OBJECTIVES

In this lab you will add the *Switch and LED Backpack* onto the uPAD. You will become familiar with using I/O ports to control switch circuits and LED circuits. You will also exercise your programming skills and utilize the DAD/NAD (Analog Discovery board) oscilloscope and logic analyzer functions.

REQUIRED MATERIALS

- Reread *Lab Rules and Policies* document
- DAD/NAD Analog Discovery board
- uPAD 1.4 Development Board
- Switch and LED Backpack + Schematic
- USB A to B Cable
- Atmel AVR XMEGA AU Manual (8331 and 8385)
- AVR Instruction Set Manual

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.

READ the *Switch and LED Backpack* schematic thoroughly to see which orientation the Backpack will go on the uPAD 1.4 Board.

If a program/design does not work, utilize the Atmel Ice via PDI debugging capability, along with your DAD/NAD 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: GPIO AND LEDs

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 C.

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-low** LED circuits to connect to **Port C** (with Port C configured as outputs). Include this circuit design to your pre-lab submission.

To correctly add the *Switch and LED Backpack* (see Figure 1) onto the uPAD 1.4, match the J1, J2, J3, and J4 Headers with PORT A, PORT C, Pins: (V+, +5.0 V, +3.3V, /RESET, PDI_DATA, +2.5 V, GRD, GRD), PORT F respectively.

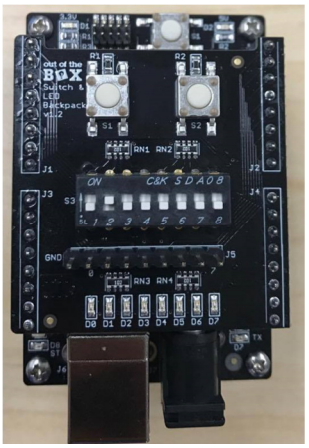

Figure 1: LED & Switch Backpack Orientation

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.

Remember to use J5 Header on the Switches and LEDs Backpack to read signals with the oscilloscope from PORT C (LEDs).

PART B: GPIO AND SWITCHES

In this part of the assignment, you will write a small program called **`Lab1b.asm`**, that reads the 8 DIP switches and stores them at data memory address 0x3744 (in SRAM). Your program will display the value on the 8 **active-low** LEDs, based on pressing two tactile switches S1 and S2 on the backpack. (**Refer to the schemetic sheet for a description of the tactile switches.)**

Write two subroutines, one that displays the stored value from SRAM to the LEDs and the other that retrieves the data from switches and then stores into the specified SRAM address.

Your program will use the two tactile switches. Create a subroutine that returns the values of the two tactile buttons. Your program should continuously read these switches until one is pressed. If S1 is pressed, store the current switch bank data into SRAM. If S2 is pressed, display the value from SRAM onto the active-low LEDs.

Rememeber to draw a flow chart or pseudo-code for your code. For debugging help, place breakpoints on each subroutine and see if your code jumps to the proper routine if the specific button is pressed.

PART C: DELAY

In this section you will write a program called **`Lab1c.asm`** to toggle an output at a specified rate. You

can start writing this program by designing a delay subroutine (DELAY_10ms). Since your board runs at a 2 MHz system clock (until we change this in Lab 2), 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 toggles an output (any pin on Port C) at 50 Hz, i.e., by using/calling DELAY_10ms. (Remember that frequency is the reciprocal of period, i.e., f = 1/T.) The output is a square wave with a 50% duty cycle, as shown in Figure 2, where X is 10 ms. (The LED will be on for half the period and off for the other half.) Observe this waveform using the DAD/NAD.

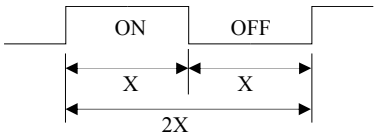

Figure 2: Blinking output, where X =10 ms

The waveform can be quickly displayed with the DAD/NAD 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 a screen shot and include it, the average frequency, and the average period in your in the Appendix of your pre-lab report (as stated in the *Lab Rules and Policies*, part 14, item i).

You will find that the frequency is not even close to 50 Hz. Modify your DELAY_10ms subroutine to obtain a frequency as close to 50 Hz as possible. Take another screen shot and include it, the average frequency, and the average period in your lab report.

It will be useful (and required) to make a subroutine that can delay a select number of milliseconds. Use your corrected DELAY_10ms subroutine to create 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 (remember to push/pop any registers used).

PART D: GPIO AND TIMING

In this part of the assignment, you will write a program (`lab1d.asm`) that will perform a "KITT, Knight Rider" LED pattern at a rate of 50 ms (i.e., the LEDs should move, i.e., shift, **every 50 ms**), while blinking at a rate of 50 Hz. The KITT, Knight Rider pattern is basically turning two LEDs on at the outer pins and moving this pattern into the inner two pins. For this lab, the KITT, Knight Rider pattern is as follows: LED 0 (far right) is on (others are off), LEDs 0 and 1 are on (others are off), LEDs 1 and 2 are on (others are off), …, LEDs 6 and 7 are on (others are off), LED 7 (far left) is on (others are off), LED 0 is on (others are off), …. This pattern should repeat forever at the specified rate. I suggest that you put this pattern in a table (so that it is easy to change the pattern).

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

Note: You will need to demonstrate understanding how to analyze the pattern signal using the Logic Analyzer in Waveforms on Port C (J5 Header on the *Switch and LED Backpack*).

LAB PROCEDURE

- Demonstrate your Part B on the board by loading any value from the switches and displaying it on the LEDs from S1 and S2.
- Demonstrate your Part D by using the DAD/NAD board to see the LEDs shifting and blinking, as specified. Use the LSA to see the pattern moving every 50 ms. (Only demonstrate Part C if you can **NOT** get Part D working.)

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.