Lab 2: First Board Expansion (Switches and LEDs)  

**OBJECTIVES**
In this lab you will perform the first of several physical expansions of your EEL3744 board, the uPAD Proto Base, 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 3744 (uPAD and uPAD Proto Base) kit and tools
- Soldering iron, wire wrap, and wire wrap tool
- 1 - DIP LED package
- 1 – DIP Switch package
- 1 – 470 Ω SIP Resistor pack
- 1  – 2.2 kΩ SIP Resistor pack
- 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 placed 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 uPAD Proto Base breadboard area (near the bottom of the board). There are two outlined regions, each with 48 numbered columns and 3 rows. In the aforementioned regions, each column of 3 holes are electrically connected, i.e., shorted together. These internal connections are indicated by lines on the board’s silkscreen.

Power (Vcc = 3.3 V) and ground should generally be connected by solder using low gauge (large diameter) wire, as opposed to wire wrap wire (small diameter). In this course, because we are running our circuits at low speeds, you are allowed 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 (3.3 V) can be single wrapped to one of the AND gate inputs).

**ALWAYS** remove power when soldering or wire wrapping.

**ALWAYS** 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 your code in this lab with the Atmel Studio Simulator before the addition of any hardware.

Prior to your lab, if a program/design does not work, utilize the Atmel Ice via PDI debugging capability, along with your DAD board, and your prior electrical and computer engineering knowledge to fix any errors in your hardware and/or software (code). This should occur **BEFORE** you come to lab. Visit a TA or Dr. Schwartz, if necessary, but come to lab prepared!

**Part A**
In this section you will ultimately design a program to continuously read 8 input switches connected to Port F and echo (i.e., output) their value to 8 LEDs connected to Port E. As always, you should create this design by breaking it in to smaller designs first and solving each of these designs in turn.

1. Write a small program to write to Port E. Test your program with the simulator.
2. Write a small program to read from Port F. Test your program with the simulator.
3. Combine these two programs into a third program that reads from Port F and outputs the values read to Port E. 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 circuits to connect to Port E (used as an output port) and 8 switch circuits to connect to Port F (used as an input port). 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 an LED is lit when the corresponding XMEGA output is high. Design pull-up switch circuits using another SIP resistor pack.

After you have verified that your circuit designs are complete, you are ready to start building on the uPAD Proto Base.

A suggested LED and switch circuit layout is shown in Figure 1, utilizing the uPAD Proto Base breadboard area. The white vertical lines that appear in the breadboard area in Figure 1 represent traces (wires) inside the PCB that you cannot see. Note that the side of the LED with text on it (the anode side) faces the uPAD and is aligned on the breadboard as shown in Figure 1, i.e., starting in the fourth column of holes. Use the 470 Ω SIP resistor pack (highlighted in purple in Figure 1) for the LED circuits.
The common pin of the SIP resistor pack is on the right, above the green dot.

To construct your LED circuit, you will need to utilize an 8-pin single row wire wrap header (yellow in Figure 1) for the 8 port signals and 1-pin single row wire wrap header (green) for the common SIP resistor pin. (Instead of the 1-pin header, you can just make the adjacent switch’s 9-pin header [red then purple] into a 10-pin header.). 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 the emulator or with your small output program. If the LED does not light up as desired, analyze and then repair your circuit. When your single LED circuit works properly, you should then complete the construction of the rest your LED circuits and test them the emulator and then with your small output program.

To construct your switch circuits, you will need to utilize two single row wire wrap headers (one 8-pin and one 9-pin or 10-pin), as shown in Figure 1 (green on top and purple with red on bottom, where red is the common SIP resistor pin). Solder and wire wrap a SINGLE switch circuit and then test this with your small input 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 only program that you must submit for part A of this lab.

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

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.

Note: 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) (DELAY_250us). 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 2.000 kHz, i.e., by using/calling Delay250µs.

The waveform that is output to Port E will blink a single LED (of your choice) with a square wave with a 50% duty cycle, as shown in Figure 2, where X is 250 µs. (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 DELAY_250us 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 register, e.g., R16. The delay should be the number in the register × 10 ms. It is okay if the largest allowable value is 127 giving a maximum delay of 1.27 s, but a maximum delay of 2.55 s is also allowable (with an allowable value up to 255). The minimum delay should be 10 ms.

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 F bit 6 is clear, do the following:
   - Output a 1 to bit 0 and bit 1 of Port E and 0’s to all the other bits. Start with 0000 0011.
   - Shift the bits to the left approximately every 0.240 s. The output should ‘rotate’ left so bit 0 would go to bit 7, i.e., 0000 0111, 0000 0110, ..., 1100 0000, 1000 0001, 0000 0011, 0000 0110, ...
   - Repeat forever, checking for changes in Port F between each change in outputs.

2. If Port F bit 6 is set, do the following:
   - Put a 0 in Port E bits 3 and 4, with all other bits as 1, i.e., 1110 0111.
   - Shift the zeroes one bit every 0.420 s as follows: 1110 0111, 1100 0011, 1000 0001, 1100 0011, 1110 0111, 1100 0011, ...
   - Repeat forever, checking for changes in Port F between each change in outputs.

When the value of Port F bit 6 changes, start from the beginning situation, i.e., 0000 0011 or 1110 0111.

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 240 ms delay program fragment (or subroutine, DELAY240ms). As was done in Part B, use the DAD oscilloscope function to make the delay subroutine as close to 0.240 s as possible.
3. Write a program fragment/subroutine (KITT_1) that performs the operation described above when Port F bit 6 is clear, delaying 0.240 s between changing outputs (with Port E connected to LEDs). [KITT stands for Knight Industries Three Thousand.]
4. Write a program fragment/subroutine (KITT_2) that performs the operation described above when Port F bit 6 is set, delaying 0.420 s between changing outputs.
5. Use Port F, bit 6 to determine which of the two program fragments/subroutines (KITT_1 or KITT_2) 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 programs in the design procedure above.

LAB PROCEDURE
Demonstrate each of the following using your boards and your DAD

Demonstrate your Part A (echo) program.

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

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

Preparation for Future Labs:
1. If you have not already done so, insert the CPLD into its designated socket (U2). Be careful to insert it properly, i.e., so the text should be oriented aligned as shown in Figure 3. Test that you can program the CPLD without problems.

![Figure 3: Placement of CPLD.](image-url)
2. Solder your 74’574 and 74’573 chips to the surface mount area of your uPAD Proto Base, as described below. Figure 3 shows the locations of the three SOIC (Small Outline Integrated Circuit) pads.
   a. Locate the left SOIC surface mount position just above your LED bank, as shown in Figure 4. Begin by applying flux the solder pads. Place the 74’574 surface mount chip with the notch oriented towards the left. The notch indicates that pin 1 of the chip is on the bottom left pad, as shown in Figure 5. Solder the two corner pins to chip in place.
      i. Add two single row 8-pin headers on each side (above and below) the 74’574. You will wire wrap to these header pins in lab 3.
      ii. Solder a 0.1µF bypass capacitor in the bypass capacitor area (on the left, labeled C7, as shown in Figure 6).
   b. Place the 74’573 surface mount chip in the middle SOIC surface mount position (to the right of your 74’574). Solder the 74’573 as described above for the 74’574.
      i. Add two single row 8-pin headers on each side (above and below) the 74’573. You will wire wrap to these header pins in lab 3.
      ii. Solder a 0.1µF bypass capacitor in the bypass capacitor area (labeled C8, as shown in Figure 6).

3. If you have time, solder the 32k × 8 SRAM in the right-most SOIC location. You will use the SRAM in lab 4.

REMINDER OF LAB POLICY
Please re-read the Lab Rules & Policies so that you are sure of the deliverables (both on paper and through Canvas) that are due prior to the start of your lab.