Lab 2: GPIO Expansion (LEDs and Keypad)

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 keypad 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)
- Digilent Analog Discovery (DAD)

PRELAB REQUIREMENTS

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

Getting Started
ALWAYS create a flowchart or pseudo-code of the desired algorithm **BEFORE** writing any code.

All devices (including the LED DIP, switch DIP, SIP resistor(s), keypad connector, etc.), should be placed on the top of your PCB; all wire wrapping will occur on the bottom of the PCB. **If you have not already mounted these, you should do so prior to the start of this 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
All of the more than dozen XMEGA 8-bit ports can be used as general purpose inputs or outputs (GPIO). In this section you will ultimately design a program to output to 8 LEDs connected to Port E.

First, you need to design the LED circuits. You should have learned how to design and construct LED 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) eight active-high LED circuits to connect to Port E (with Port E configured as outputs). Include this circuit design to your pre-lab submission. Use a SIP resistor pack for the LED circuits and design your circuits so that each LED is lit when the corresponding XMEGA output is high, i.e., design active-high LED circuits.

If you have not already done so, add the LED circuit to your uPAD Proto Base as shown in Figure 1. After constructing your LED circuit, wirewrap one of the pins from Port E to one of the LED circuit inputs. Test this LED with the emulator using a small output program called **`lab2a.asm`**. If the LED does not light up as desired, analyze and then repair your circuit. When you are confident that this single LED is working correctly, it is safe to wirewrap the rest of Port E to the other LED circuit inputs. At this point you should test all of your LEDs with the emulator and the small output program **`lab2a.asm`**. It is required that you **always** create a flowchart or pseudo-code of the desired algorithm **BEFORE** writing any code. All flowcharts or pseudo-code should be included your submitted lab document (as stated in the *Lab Rules and Policies*.

Remember that before using a GPIO, you have to set up the data direction register (PORTx_DIR) as either an input or an output. (Additional options are also available.) To output to a GPIO port, available registers are PORTx_OUT, PORTx_OUTCLR, PORTx_OUTSET, and PORTx_OUTTGL. See the GPIO Output example program (GPIO_Output.asm) on our website.

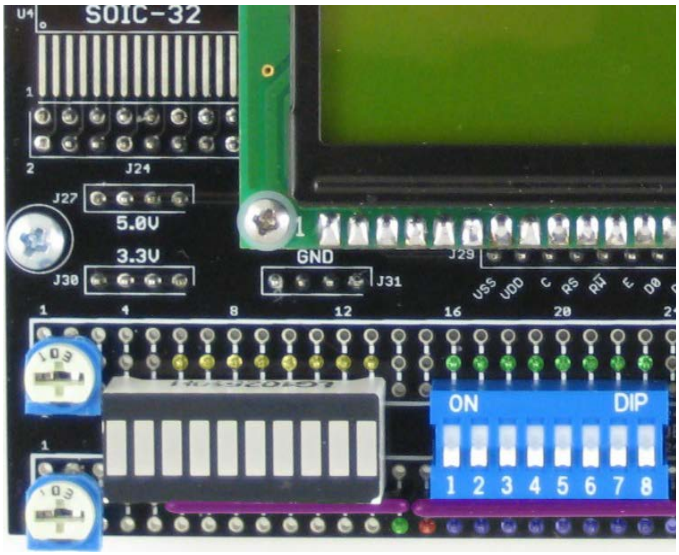

Figure 1: A possible layout of added hardware. Yellow is the 8-pin header for LED signals, green are ground pins, red is Vcc (3.3 V) and blue pins are switch signals. The SIP resistors are highlighted purple.

Part B
In this part of the assignment, you will write a program that will blink the LEDs such that the top and bottom nibbles alternate (i.e., the left four LEDs are on and the others off, followed by the right four LEDs on and the others off: 0xF0, 0x0F, 0xF0, …) at a frequency of 3.00 kHz. You will then measure the frequency at which any of the blinking LEDs blink 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_167us). 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 the LEDs at 3.00 kHz, i.e., by using/calling DELAY_167us. (Remember that frequency is the reciprocal of period, i.e., f = 1/T.)

The waveform that is output to Port E will blink every other LED with a square wave with a 50% duty cycle, as shown in Figure 2, where X is 0.167ms = 167 µs. (The LED will be on for half the period and off for the other half.) Observe this waveform using the DAD oscilloscope function.

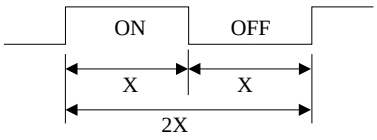

Figure 2: Blinking LED, where X =0.167ms =167µs.

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. Report this frequency and period in your report.

Modify your DELAY_167us subroutine to obtain a frequency as close to 3.00 kHz as possible with a program called **`lab2b.asm`**. Obtain a screenshot from your laptop with the DAD oscilloscope function displaying the 3.00 kHz waveform, the average frequency, and the average period. Report this frequency and period in your report. 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 13, 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. 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 (called **`lab2c.asm`**) to interface with a keypad. To use the keypad, you will need 4 input pins and 4 output pins. You must use Port F for the keypad. Note that the chosen port must **NOT** have any other conflicting (input) devices connected to it.

The keypad included in your kit is similar to a **non**-touchscreen cell phone keypad. See Figure 3 for a description of a keypad. Each key has two contacts, one attached to a "row" wire and the other attached to a "column" wire. When a key is pressed, the column wire and row wires are connected (with a small resistance). There is no power supplied to the keypad except through inputs to the row or column wires. Figure 3 shows how the keypad is interfaced with a microprocessor's port(s). A pull-up or pull-down resistor is used on either each row wire or each column wire. Please note that if you use a keypad different from the one supplied in this semester's kit, the pinouts may change. (The pinouts can be easily determined with a multimeter set on resistance.) You should verify the pin arrangement to be sure.

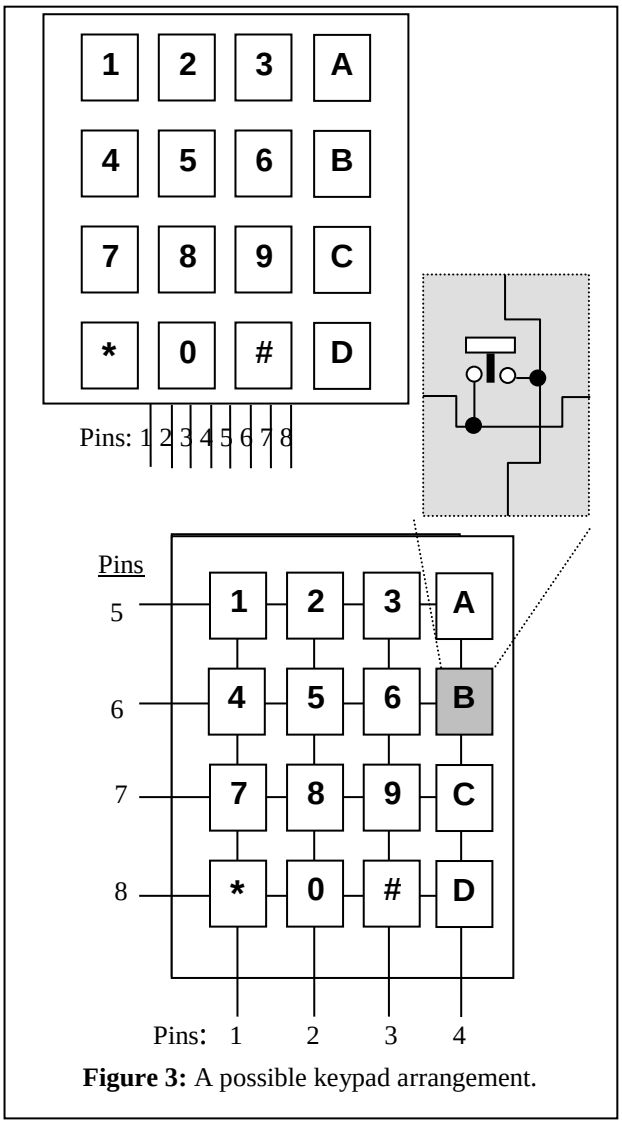

Figure 3: A possible keypad arrangement.

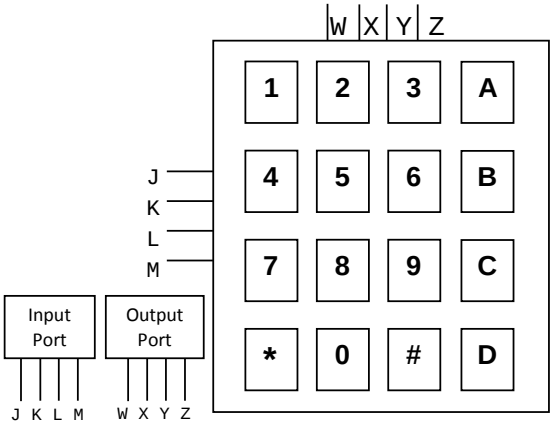

Figure 4: Keypad circuit.

When the keypad is connected to the microprocessor (as shown in Figure 4) and a key is pressed, a connection is established between an input port pin and an output port pin. A value written to the output port can be read from the input port. But what will be read on the unconnected (key not pressed) pins of the input port? Pull-up or pull-down resistors are used so that the input port pins do not have floating (i.e., indeterminate) values. The resistance is necessary to limit the current drawn by the port. With pull-up resistors, the input ports will be at "1" (Vcc = +3.3V) when no key is pressed. When a key is pressed, a "0" (GND) will appear on the input if the corresponding output is at "0".

You will use the XMEGA's built in pull-up (or pull-down) resistors as mentioned in the I/O PORT section of the manual (doc8331). The relevant register on the input port is PORTx_PINnCTRL. You must have pull-up (or pull-down) resistors; if not, when no keys are pressed, the inputs will all be floating (and thus the values that you read will be unpredictable).

Note that in the below description, the words row and column can be reversed (with a corresponding modification to the circuit).

The keypad is read by scanning it one column at a time (Figure 4). First, write a 0b0111 (where 0b is a prefix representing binary numbers) to the appropriate output port pins (columns), and then read the input port (rows). This puts a zero on only the first column of the keypad. If any of the keys in column 1 are pressed, the corresponding input bits will be zero. If no key in column 1 is pressed, all inputs will be pulled high by the pull-up resistor. Next, write a 0b1011 to the output port pins (columns) to scan the second column. Continue by writing a 0b1101 then a 0b1110 to the output port pins to scan the third and fourth columns, respectively.

- **Warning:** Do **not** push two keys at once. Pressing two keys in the same row will connect output pins!

1. To implement the keypad interface, you need 4 input pins and 4 output pins. Normally, you could either use external I/O ports (like those from a future lab) or **a single** 8-bit XMEGA port. In this lab, we will use the Port F (with 4 pins for input and 4 pins for output). (If we used external ports, we would need to provide either pull-up or pull-down resistors, but since the XMEGA has built-in pull-up or pull-down resistors, this will not be necessary.)
2. Wire up your keypad as shown in Figure 4. Add a detailed schematic of keypad connections to the schematics done in previously in this lab and add this to your pre-lab report. You should somehow show the internal XMEGA pull-up or pull-down resistors on your schematic.
3. Write a subroutine to scan the keypad and determine the key pressed. Your subroutine should return one of 0x0 to 0xF, corresponding to the keypad keys. (Use 0xE for '*' and 0xF for '#'.) If no key is found, return 0xFF. NOTE: You will use your keypad extensively in future labs. Make your scan subroutine easily portable between programs.
4. Test your subroutine with a program **`lab2c.asm`**. Emulate this program to prove that it functions appropriately.

Part D

In this part of the assignment, you will use both peripherals (LEDs and keypad) together and write a program that outputs a hex value to the LEDs corresponding to button presses on the keypad.

Write a main routine (called **`lab2d.asm`**) to continuously scan the keypad (by calling the subroutine) and display the pressed key (0x0 to 0xF) on the LED bank. The LEDs should remain lit for as long as the key is pressed. If no key is pressed, display an LED pattern of your choice (such as an error code).

Keypad Placement

If you have not already done so, you will connect the keypad to the uPAD Proto Base board using the supplied ribbon cable. The keypad breakout board will be soldered to the keypad board and the cable will connect to the keypad breakout board to the uPAD Proto Base board.

Cut two 8-pin and four 5-pin male headers from your header strips. You will insert one 8-pin and two 5-pin male headers through the **TOP** of the keypad breakout board (long side sticking up, perpendicular to the board), short side through the board. Solder one of the 5-pin headers as specified; make sure that this header is **as straight as possible**. Solder the second 5-pin header as specified; again make sure that this header is **as straight as possible.** Solder the 8-pin header.

Insert the 8-pin header up through the eight holes below the bottom row of keys on the keypad board, as shown in

Figure 5. Solder these eight pins to secure the keypad break board to the keypad board.

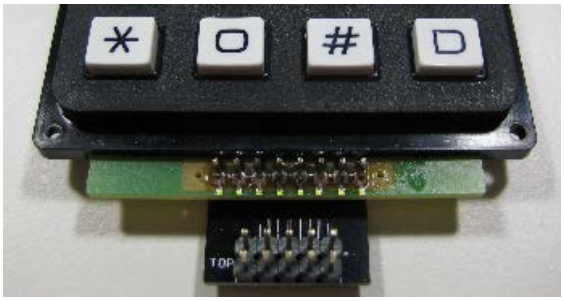

Figure 5: Solder male header into the keypad breakout board, then to keypad itself.

Locate the J23 header socket on the uPAD Proto Base board. You will install two 5-pin headers into these holes. Insert the short pins of one 5-pin header through the board from the top and solder on the bottom; make sure that this header is **as straight as possible**. Insert the short pins of another 5-pin header through the board from the top and solder on the bottom; make sure that this header is **as straight as possible**.

Solder the remaining 8-pin header into J22. Insert the short end through the bottom of the board (so that you can wire-wrap on the bottom of the uPAD Proto Base). You will wire-wrap from the J22 header pins to the appropriate port pins.

[Note: The J21 and J20 header sockets are used for pull-up (or pull-down) resistors. Since we will use internal pull-up or pull-down resistors for the keypad, those will not be needed. If you were using external pull-up resistors, you would solder an 8-pin header into J21 as with J22, solder a SIP resistor into one of the columns of J20, and a 9-pin or 10-pin header into the other column of J20. You would also wire-wrap or connect jumpers across J21 to J22 for the 4 relevant pins requiring pull-up (or pull-down) resisotrs. Finally, you would wire-wrap from J20 header pins to the appropriate port pins.]

Figure 6 shows how you connect the keypad to the Proto Base using the ribbon cable in your kit.

Figure 6: Keypad connected to uPAD Proto Base using ribbon cable.

LAB PROCEDURE

Demonstrate each of the following using your boards and your DAD.

- 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 D program. If your Part D code does not work, demonstrate Part A and Part C.

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 7. Test that you can program the CPLD without problems.

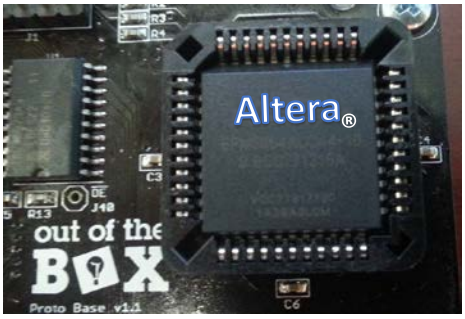

Figure 7: Placement of CPLD.

2. Solder your 74’574 and 74’573 chips to the surface mount area of your uPAD Proto Base, as described below. Figure 8 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 8. 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 9. Solder the two corner pins to the chip in place. Have your TA check your work. When your TA approves your positioning, solder the rest of the pins.

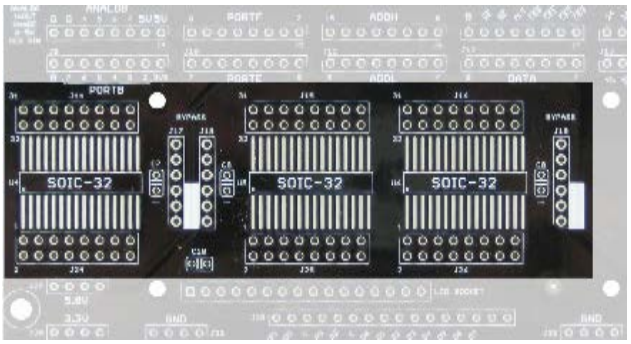

Figure 8: Locations (from left to right) of the 74’574 and 74’573 on uPad Proto Base.

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.

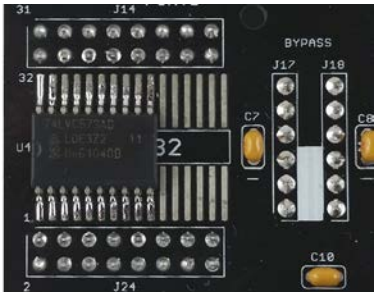

Figure 9: Placement of 74’574 (even though the picture shows a 74’573).

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.