BBM 432 – Embedded Systems
Lab 3

Based on Lab 8 of the EDX course UT.6.01x Embedded Systems - Shape the World.

Preparation

You will need a LaunchPad and access to TM4C123_LaunchPadUsersManual.pdf. In this lab, you will need a switch, a 10kΩ resistor, three LEDs, and three 470Ω resistors.

Starter project

You can use the last week's lab file as the starter project.

Purpose

Lab 3 is our first lab requiring you to build circuits on the breadboard and connect them to the LaunchPad. The purpose of this lab is to learn how to interface a switch and an LED.

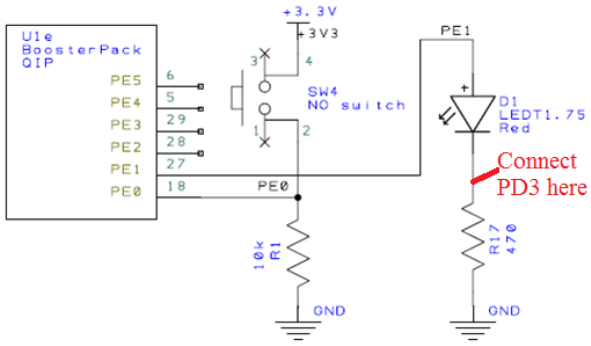
System requirements

In this lab, you will implement an external circuitry for the Lab 2 **but you will use Systick timer to measure the time instead of loops**. On the breadboard, you will attach a switch and three LEDs (green, red and yellow).

1) Decide on the ports that you will connect your switches and LEDs. **You will not use PortF**.
2) Configure the ports that you connect switches as input ports and LED ports as output ports.
3) Each time the switch PF0 is pressed, the color order should reverse, i.e. to green, red, blue and so on. Staying times at for different colors should stay the same (0.5s at blue, 1s at red, 1.5s at green). Make sure that you do not sense the pressing of a switch multiple times.

Note: The color order should reverse every time the switch is pressed.
Note: For example, if the switch is pressed in the middle of the blue period, your system should wait till the end of the blue period. After than it should switch to green and so on.

Part a. Decide on which ports to use as input or output. First, draw a circuit diagram similar to the figure below. Note that there will be three LEDs and a single switch in your experiment. You can choose the switch and LEDs to be positive or negative logic as you prefer. State clearly in your report whether the switches are positive or negative logic. Your external circuit should be compatible with your circuit diagram.



Notice in the figure that the PE0 and PE1 wires cross over each other in the circuit diagram, but they are NOT electrically connected, because there is NO dot at the point of crossing. Conversely, the PE0 signal to the microcontroller, pin2 of the switch and one end of the 10kΩ R1 resistor ARE electrically connected, because there IS a dot at the point of crossing.

Part b. Write the software that satisfies the requirements for this lab. You have already have a code from Lab 2 but you need to make some changes regarding the input/output ports. You will not use Port F this time. Since you use a different port, you need to replace the constant declarations to access port registers using symbolic names. You can include the following file for the register definitions:

http://users.ece.utexas.edu/~valvano/Volume1/tm4c123gh6pm.h

In this lab, you have to change your code such that you use SysTick timer to measure the time. You have to configure the timer as we have discussed in class. You can refer to 10.1 and 10.2 from the following link:
http://users.ece.utexas.edu/%7Evalvano/Volume1/E-Book/C10_FiniteStateMachines.htm

You will still check if the timer has expired inside a loop using the following code:

```
while((NVIC_ST_CTRL_R&0x00010000)==0){ // wait for count flag
  }
```

This is still not a good practice. For a better method, you have to wait till we learn about interrupts.

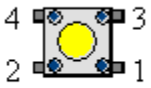
Part c. After the software has been debugged on the simulator, you will build the hardware on the real board. To build circuits, we'll use a solderless breadboard, also referred to as a protoboard. The holes in the protoboard are internally connected in a systematic manner. The long rows of holes along the outer sides of the protoboard are electrically connected. Some protoboards have four long rows (two on each side), while others have just two long rows (one on each side). We refer to the long rows as power buses. If your protoboard has only two long rows (one on each side, we will connect one row to +3.3V and another row to ground. If your protoboard has two long rows on each side, then two rows will be ground, and one row will be +3.3V. Use a black marker and label the voltage on each row.

In the middle of the protoboard, you'll find two groups of holes placed in a 0.1 inch grid. Each adjacent row of five holes is electrically connected. We usually insert components into these holes. IC chips are placed on the protoboard, such that the two rows of pins straddle the center valley. To make connections to the TM4C123 we can run male-male solid wire from

the bottom of the microcontroller board to the protoboard. For example, assume we wish to connect TM4C123 PE1 output to the + side of the LED as shown in the figure. First, cut a 24 gauge solid wire long enough to reach from PE1 and + side of the LED. Next, strip about 0.25 inch off each end. Place one end of the wire in the hole for the PE1 and the other end in one of the four remaining holes in the 5-hole row shared by the + side of the LED.

You should watch the green power LED when you first power up a new circuit. If the green power LED on the LaunchPad does not illuminate, you should quickly disconnect the USB cable.

Notice the switch has 4 pins in a rectangular shape, as shown in Figure 8.3. Each button is a single-pole single-throw normally-open switch. When you press the switch pins 1-4 and 2-3 are connected.



Do not place or remove wires on the protoboard while the power is on.

The next step is to build the LED output circuit. Using the data sheet, hold an LED and identify which pin is the anode and which is the cathode. LEDs emit light when an electric current passes through them, as shown in Figure 8.4. LEDs have polarity, meaning current must pass from anode to cathode to activate. The anode is labelled **a** or + , and cathode is labelled **k** or **-**. The cathode is the short lead and there may be a slight flat spot on the body of round LEDs. Thus, the anode is the longer lead. Furthermore, LEDs will not be damaged if you plug it in backwards. LEDs however won't work plugged in backwards of course.

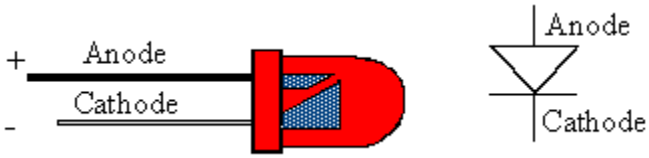


Figure 8.4. A drawing and the circuit symbol for an LED. "Big voltage is on the big wire."

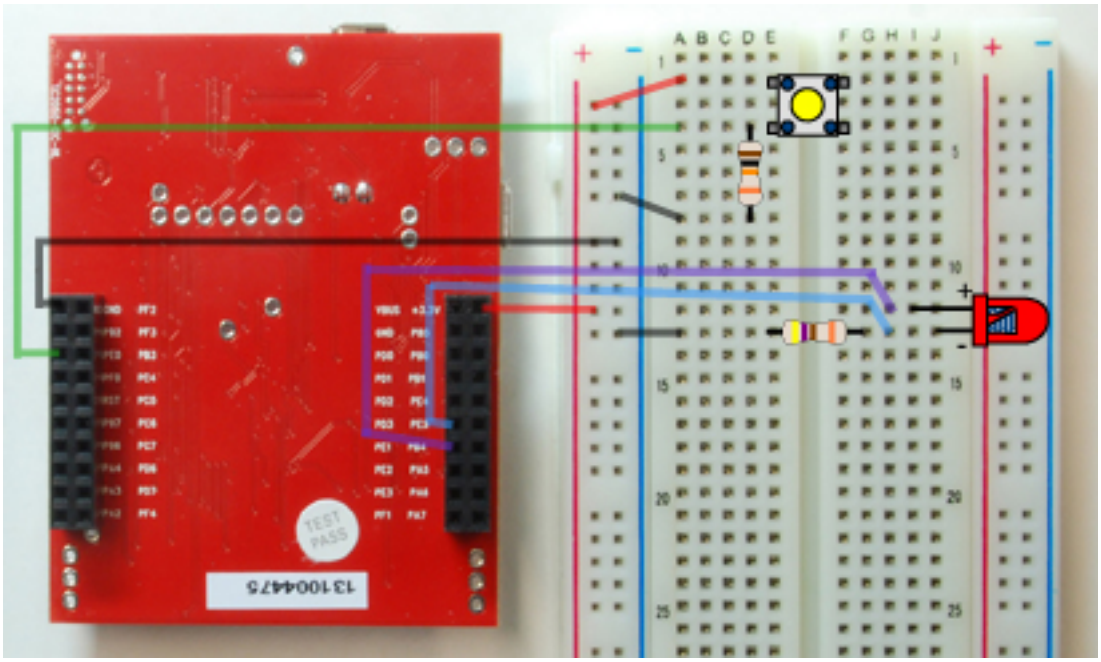


Figure 8.5. A photo showing one possible layout of the circuit. The black wires are ground. The red wires are +3.3V. The purple wire is PE1. The green wire is PE0. The blue wire connects PD3 to the signal between the LED and the resistor; it will be used to measure LED current in Part d). Brown-black-orange resistor is 10kΩ. Yellow-purple-brown resistor is 470Ω. Again, it doesn't matter what color the wires are, the colors are used to identify which wires are which signals. **For the switches you have, probably pins 1-4 and 2-3 are connected when the switch is pressed. So your circuit will be different from this one.**

Part d) Debug your combined hardware/software system on the actual TM4C123 board. First, we will use the debugger to observe the input pin to verify the proper operation of the switch interface. You will have to single step through your code that initializes Port E, and PE0. You then execute the **Peripherals->SystemViewer->GPIO->GPIOE** command. As you single step you should see the actual input as controlled by the switch you have interfaced, see Figure 8.1.

We can verify all aspects of the LED interface are proper by beginning in the **Peripherals->SystemViewer->GPIO->PORTE** debugger window (depending on your choice of ports). In the debugging window you should see bit 1 of Port E is 1. You can calculate the current across the LED using Ohm's Law. The LED current will equal the current in the resistor, which is

$$I_d = V_{PD3}/470\Omega$$

Assuming the port output voltage, V_{OH}, on PE1 is 3.0V, the voltage across the LED is about

$$V_d = (3.0\text{-}V_{PD3})$$

The LED voltage is only an approximation because we are guessing what the voltage on PE1 will be.