OBJECTIVES

In this lab you will perform the first of several physical expansions of your EEL 4744 uTinkerer PCB by adding LED and switch circuits. You will also exercise your programming skills and familiarize yourself with the Digilent Analog Discovery (DAD) board's oscilloscope function.

REQUIRED MATERIALS

- Reread *Lab Rules and Policies* document
- EEL 4744 (uTinkerer PCB) Board kit and tools
- Soldering iron, wire wrap, and wire wrap tool
- 1 - DIP LED package
- 1 – DIP Switch package
- 2 – SIP Resistor packs (470Ω and 1.2kΩ)
- Digilent Analog Discovery (DAD) kit

PRELAB REQUIREMENTS

REMEMBER:
You must adhere to the *Lab Rules and* Policies document for **every** lab. Re-read, if necessary.

All four parts (LED and Switch DIPs, and resistors) should be place on the top of your PCB; all wire wrapping will occur on the bottom.

Check with your TA **BEFORE** soldering any parts to ensure correct placement.

Getting Started
Look closely at your EEL 4744 uTinkerer Board expansion area. Find the pins in the white box that has three rows. Use your multimeter to measure the resistance between pins in the same column. Notice that the resistance is very low; that is because the three pins in the same column are connected. The same is true for the pins in the white box that has two rows, i.e., pins in the same column are connected.

Power and ground should generally be connected by solder using low gauge (large diameter) wire, as compared to wire wrap wire (small diameter). In this course, because we are running our circuits at low speeds, I will allow you instead to double wrap power and ground connections, i.e., use two separate wires to make two connections for your power and ground when they provide power to a circuit. When power or ground is a default signal value, a single wrap is sufficient (e.g., if a 3-input AND-gate is used for a 2-input AND function, Vcc can be single wrapped to one of the inputs).

ALWAYS remove power when soldering or wire wrapping.

Part A
In this section you will design a program to read 8 input switches connected to Port D and echo (i.e., output) their value to 8 LEDs connected to Port F. This will repeat over and over again in an endless loop.

Create a flowchart or pseudo-code of the desired algorithm **BEFORE** writing any code.

Whenever possible, you should simulate your design **before** you try it on your board. You can verify the correctness of your code in this lab with the Atmel Studio Simulator before the addition of any hardware.

1. Write a small program to write to Port F. Test your program with the simulator.
2. Write a small program to read from Port D. Test your program with the simulator.
3. Combine these two programs into a third program that reads from Port D and outputs the values read to Port F. Test your program with the simulator. This is your part A.

After verifying your programs with the simulator, you are now ready to design your circuits. You should have learned how to build LED circuits and switch circuits in EEL 3701. If necessary, see the below document for a refresher:

http://mil.ufl.edu/3701/docs/hardware_get_started.pdf .

Design (on paper or computer) 8 LED output circuits to connect to Port F and 8 switch circuits to connect to Port D. These circuit designs should be added to your pre-lab submission. Use a SIP resistor pack for the LED circuits and design your circuit so that the LEDs are lit when the XMEGA output is high. Design pull-up switch circuits using another SIP resistor pack.

After you have verified that your circuit designs are correct, you are ready to start building on the uTinkerer PCB.

A suggested LED and switch circuit layout is shown in Figure 1, utilizing the *DIP breakout island* between headers J34 and J35. The yellow vertical lines that appear in the top drawing in Figure 1 represent traces (wires) inside the PCB that you cannot see. Note that one side of the LED goes into the top row of the lower white box (that has three rows of pins) and the other side of the LED goes into the bottom row of the upper white box (that have two rows of pins). Use the 470Ω SIP pack for the LED circuits and 1.2kΩ SIP pack for the switch circuits.

To construct your LED circuit, you will need to utilize a single row wire wrap header and a single wire wrap header pin. Solder the LED circuits in place by soldering **ONLY ONE** of the 8-LED circuits. A suggested layout is shown in Figure 1. Solder this circuit in place and add the necessary wire wraps.

After constructing your single LED circuit, test it with your first small program to verify that the single LED lights up appropriately. If not, analyze and then repair your circuit. When your single LED circuit works properly, you should

then complete the construction of your 8 LED circuits. Then test your 8-LED circuits with your first program.

To construct your switch circuits, you will need to utilize two single row wire wrap headers, as shown in Figure 1. Solder and wire wrap a **SINGLE** switch circuit and then test this with your second small program. When you are satisfied that your single switch circuit functions properly, complete your switch circuit construction. Then test your 8-switch circuits with your second program.

Now test your circuits with your third program that echoes your inputs switch values to your output LEDs. This is the program you must submit for part A of this lab.

Note: The wire wrapping takes a fair amount of time, so start early!

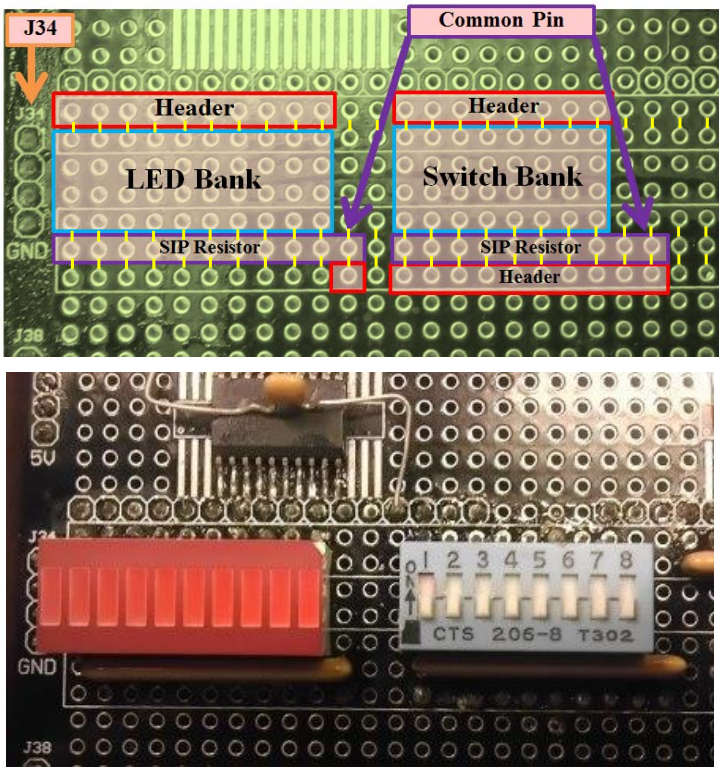

Figure 1: A possible layout of added hardware. An even better idea would be to shift the LED Bank and LED Header one pin to the right and reverse the SIP resistor so that the common pin is on the left (instead of the right), nearer the GND. I also suggest a similar change for Switch Bank.

Part B

In this part of the assignment, you will write a program that will blink a single LED at a frequency of 2 kHz. You will then measure the frequency at which the LED blinks using the DAD oscilloscope function.

Create a flowchart or pseudo-code of the desired algorithm **BEFORE** writing any code.

You can start writing this program by designing a delay program fragment (or subroutine) (Delay2k). Since your board runs at 2 MHz, it would be a good assumption to say that each instruction (on average) takes 0.5 μs = 1/(2 MHz). Use this assumption to write a program that blinks an LED at 2 kHz by using/calling Delay2k.

The waveform that is output to Port F will blink a single LED (of your choice) with a square wave with a 50% duty cycle. (The LED will be on for half the period and off for the other half). Observe this waveform using the DAD oscilloscope function.

The waveform can be quickly displayed with the DAD using the ***AutoSet*** option next to the 'Run' button in the oscilloscope window. The ***AutoSet*** function is not always reliable, so it is useful to know how to manually adjust the oscilloscope. Alter the values in the boxes labeled ***Time***, ***C1***, and ***C2*** in the far right column and note the effect of each of these controls.

Use the ***Measurements*** tool under View (or press Ctrl+M) to display the values of the average frequency and the average period of the waveform. Take note of any other measurements in the ***Measurements*** tool that may be useful.

Modify your Delay2k subroutine to obtain a frequency as close to 2 kHz as possible. Obtain a screenshot from your laptop with the DAD oscilloscope function displaying the 2 kHz waveform, the average frequency, and the average period. The screen shot should also display your name in big letters. Save this screen shot in the Appendix of your pre-lab report (as stated in the *Lab Rules and Policies*, part 6, item i).

It would be useful to make a general purpose program that can delay a select number of milliseconds. It is not required, but I suggest that you make a subroutine called DelayX10ms, where X will be a number passed into the DelayX10ms subroutine in a resister, e.g., R16. The delay should be the number in the register × 10 ms. This would give you delays from 10ms through 2.550 s.

Note for students who do not have a DAD: You can utilize the lab's DAD in a TA lab office hour **before** your lab to get the required information and screenshot. But if you cannot borrow a DAD for your lab (from someone **NOT** in your lab section), you should **also learn** how to use one of the lab's **actual oscilloscopes** in order to get most of the same information (since you may need to use that device during your lab). You may have to take turns using one of the lab's two oscilloscopes or the lab's single DAD.

Part C

In this part of the assignment, you will write a program to read the input switch values and perform specific functions depending on the state of the inputs.

It is best to break the large programs into smaller pieces; build up your program as you progress, rather than designing the entire program at once. I suggest that you use this technique for the rest of the course (and for the rest of your life). (This is the ***Be the Tortoise and not the Hare*** approach to problem solving.)

The program requires you to do the following:

1. If Port D bit 5 is set, do the following:
   - Output a 1 to the upper two bits of Port F (bits 6, and 7).
   - Shift the bits to the right approximately every 0.370 seconds. The output should 'rotate' so bit 0 would go to bit 7, i.e., 1100 0000, 0110 0000, …, 0000 0110, 0000 0011, 1000 0001, 1100 0000, …
   - Repeat forever, checking for changes in Port D between each change in outputs.
2. If Port D bit 5 is clear, do the following:
   - Put a 1 in Port F bit 0, with all other bits as 0.
   - Shift the 1 left once every 0.370 seconds (with all the other bits showing 0). Once the 1 reaches bit 7, shift the 1 right every 0.370 seconds until it reaches bit 0 again, i.e., 0000 0001, 0000 0010, …, 0100 0000, 1000 0000, 0100 0000, … ,0000 0010, 0000 0001, …
   - Repeat forever, checking for changes in Port D between each change in outputs.

For simplicity, you may assume that when the value of Port D bit 5 changes, the new operation should continue from the present state. Alternatively, when the value of Port D bit 5 changes, you may start from the beginning situation, 1111 0000 or 0000 0001, if this would be easier for you.

Design Procedure:
In general, flowcharts or pseudo-code are very useful, especially before writing large programs.

1. Create a flowchart or pseudo-code of the desired algorithm **BEFORE** writing any code.
2. You can start writing this program by designing a 0.370 second (=370 ms) delay program fragment (or subroutine) (Delay370ms). As was done in Part B, use the DAD oscilloscope function to make the delay subroutine as close to 0.370 seconds as possible.
3. Write a program fragment/subroutine (KnightRider1) that performs the operation described above when Port D bit 5 is set, delaying 0.370 s between changing outputs (with Port F connected to LEDs).
4. Write a program fragment/subroutine (KnightRider2) that performs the operation described above when Port D bit 5 is clear, delaying 0.370s between changing outputs.
5. Use Port D, bit 5 to determine which of the two program fragments/subroutines (KnightRider1 or KnightRider2) should be performed.

Assemble and simulate all programs **BEFORE** coming to lab. Even if you cannot get the entire Part C program functioning properly, you can earn partial credit if you can demonstrate some of the above programs in the design procedure above. It is a good idea to program the boards ahead of time and verify that both programs work. If they do not, utilize the JTAG debugging capability to fix any errors in your code.

LAB PROCEDURE

Demonstrate your lab2 Part A (echo) program on your board.

Demonstrate your lab2 Part B program and that you are able to measure the waveform with the DAD oscilloscope.

Demonstrate your lab2 Part C program on your board. If your lab2 Part C code does not work, demonstrate the portions that do work correctly.

If your code does not work, utilize the JTAG debugging capability along with your multimeter, DAD board, and electrical and computer engineering knowledge to fix any errors in your hardware and/or software (code).

Preparation for Future Labs:

1. If you have not already done so, insert the CPLD into its designated socket. Test that you can program the CPLD without problems. The TA's can provide a test program that will cause no damage (i.e., do nothing).
2. Solder your 74'573 and 74'574 chips to the surface mount area of your uTinkerer, as described below. Figure 2 shows the placement of the 74'573 and 74'574.
   a. Place the 74'573 surface mount chip in the left surface mount position just above your LED bank, as shown in Figure 2. Begin by soldering the two corner pins to chip in place. Add two single row 10-pin headers on either side (above and below) the 74'573. You will wire wrap to these header pins in lab 3. Solder a 0.1μF bypass capacitor between the header pins corresponding 74'573's Vcc and GND pins, i.e., pins 10 and 20 (as shown in Figure 3). Before soldering the capacitor, trim the leads appropriately. Be careful that the capacitor leads will not touch any non-desired pads or pins.

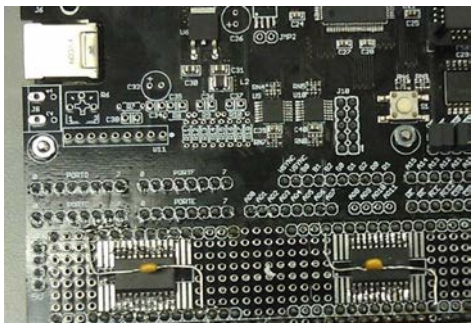

Figure 2: Placement of 74'573 (bottom left) and 74'574 (bottom right) on uTinkerer.

b. Place the 74'574 surface mount chip in the middle surface mount position (to the right of your 74'573). Solder the 74'574 as described above for the 74'573. After the chip is in place, solder a 0.1μF bypass capacitor between the header pins corresponding to the 74'574's Vcc and GND pins, i.e., pins 10 and 20. Add two single row 10-pin headers on either side (above and below) the 74'574.

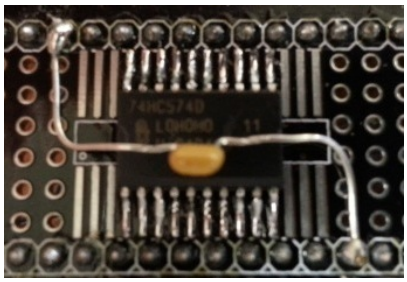

Figure 3: Example capacitor placement across a chip.

3. If you have time, solder the 32k x 8 SRAM in place as shown in Figure 4.

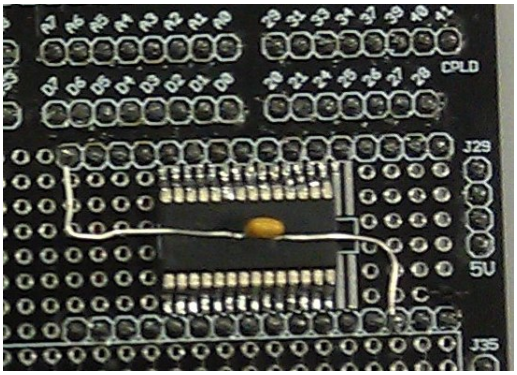

Figure 4: SRAM placement on uTinkerer.

REMINDER OF LAB POLICY

Please re-read the *Lab Rules & Policies* so that you are sure of the deliverables (both on paper and by email) that are due prior to the start of your lab.