California State University, Channel Islands (CSUCI)
Department of Computer Science

COMP-462: Embedded Systems
Lab Report
Fall 2019

Lab Number:	3
Lab Topic:	Breathing LED and Interfacing With Switches

Student Name:	Keith Skinner
Student ID:	002372111
Student Major:	Computer Science

Objective

The purpose of this lab is to learn how to interface a switch and an LED and to program the LED to operate at a variable duty-cycle determined by the switch. You will also perform explicit measurements on the circuits in order to verify they are operational and to improve your understanding of how they work.

Introduction

The primary task of the lab is to make an LED toggle at 2 Hz with varying duty-cycle. The external hardware for the lab includes, Two buttons, one LED, and associated components needed to interface them. The negative logic switch on PF4 exists on the LaunchPad itself. On the TM4C123 the default bus clock is 16 MHz ±1%. However using TExaS_Init engages the Phase-Lock-loop (PLL) and runs the TM4C123 at 80 MHz. At 80 MHz one clock-cycle takes 1/80,000,000 seconds or 12.5 nanoseconds. The SUBS and BNE instructions are executed 800 times. The SUBS takes 1 cycle and the BNE takes (1+P) where P can vary between 1 and 3 cycles. In simulation P is 2 making the wait loop be of 4 cycles. On the real-board P can vary because of optimization using a pipeline. The minimum time to execute this code is 800*(1+(1+1))*12.5 ns= 30 us. The maximum time to execute this code is 800*(1+(1+3))*12.5 ns= 50 us. Since it is impossible to get an accurate time value using the cycle counting method, we will need another way to estimate execution speed. An accurate method to measure time uses a logic analyzer or oscilloscope. In the simulator, we will use a simulated logic analyzer, and on the real board we will use an oscilloscope. To measure execution time, we count rising and falling edges on a digital output pin that occur at known places within the software execution. We can use the logic analyzer or oscilloscope to measure the elapsed time between the rising and falling edges. In this lab we will measure the time between edges PE3.

Procedure

Stage 1: Write a simple main loop that toggles an LED (PE3) at 2 Hz with a 50% duty-cycle calling a delay subroutine.

Stage 2: Rewrite code in Stage1 so you can program a target duty-cycle (say 30%). Verify in simulator using the Logic Analyzer that you indeed have a 2 Hz signal with the target duty-cycle of 30 % (on for 150 ms, and off for 350 ms).

Stage 3: Add software to read the switch on PE2, and use the switch to modify the duty-cycle. Note that the duty-cycle change occurs on a press followed by a release. What the LED does while touching the switch does not matter. So, while the switch is being touched, you can stop the oscillations, or continue the oscillations, your choice. The change takes effect on the release. Test in the simulator and verify function using the simulated Logic Analyzer (this video shows a similar lab solution, but with different pins, different frequencies and different duty cycleshttps://youtu.be/5fD71LdAXZs). In the following Logic Analyzer recording, notice the LED stops toggling when the switch is pressed, then the

toggle resumes when the switch is released. Feel free to implement other similar strategies as long as the duty cycle is changed exactly once each time the switch is pressed and released.

Stage 4: Build the circuit (see figure below) and check twice to make sure you have it correct. Connect the real-board and flash it with the code you wrote in Stage 3. You will see a flashing LED and the effect of changes in the duty-cycle are visible to the naked eye. Note at this point that the Logic Analyzer in Keil is no longer available as a tool as it only works in Simulation.

Stage 5: At this point you implemented 90% of the requirement of this lab.Now you will add the breathing feature which is enabled when PF4 (SW1) is pressed and disabled when released.

Problems

I spent too much time wondering if I needed to do Interrupts for the button instead of copying and pasting the code to check if the button was pressed. I wanted my loop to be “pure”. I spent a lot of time trying to figure out why the PF4 switch wasn’t working when I needed to do extra steps. Not just enable the port, set pin direction, enable the pin.

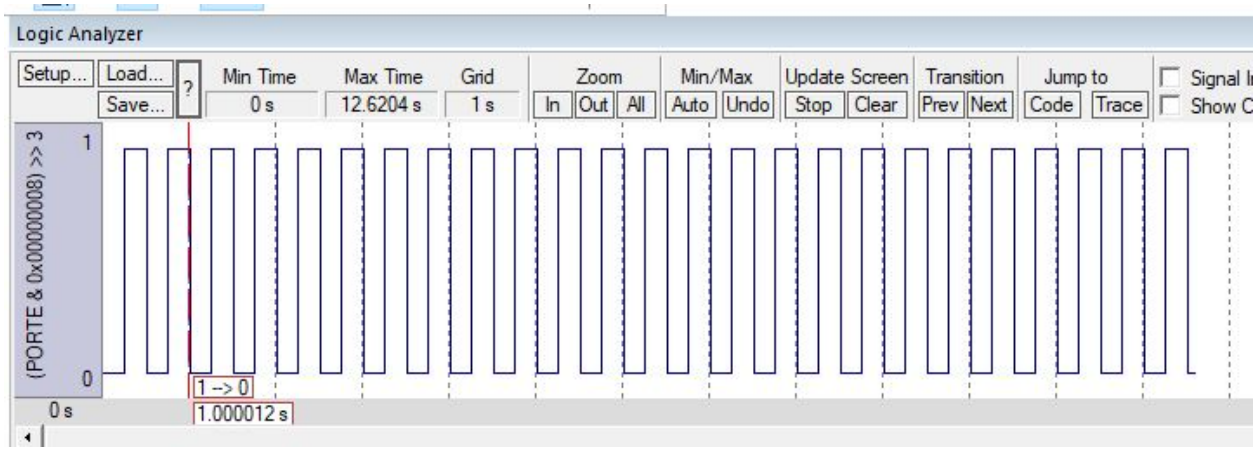
I don’t know how to create a function call “stack” and calling subroutines in subroutines was not working.

In part 4, although I was able to set up the circuit and the effect was essentially the same. The timing was off as expected. By how much, I didn't measure.

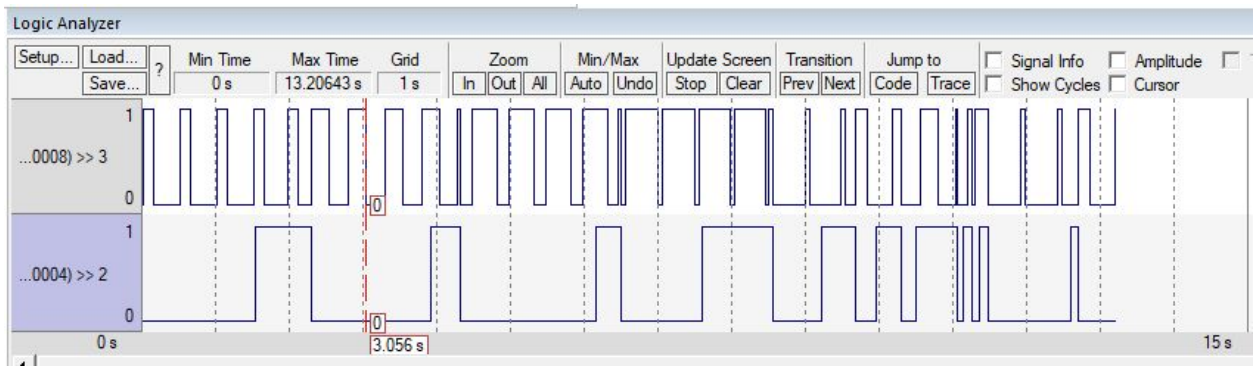
Results

During part 3, I chose to make my loop increment the duty cycle by 20 percent each time PE2 was pressed and released and only acted on release. Also I chose to end the current delay when doing so, causing odd spikes, but I wanted to see results immediately. To do this I had to rework my delay function quite a bit. One problem I ran into was calling subroutines inside subroutines. This made my delay function act erratically. This is something I need to ask the teacher about for future labs. I ended up inlining all my calls to subroutines and it got messy, especially since the structure is still there.

Results of Part 1:



Results of Part 3:



Results of Part 5: