ELEC 327 (2014) Lab #3: Communicating with the world (serial data transmission)

Part 1: Communicate with computer via serial (DO THIS FIRST!!!!)

In this lab, you’ll talk to the computer via serial communication. You’ll be using the hardware UART module on the MSP430. You can keep your MSP430 on the Launchpad for this (you’ll be using the Launchpad power anyways). You’ll be using hardware serial in both parts, so you need to rotate the TX and RX jumpers to the “HW” position. If you’re confused as to what this means, make your MSP430 pins look like those on figure 2 of this tutorial.

a) In general, what does it mean to use hardware vs software serial communication. Is the UART protocol synchronous or asynchronous? What does being synchronous or asynchronous mean? Your answer to this should be able to explain whether or not you are tied to a particular clock speed when communicating via UART. Don’t just take blind guesses at this; read chapter 15 of the user guide (or at least parts of it). You shouldn’t need to touch your MSP430 to answer questions like these.

b) Write a program (does not need to be turned in) that reads serial input characters into an array until an end-of-line character is received (“\n”) and echoes those received characters back at the computer through the serial line. For instance, if you receive “hello\n”, you should send “hello” back.

   a. Feel free to look up how people implement serial communication to get a feel for the setup steps you need to take. If you do this, though, you should look up what each line of the setup does in the user guide (each variable is explained fairly well in the serial communication chapter). If you don’t understand how to modify this setup, you won’t know how to connect to any other devices (like those in your midterm project, potentially).

   b. When you receive serial data, it goes into UCAxRXBUF. When something is in UCAxRXBUF, the serial receive interrupt is fired. You have to actually read UCAxRXBUF in your interrupt routine, or else the serial receive interrupt will be continuously called.

   c. You send values back to the computer by putting values in UCAxTXBUF. However, as you can imagine, rapidly putting values in there wouldn’t guarantee that it would be read by the other end by the time you put the next value in. Use a line like

      while (!(IFG2 & UCA0TXIFG));

      to hang your program until it’s ready to send another value.

   d. You’ll also need to get a serial terminal on your computer to test this. Good ones to try are TeraTerm or RealTerm, but you can use whichever one you want.

Part 2: Controlling “NeoPixels” with MSP430

For this part, we’ll be using NeoPixels: strands of LEDs that can be controlled with one data wire in a serial-like manner. The control is serial-like since there is no built in hardware to do
communication like this, so you must code the protocol yourself. Adafruit has an entire wiki explaining NeoPixels (http://learn.adafruit.com/adafruit-neopixel-uberguide/overview). We’ve included a “skeleton file” that sets the clock speed to 16MHz and uses an assembly subroutine to set the NeoPixel colors. Cut yourself a strip of 5 NeoPixels (cutting after the last LED’s capacitor), solder 3 wires to the 3 signal lines, connect the power pins to your launchpad’s power with female-female wires, and connect the data pin to P1.7 on the launchpad (P1.7 is the pin the given assembly code uses, though any digital pin can work). Set up a new project with the given .c and .asm files in the workspace, and verify that your strip is running correctly.

Play around with this program and try to change your LEDs to be all green, all red, all blue, or a mix of colors (you don’t need to turn these in).

Here are some tips, guidelines, and other useful info for the next part:

- The NeoPixels draw a fair bit of current. As far as we can tell, at maximum intensity with 3.6V power, the maximum current draw for these 5 LEDs is just less than the 100 mA maximum the Launchpad can provide when running on USB power. However, you may want to set up an external power source (i.e., 5 V DC). There are breadboard-compatible jacks and 3-pin regulators to power your MSP430 from the same supply. If you want to use the USB instead, make sure to connect the grounds of the two power supplies.
- You need to run the MSP430 at 16 MHz.
- For a strand of 5 LEDs, the assembly routine acts on an array of length 15 (one entry for each color of the LED). The first entry in the array is the LED farthest down the line, and the 3 entries for each LED go in the order “GRB” (NOT “RGB”). You can manipulate this array however you like, just keep this syntax that the assembly routine expects in mind.
- We are providing an assembly language function that runs the NeoPixel in “slow” 400 KHz fallback rate. **For extra credit you can re-implement this function to work for the “fast” 800 KHz mode (see the adafruit page for proper timing details for this mode – there’s an error in the datasheet).**

**Part 3: Controlling NeoPixels with a computer**

With the previous section done, you can set colors by hard coding it into your program. In this part, you’ll develop your own communication protocol to control the MSP430 and your strand via a serial connection. In particular, your program should be able to process and execute the command set outlined below. For extra credit, program in dynamic sequences as
well (i.e., colors rotating, changing, flashing, etc). Submit the final code as neopixel.c, and
demo this final solution.

Command Set (case insensitive)

1.) “set(PIXEL,G,R,B)” (PIXEL = {1,2,3,4,5,or 0 = all) assume maximum of 5 neopixels,
decimal value for each color between 0 and 255)
Your MSP430 should in general keep track of the values of the array of neopixels. This
command should change the value of one of the neopixels (or all if the value 0 is
specified). Note that the “PIXEL” and “RGB” values will be the ASCII character ‘1’, ‘2’, ‘3’,
etc!
2.) “black” / “fade off”
Set all neopixels off. For extra credit have them reduce intensity slowly to fully off
when the “fade off” command is specified. Note that this command should not erase
the current array (these values will be used for the “go” command).
3.) “go” / “fade on”
Display pre-stored pixel value array. If neopixels are already on, this command can be
ignored.
4.) “blink” / “noblink”
Cause Neopixels to blink on and off. “noblink” should return the array to solidly on.
5.) Come up with something else creative
One example would be “running right” / “running left” (assuming an array of 5, colors
move up or down array.) Fades, fancy rainbows, etc.