysis. What are the waveform characteristics? Derive the 115200 baud (bits per second) rate from the scope data. What defines the start and stop of the signal? How can you tell it is the letter you typed? Make sure you understand the output level for the start bit, the order of transmission of the bits, polarity of the bits, etc. See here and Horowitz & Hill §12.10.4 for more information about the signal.

3. Try changing the baud rate at the terminal and send packets again. Why are they no longer decoded correctly?