**EE445M – Lab 1: Graphics, LCD, ADC, Timer, and Interpreter**

Sourabh Shirhatti and Nelson Wu, Spring 2015

**Objectives**

The objective of this lab ­­was to introduce us to the Tiva TM4C123 LaunchPad and some of its capabilities. We used an interrupting serial port (employing both hardware and software FIFOs) and a command line interpreter to issue commands to the ADC and LCD. A graphics LCD driver split the display into two segments to be accessed by different threads in future labs. Our ADC driver could be triggered by both software and timer interrupts, providing for flexibility. Periodic timer interrupts were developed for use in thread scheduling.

**Hardware Design**

None

**Software Design**

1) Low level LCD driver (ST7735\_.c and ST7735\_.h files)

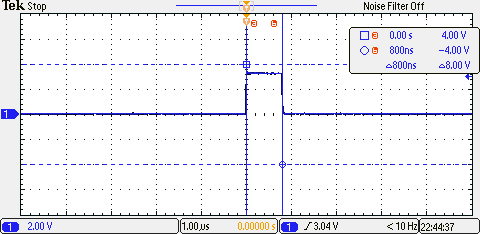
2) Low level ADC driver (ADC.c and ADC.h files)

3) Low level timer driver (OS.c and OS.h files)

4) High level main program (the interpreter)

**Measurement Data**

1. Estimated time: 15 instructions \* 1 CPI \* 20 ns/cycle = 300 ns
2. Measured time: 800 ns



**Analysis and Discussion**

1. ADC Range = 0 - 3.3V

ADC Resolution = 0.00080566406 V

ADC Precision = 12 bits (4096 alternatives)

1. ADC conversion can be started through software (manually calling for a sample), with timer interrupts (at a periodic rate), analog comparators (voltage level), PWM (external pulse signal), and GPIO trigger (pin toggle). We chose to use a software trigger and a timer (periodic) trigger. Software triggers are useful for debugging because we can call for a sample at some point while a program is running to see what the current input is. A periodic trigger allows us to sample some external signal and accurately analyze it for use in the rest of the program.
2. We used method one (setting an output port high at the start of the ISR, and clearing it at the end of the ISR). Profiling methods other than the two mentioned could include reading SysTick (or another timer) at the beginning and end of the ISR, subtracting the two counts, and multiplying by the SysTick period. Our method required the use of a scope or logic analyzer with the appropriate resolution, while the SysTick method would all be in software. Toggling a pin in the ISR is easier than measuring it indirectly because the main program would need to directly access the port (without a read-modify-write sequence); otherwise, the interrupt could occur in the middle, leading to an inaccurate measurement. Toggling in the ISR only requires the addition of two lines of C code, so it is simple and fast. The software method requires more overhead, and it also takes time to read SysTick and output the value.
3. 800 ns/(15 assembly instructions) = 53.3 ns

It takes an average of 53.3 ns to execute one instruction. Running at a system clock frequency of 50 MHz (20 ns clock period), one instruction completes in an average of 2.665 clock cycles.

1. SysTick Range: fbus/(n+1) = interrupt frequency (n is the value in NVIC\_ST\_CURRENT\_R)

SysTick Resolution: 1/fbus = seconds between each tick

SysTick Precision: 24 bits (size of value in NVIC\_ST\_CURRENT\_R)