EECE 544: Embedded System Design
Lab 5
Position Measurement System

1. Pre-Lab

- Download, unzip, open, compile and run **Lab6** into your computer.
- Read section 10.4 in your book.

2. Objective

The objective of this lab is:

- Learn how to sample analog signal using the ADC interface in the TM4C123.
- Develop an ADC device driver.
- Practice calibration technique.
- Use fixed point numbers.
- Develop interrupt-driven sampling device driver.

3. Overview

In this lab you will be developing a position meter using the 12 bit ADC build into the TM4C123 and a slide potentiometer. A linear slide potentiometer can be used to convert position into resistance (0≤R≤12KΩ). By connecting the two ends of the potentiometer to voltage source and ground you will be able to convert resistance into voltage ranging from 0-3.3V. The converted voltage will be corresponding to the position of the slider and will be used as input to the ADC PIN. You will develop software to sample the analog signal and convert it to distance using fixed point format. The calculated measures from your program will be displayed on LCD. The library for the LCD is provided as part of the project.

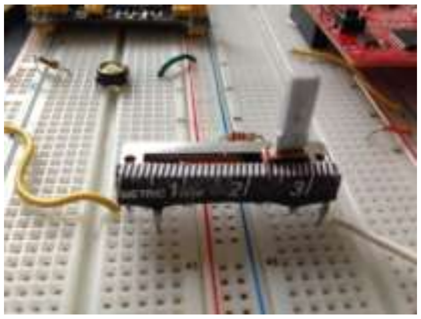

Figure 1

4. Procedure

Part 1: Prepare the position tracker

1. Make a copy of a ruler (cm) and glue it to the side of the potentiometer.
2. Wrap a wire around the slider (armature) to be used as pointer, as shown in figure 1. The full scale range may be anything from 1.0-2.5 cm.
3. Solder three wires to the sides of the potentiometer, name them PIN1, PIN2 and PIN3. Refer to the potentiometer data sheet (652-PTA20432015CPB10) and Connect PIN1 to 0V, PIN2 to 3.3V and PIN3 to V_{in} (voltage input to TM4C123) as shown in figure 2.

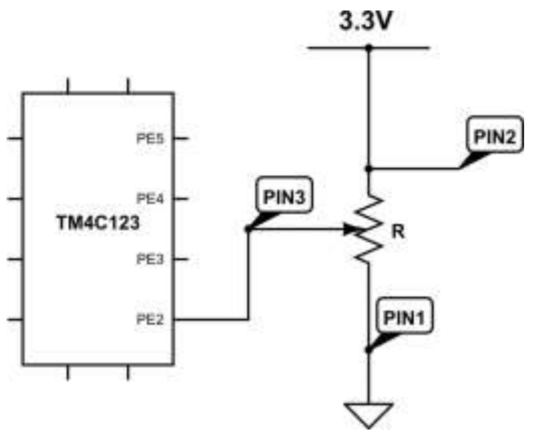

Figure 2

Part 2: ADC and Calibration.

1. Write function ADC_Init to initialize the ADC interface on PIN PE2. Use any channel and any sequencer you wish.
2. Write function ADC_In to sample the ADC.
3. Write a main program (main1) in lab6.c file to test the two functions created in 1 and 2. Your program should initialize the ADC and store the sampled signal in a predefined variable.
4. Connect your system and debug it using TM4C123.
5. Write a main program (main2) to collect calibration data and measure the time it takes the ADC to perform one conversion. Use the program to fill out table 1 below (please note the first two values are example only; fill out your own measurements). Use voltmeter to measure the analog input.

Table 1

Position	**Analog Input (V)**	**ADC Sample**	**Correct Fixed-Point**	**Measured Fixed-point output**
0.5	**0.04**	**0x35**	**500**	
1.0	**1.14**	**0x53C**	**1000**	

6. Use the calibration data to write a function in C that converts 12-bit sampled data (ADC sample) into fixed point integer value representing the measured position. For example the function that describes the data in table 1:

   Corrected Fixed Point Position = (116 * ADC + 88246)/256

7. Initialize any PIN as Output (e.g. PF2). You will be using the PIN output to measure the time it takes the ADC to perform one conversion. Your main program should look something like this:

```
int main(void){
      unsigned long FData;
       PLL_Init();              // Bus clock is 80 MHz
       ADC_Init();              // turn on ADC, set channel to 1
      PortF_Init();             //initializes port F for debugging
 while(1){                      // use logic analyzer to measure execution time for ADC
      PF2 =0x04;
      Data= ADC_In();
      PF2=0;
 }
}
```

8. Take a snapshot of your logic analyzer and record the time for one conversion, as seen in figure 3.

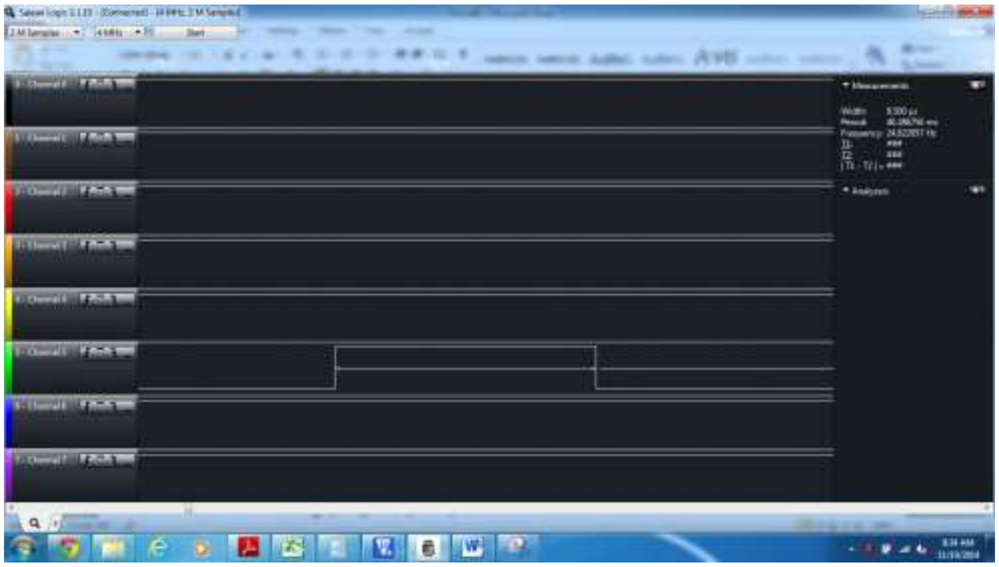

Figure 3: The time for this conversion is 9.5µs

Part 3: SysTick and software interrupt.

1. Write a function SysTick_Init to initialize the SysTick system to interrupt at a frequency of 40 Hz.
2. Write a SysTick interrupt handler that samples the ADC and perform the following tasks:
    - **a.** Toggle an LED to measure ISR execution time (use any PIN).
    - **b.** Sample the ADC.
    - **c.** Save the 12-bit ADC sample data into a mailbox variable call it ADCMail.
    - **d.** Set a mail flag ADCStatus to a new value indicating new data is available.
    - e. Toggle the LED.
    - **f.** Return from interrupt.

3. Write your last main program to test the whole program. The main will initialize PLL, Sys_Tick, ADC and the LCD (refer to Nokia5110.h file for the LCD functions). After initialization the main will perform the following:
    - **a.** Wait for the mailbox flag to be set.
    - **b.** Read the 12-bit ADC sample from the ADCMail.
    - **c.** Clear the mailbox flag ADCStatus to imply the mailbox is empty.
    - **d.** Convert the sample into a fixed point number.
    - **e.** Use the provided LCD functions to display the result into the NOKIA5110 (your TA will help you connect the LCD to your board).
4. Debug the system using the real TM4C123 and the LCD. Use the logic analyzer to observe the sampling rate (the sampling rate can be measured by observing the time between the start and end of the LED toggle).
5. Use your system and collect 7 data points to fill out table 2. The true position column should contain the measured position by the armature and the measured position should be what your software has calculated. Calculate the accuracy of your system using the following equation:

$$\textbf{Average accuracy} = \frac{1}{n}\sum_{i=1}^{n} | x_{ti} - x_{mi} |$$

Table 2

True Position x_{ti}	**Measured Position x_{mi}**	**Error (x_{ti}-x_{mi})**

5. Demonstration

When demonstrating the program, you are expected to explain each line of code if asked. Each student in a team will be asked a different question to demonstrate his or her understanding of the project at hand.

6. Deliverables

1. Table 1 with the calibration data and the results. Note you need to fill out the last column from the display result.
2. A photo showing your position measuring system.
3. A snapshot of the interrupt sampling on the logic analyzer, make sure your sampling rate was 40 Hz.
4. A snapshot of the ADC conversion time.
5. Table 2 with the accuracy data.
6. The source code for lab6.c and ADC.c