This laboratory assignment accompanies the book, [*Embedded Systems: Real-Time Interfacing to ARM Cortex M Microcontrollers, ISBN-13: 978-1463590154*](https://www.amazon.com/Embedded-Systems-Real-Time-Interfacing-Microcontrollers/dp/1463590156), by Jonathan W. Valvano, copyright © 2024.

## Goals

• To review the TM4C123 LaunchPad

• To familiarize yourself with Keil uVision5 for the ARM Cortex M processor

• Interface a switch and LED

• Use the lab power supply apply voltage and measure current

• Use a DVM to measure voltage, resistance, current and calculate power

• Use a scope to measure voltage versus time

## Review

• Lab policies in **Lab00\_intro.docx**,

• LED data sheet

• Switch data sheet

## Specific reading assignments: [ebook Chapter 1](https://users.ece.utexas.edu/~valvano/EE445L/ebook/Chapter1_Introduction.htm) or print book (Sections 2.1, 2.2, 2.4, 3.5) and from [Chapter 3](https://users.ece.utexas.edu/%7Evalvano/Volume1/IntroToEmbSys/Ch3_IntroToInterfacing.html) and [Chapter 5](https://users.ece.utexas.edu/%7Evalvano/Volume1/IntroToEmbSys/Ch5_FiniteStateMachines.html) of ECE319K ebook

<https://users.ece.utexas.edu/%7Evalvano/Volume1/IntroToEmbSys/Ch3_IntroToInterfacing.html>

<https://users.ece.utexas.edu/%7Evalvano/Volume1/IntroToEmbSys/Ch5_FiniteStateMachines.html>

## Starter files these projects are in LabStarterFiles.zip

• **Lab1** project

**Pull the Lab1 project from GitHub**

**Team Size: 1 (each student performs Lab 1 on their own)**

## Required Hardware

EK-TM4C123GXL [http://www.ti.com](http://www.ti.com/tool/ek-tm4c123gxl?keyMatch=tm4c123g&tisearch=Search-EN-Everything)  $23.99

Switch

LED

Resistors

## Background

The objectives of this lab are to review the TM4C123 development environment and put together some basic routines to input from a switch and output to an LED. The switch requires a resistor to convert the 0/∞ impedance of the switch to a logic 0/3.3 voltage. The LED requires a resistor to establish the desired P=V\*I operating point.

## Specifications

You will design, build, and test hardware interfaces for a switch and an LED. You can use either an internal or external resistor for the switch. Do not add capacitors to the switch circuit; we will debounce the switch in software. You can use positive or negative logic for the two interfaces. The overall behavior of the system is to toggle the LED after every second touch of the switch. To determine which pins to use for the switch and LED. Take the number part of your EID and perform a modulo 7 operation to yield an integer from 0 to 6. Table 1.1 determines which pin to use.

|  |  |  |
| --- | --- | --- |
| EID%7 | Switch | LED |
| 0 | PE0 | PB4 |
| 1 | PE2 | PC6 |
| 2 | PE3 | PC5 |
| 3 | PC7 | PE0 |
| 4 | PC6 | PE2 |
| 5 | PC5 | PE3 |
| 6 | PB4 | PC7 |

Table 1.1: Lab 1 GPIO pin assignments.

You will develop and test these four functions

**LED\_Init** Initialize a pin for GPIO output

**LED\_Out** Set the GPIO output pin high or low

**Switch\_Init** Initialize a pin for GPIO input

**Switch\_In** Read and return the GPIO input

## Preparation (do this before your lab period)

1. Get an LED, and a switch from checkout. Install Keil 5 or CCS 12.8, and verify the **Lab1** project compiles, downloads, and runs on your LaunchPad.
2. Use the variable power supply to test the LED. Collect four current versus voltage measurements around 2V. Make sure your LED operating point is 1 to 2 mA for a voltage of about 2V.
3. Design the hardware interfaces for the switch and LED, and draw the circuit in KiCad. Get the resistors needed to build the hardware

## Procedure (do this during your lab period)

**0.** If you bought a used LaunchPad, ask the TA how to run the tester project.

**1.** Implement and debug the four functions listed above.

**2.** Use the DVM to measure the voltage across and the current through the LED when software activates the LED. Calculate LED power.

**3.** Use the analog scope to visualize the switch bounce occurring when the switch is touched and released. Determine the longest time the switch bounces. You might have to flick the switch with your finger to induce the bounce. If you get no bounce at all on your switch, find another student whose switch does bounce and collect voltage versus time response using the analog scope on that switch.

**4.** Write software that toggles the LED after every second touch of the switch. Figure 1.1 shows an FSM graph describing the expected behavior, assuming the switch and LED are interfaced with positive logic. Notice the FSM waits for both the touch and release of the switch. Your solution must include an FSM with the STG/STT implemented as a data structure in ROM using the style/syntax introduced in ECE319K. There are two good options to remove the switch bounce. Option 1: you could add a wait time of 10ms to each state. The main loop does:

a) output depending on new state;

b) wait;

c) input;

d) change state depending on state and input.

Option 2: you could run the FSM engine in a 10ms periodic SysTick ISR. The ISR does:

a) input;

b) change state depending on state and input;

c) output depending on new state.

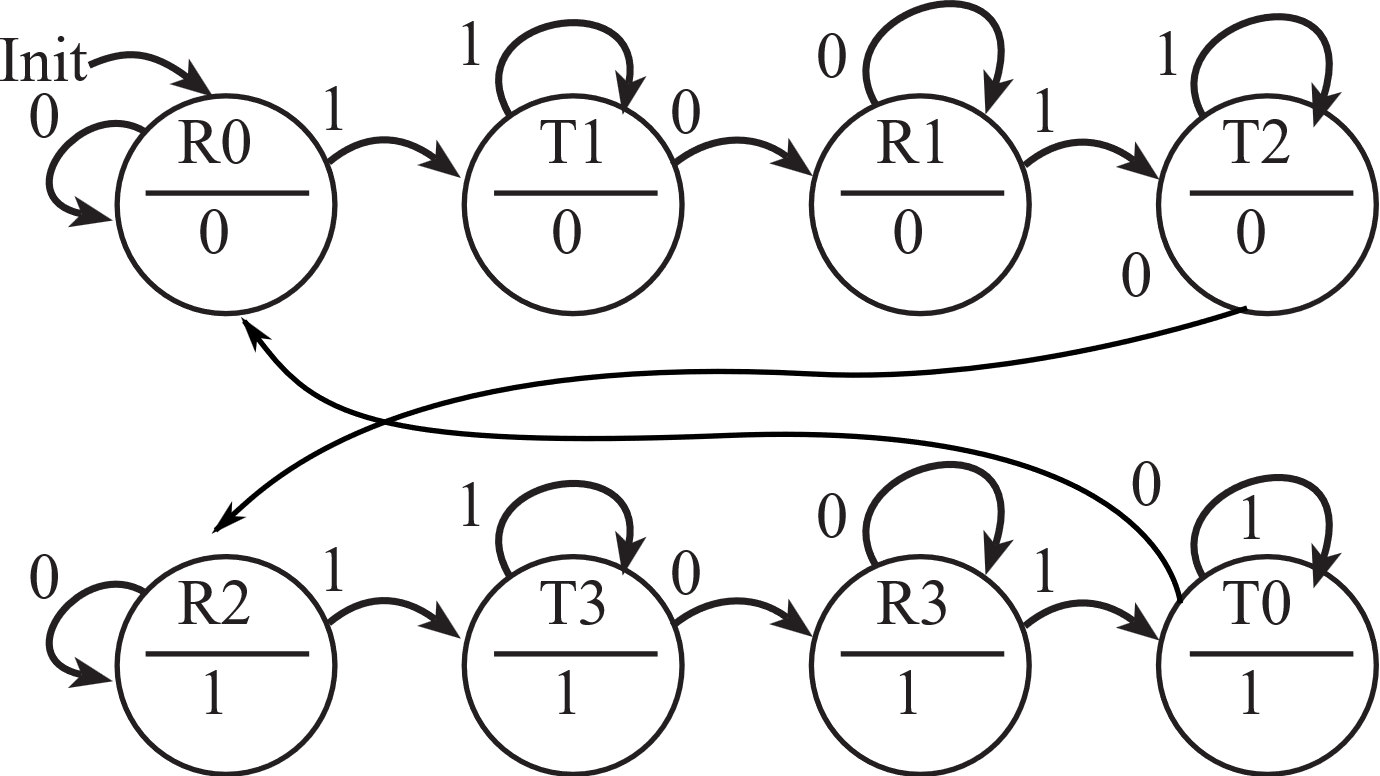


Figure 1.1: Functional behavior of Lab 1. R stands for release. T stands for touch.

## Deliverables (see Lab01Report.docx)

## Checkout (show this to the TA)

Demonstrate the operation of the system and justify your design approach. You should know how to set the scope trigger. You should know how to set breakpoints, use watch window to view local and global variables, and use the debugger to observe GPIO registers.

## Hints

1. Do not create a new project. Start with a project you know compiles and runs. If you need additional projects, make a copy of the entire project folder and, then rename the project.
2. It is good practice to make a small change and test it. Once you have some new code that works, make a back-up, so that when you add something that doesn’t work, you can go back to a previous working version and try a new approach.
3. Please add documentation that makes it easy to change and debug. Your job is to organize these routines to facilitate subsequent laboratories.
4. It is also good practice to look at the assembly language created by the compiler to verify the appropriate function. Analyzing the assembly listing files is an excellent way to double-check if your software will perform the intended function. This is especially true when overflow, dropout, and execution speed are important. We have not found any bugs with this compiler. Most reported compiler bugs (my program doesn’t do what I want) turn out to be programmer errors or misunderstanding about the C language. However, if you think you’ve found a bug, email the source and assembly listing to the TA explaining where the bug is.
5. If you end up with a 10mA LED, also get a driver ULN2003B (or 7406 or PN2222) and follow the process outlined in ebook [Section 1.8.2 LED Interfaces](https://users.ece.utexas.edu/~valvano/EE445L/ebook/Chapter1_Introduction.htm#1_8_2).