 **Electrical and Computer Engineering**

**Computer Design Lab – ENCS4110**

**Report #3**

**Experiment #10:ADC– Measure Analog Voltage Signal**

Student Name: Maisam Alaa

Student ID: 1200650

Partners Name : Sondos Ashraf 1200905

Batool Hammouda 1202874

Instructor: Ayman Hroub

Section: 4

Date: 13\2\2023

# **Abstract:**

The main objective of this experiment is to learn how to use the analog to digital module (ADC) of TM4C123GH6PM Microcontroller using TM4C123G Tiva C Launchpad. Firstly, we will learn to configure ADC modules and sample sequencer of TM4C123 using configuration registers. For demonstration purposes, we will measure analog voltage by using one of the analog input pins of TM4C123GH6PM microcontroller. Most importantly, we will discuss both polling and interrupt-based approach to use TM4C123 MCU ADC.

# **Contents**

[**Abstract:** I](#_Toc127299201)

[**Contents** II](#_Toc127299202)

[**Theory:** 1](#_Toc127299203)

[ADC TYPES 1](#_Toc127299204)

[ADC Modules in the TM4C123GH6PM MCU System 1](#_Toc127299205)

[ADC Programming with the Tiva TM4C123G 2](#_Toc127299206)

[Enabling Clock to ADC 2](#_Toc127299207)

[. Activate ADC SS 3](#_Toc127299208)

[**Procedure:** 4](#_Toc127299209)

[**Lab exercise 1:** 4](#_Toc127299210)

[**Lab exercise 2:** 6](#_Toc127299211)

[**Lab Work 1:** 8](#_Toc127299212)

[**Lab Work 2:** 11](#_Toc127299213)

[**Conclusion:** 17](#_Toc127299214)

[**References:** 18](#_Toc127299215)

**Figures:**

[Figure 1 ADC modules 2](#_Toc127299150)

[Figure 2 RCGCADC 2](#_Toc127299151)

[Figure 3 Lab Exercise1 outputs 6](#_Toc127299152)

[Figure 4 Lab Work1 outputs 11](#_Toc127299153)

[Figure 5 Lab Work 2 output 16](#_Toc127299154)

**Tables:**

[Table 1 ADC modules description 3](#_Toc127299179)

# **Theory:**

## ADC TYPES

Analog-to-Digital Converters (ADCs) transform an analog voltage to a binary number (a series of 1’s and 0’s), and then eventually to a digital number (base 10) for reading on a meter, monitor, or chart. The number of binary digits (bits) that represents the digital number determines the ADC resolution.

A mathematical relationship conveniently shows how the number of bits an ADC handles determines its specific theoretical resolution: An n-bit ADC has a resolution of one part in 2n. For example, a 12-bit ADC has a resolution of one part in 4,096, where 212 = 4,096. Thus, a 12-bit ADC with a maximum input of 10 VDC can resolve the measurement into 10 VDC/4096 = 0.00244 VDC = 2.44 mV. Similarly, for the same 0 to 10 VDC range, a 16-bit ADC resolution is 10/216 = 10/65,536 = 0.153 mV. The resolution is usually specified with respect to the full-range reading of the ADC, not with respect to the measured value at any particular instant[1].

## ADC Modules in the TM4C123GH6PM MCU System

The TM4C123GH6PM microcontroller contains two identical Analog-to-Digital Converter modules. These two modules, ADC0 and ADC1, provide 12-bit conversion precision and share the same 12 analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers[3].

These two ADC modules provide the following features:

* 12 shared analog input channels
* 12-bit precision ADC
* Single-ended and differential-input configurations
* On-chip internal temperature sensor
* Maximum sample rate of one million samples/second (MSPS)
* Optional phase shift in sample time programmable
* Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
* Five flexible trigger controls
  + Software Trigger Controller (default)
  + Timers
  + Analog Comparators
  + PWM
  + GPIO
* Hardware averaging of up to 64 samples
* 2 Analog Comparators

## ADC Programming with the Tiva TM4C123G

To program these two modules, ADC0 and ADC1, we need to understand some of the major registers. Figure 1 shows a simplified block diagram of a Tiva ADC module[3].

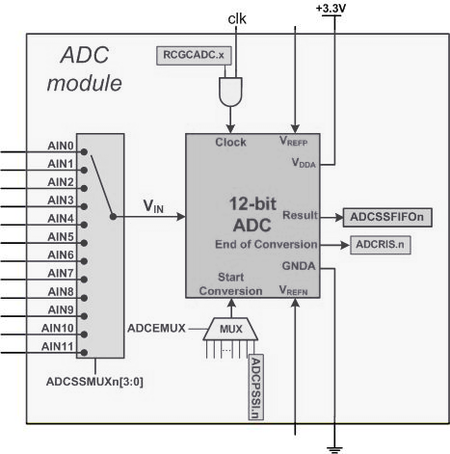
[](http://shukra.cedt.iisc.ernet.in/edwiki/File:Tm4c_adc_module.png)

Figure 1 ADC modules

## Enabling Clock to ADC

First thing we need to do is to enable the clock to the ADC0 or ADC1. Bit 0 and bit 1 of RCGCADC register are used to enable the clock to ADC0 and ADC1 modules, respectively. The RCGCADC is part of the System Control register and is located at base address of 0x400F.E000 with offset 0x638. That means, the RCGCADC is located at physical address of 0x400FE638 (0x400FE000 + 0x638 = 0x400FE638) in memory map. See Figure 2[2].

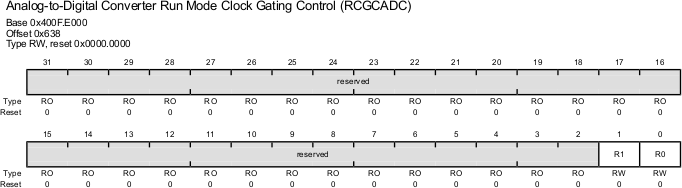
[](http://shukra.cedt.iisc.ernet.in/edwiki/File:Tm4c_rcgcadc_r.png)

Figure 2 RCGCADC

Table 1 ADC modules description

|  |  |  |
| --- | --- | --- |
| Bit | Name | Description |
| 0 | R0 | 0: ADC Module 0 is disabled, 1: Enable and provide a clock to ADC module 0 |
| 1 | R1 | 0: ADC Module 1 is disabled, 1: Enable and provide a clock to ADC module 1 |

## . Activate ADC SS

ADCACTSS (active sample sequencer) register is used to enable or disable sample sequences SS0, SS1, SS2 and SS3. For example, we will be using SS3 in this tutorial. Setting ASEN3 bit of ADCACTSS register enables the SS3 and clearing it disables the SS3. Before configuring ADC channel, we must disable the sample sequencer by clearing bit3 like this: ADC0->ACTSS &= ~(1UL<<3)[2].

# **Procedure:**

## **Lab exercise 1:**

**Description:**

This ADC code of TM4C123GH6PM microcontroller reads analog input from AN0 pin and based on digital value turns on or turns off the onboard green LED of TM4C123G Tiva C launchpad. If measured digital value is less than 2048 LED will remain off. If ADC digital value is greater than or equal to 2048, LED turns on.

**Code:**

/\* TM4C123G Tiva C Series ADC Example \*/

/\* This Program controls the onboard green LED based on discrete digital value of ADC \*/

/\* If AN0 channel value is less 2048 digital value than LED turns off and otherwise remain on \*/

#include "TM4C123GH6PM.h"

#include <stdio.h> //Functions Declaration volatile unsigned int adc\_value; void ADC0SS3\_Handler(void){ adc\_value = ADC0->SSFIFO3; /\* read adc coversion result from SS3 FIFO\*/

ADC0->ISC = 8; /\* clear coversion clear flag bit\*/

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/

} int main(void)

{ volatile float voltage; /\* Enable Clock to ADC0 and GPIO pins\*/

SYSCTL->RCGCGPIO |= (1<<4); /\* Enable Clock to GPIOE or PE3/AN0 \*/

SYSCTL->RCGCADC |= (1<<0); /\* AD0 clock enable\*/

/\* initialize PE3 for AIN0 input \*/

GPIOE->AFSEL |= (1<<3); /\* enable alternate function \*/

GPIOE->DEN &= ~(1<<3); /\* disable digital function \*/

GPIOE->AMSEL |= (1<<3); /\* enable analog function \*/

/\* initialize sample sequencer3 \*/

ADC0->ACTSS &= ~(1<<3); /\* disable SS3 during configuration \*/

ADC0->EMUX &= ~0xF000; /\* software trigger conversion \*/

ADC0->SSMUX3 = 0; /\* get input from channel 0 \*/

ADC0->SSCTL3 |= (1<<1)|(1<<2); /\* take one sample at a time, set flag at 1st sample \*/

ADC0->ACTSS |= (1<<3); /\* enable ADC0 sequencer 3 \*/

/\*Iniitialize PF3 as a digital output pin \*/

SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF

GPIOF->DIR |= 0x08; //set GREEN pin as a digital output pin GPIOF->DEN |= 0x08; // Enable PF3 pin as a digital pin while(1)

{

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/ while((ADC0->RIS & 8) == 0) ; /\* Wait untill sample conversion completed\*/ adc\_value = ADC0->SSFIFO3; /\* read adc coversion result from SS3 FIFO\*/

ADC0->ISC = 8; /\* clear coversion clear flag bit\*/

/\* convert digital value back into voltage \*/ voltage = (adc\_value \* 0.0008); if(adc\_value >= 2048)

GPIOF->DATA = 0x08; /\* turn on green LED\*/ else if(adc\_value < 2048)

GPIOF->DATA = 0x00; /\* turn off green LED\*/

}

}

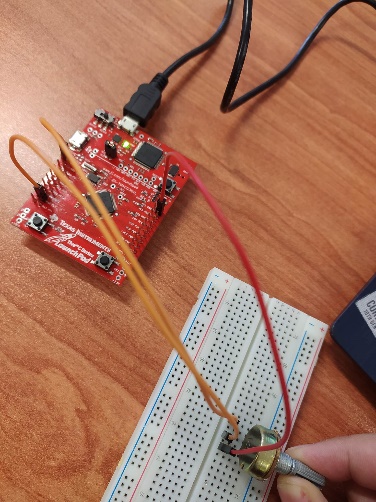
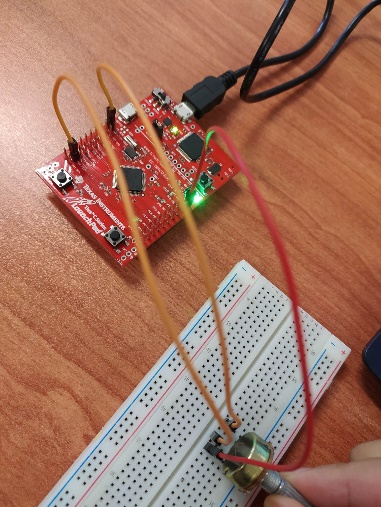


Figure 3 Lab Exercise1 outputs

## **Lab exercise 2:**

**Description:**

The same as the previous program, but in different way of implementing the code it works with interrupt mask

**Code:**

/\* TM4C123G Tiva C Series ADC Example \*/

/\* This Program controls the onboard green LED based on discrete digital value of ADC \*/

/\* If AN0 channel value is less 2048 digital value than LED turns off and otherwise remain on \*/

#include "TM4C123GH6PM.h"

#include <stdio.h> //Functions Declaration void delayUs(int); //Delay in Micro Seconds volatile unsigned int adc\_value; void ADC0SS3\_Handler(void){ adc\_value = ADC0->SSFIFO3; /\* read adc coversion result from SS3 FIFO\*/

ADC0->ISC = 8; /\* clear coversion clear flag bit\*/

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/

} int main(void)

{ char\* str = "Tiva C starting"; //Write any string you want to display on LCD char s[20]; volatile float voltage;

/\* Enable Clock to ADC0 and GPIO pins\*/

SYSCTL->RCGCGPIO |= (1<<4); /\* Enable Clock to GPIOE or PE3/AN0 \*/

SYSCTL->RCGCADC |= (1<<0); /\* AD0 clock enable\*/

/\* initialize PE3 for AIN0 input \*/

GPIOE->AFSEL |= (1<<3); /\* enable alternate function \*/

GPIOE->DEN &= ~(1<<3); /\* disable digital function \*/

GPIOE->AMSEL |= (1<<3); /\* enable analog function \*/

/\* initialize sample sequencer3 \*/

ADC0->ACTSS &= ~(1<<3); /\* disable SS3 during configuration \*/

ADC0->EMUX &= ~0xF000; /\* software trigger conversion \*/

ADC0->SSMUX3 = 0; /\* get input from channel 0 \*/

ADC0->SSCTL3 |= (1<<1)|(1<<2); /\* take one sample at a time, set flag at 1st sample \*/

ADC0->ACTSS |= (1<<3); /\* enable ADC0 sequencer 3 \*/

/\*Iniitialize PF3 as a digital output pin \*/

SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF

GPIOF->DIR |= 0x08; //set GREEN pin as a digital output pin

GPIOF->DEN |= 0x08; // Enable PF3 pin as a digital pin

/\* Enable ADC Interrupt \*/

ADC0->IM |= (1<<3); /\* Unmask ADC0 sequence 3 interrupt\*/

NVIC->ISER[0] |= 0x00020000; /\* enable IRQ17 for ADC0SS3\*/

ADC0->ACTSS |= (1<<3); /\* enable ADC0 sequencer 3 \*/

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/ while(1)

{

/\*control Green PF3->LED \*/

/\* convert digital value back into voltage \*/ voltage = (adc\_value \* 0.0008); // sprintf(s, "\r\nVoltage = %f", voltage); if(adc\_value >= 2048)

GPIOF->DATA = 0x08; /\* turn on green LED\*/ else if(adc\_value < 2048)

GPIOF->DATA = 0x00; /\* turn off green LED\*/

}

}

## **Lab Work 1:**

**Description:**

We modified the code in part (1) above, so that a RED led is on only when the value is greater than 1000. The GREEN led is on only when the value is below 1000.

**Code:**

/\* TM4C123G Tiva C Series ADC Example \*/

/\* This Program controls the onboard green LED based on discrete digital value of ADC \*/

/\* If AN0 channel value is less 2048 digital value than LED turns off and otherwise remain on \*/

#include "TM4C123GH6PM.h"

#include <stdio.h>

//Functions Declaration

void delayUs(int); //Delay in Micro Seconds

volatile unsigned int adc\_value;

void ADC0SS3\_Handler(void){

adc\_value = ADC0->SSFIFO3; /\* read adc coversion result from SS3 FIFO\*/

ADC0->ISC = 8; /\* clear coversion clear flag bit\*/

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/

}

int main(void)

{

char\* str = "Tiva C starting"; //Write any string you want to display on LCD

char s[20];

volatile float voltage;

/\* Enable Clock to ADC0 and GPIO pins\*/

SYSCTL->RCGCGPIO |= (1<<4); /\* Enable Clock to GPIOE or PE3/AN0 \*/

SYSCTL->RCGCADC |= (1<<0); /\* AD0 clock enable\*/

/\* initialize PE3 for AIN0 input \*/

GPIOE->AFSEL |= (1<<3); /\* enable alternate function \*/

GPIOE->DEN &= ~(1<<3); /\* disable digital function \*/

GPIOE->AMSEL |= (1<<3); /\* enable analog function \*/

/\* initialize sample sequencer3 \*/

ADC0->ACTSS &= ~(1<<3); /\* disable SS3 during configuration \*/

ADC0->EMUX &= ~0xF000; /\* software trigger conversion \*/

ADC0->SSMUX3 = 0; /\* get input from channel 0 \*/

ADC0->SSCTL3 |= (1<<1)|(1<<2); /\* take one sample at a time, set flag at 1st sample \*/

ADC0->ACTSS |= (1<<3); /\* enable ADC0 sequencer 3 \*/ \

/\*Iniitialize PF3 as a digital output pin \*/

SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF

GPIOF->DIR |= 0x08; //set GREEN pin as a digital output pin

GPIOF->DEN |= 0x08; // Enable PF3 pin as a digital pin

GPIOF->DIR |= 0x02; //set GREEN pin as a digital output pin

GPIOF->DEN |= 0x02;

/\* Enable ADC Interrupt \*/

ADC0->IM |= (1<<3); /\* Unmask ADC0 sequence 3 interrupt\*/

NVIC->ISER[0] |= 0x00020000; /\* enable IRQ17 for ADC0SS3\*/

ADC0->ACTSS |= (1<<3); /\* enable ADC0 sequencer 3 \*/

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/

while(1)

{

/\*control Green PF3->LED \*/

/\* convert digital value back into voltage \*/

voltage = (adc\_value \* 0.0008);

// sprintf(s, "\r\nVoltage = %f", voltage);

if(adc\_value >= 1000)

GPIOF->DATA = 0x02; /\* turn on red LED\*/

else if(adc\_value < 1000)

GPIOF->DATA = 0x08; /\* turn off green LED\*/

}

}

**Notes:**

It has been modified so that when the measured value is greater than 1000 LCD turns red on, else green appears.

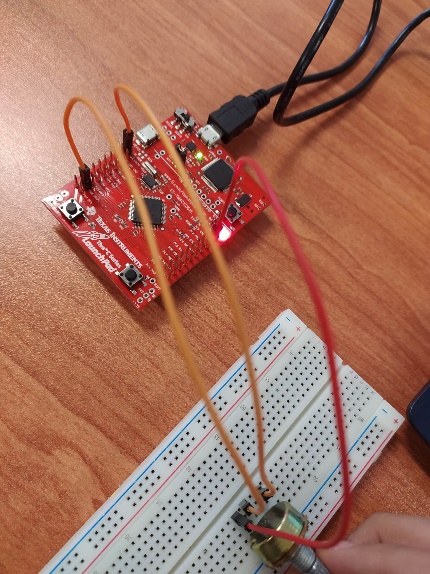
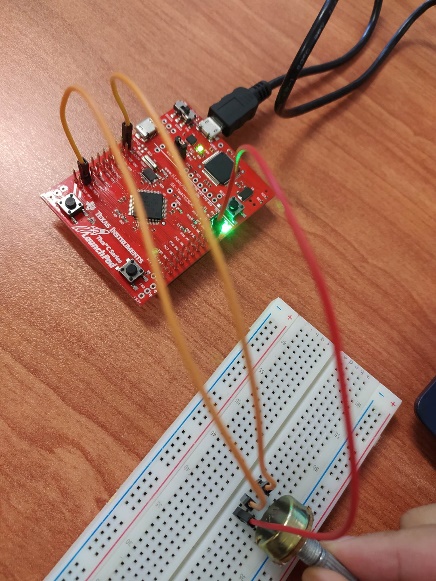


Figure 4 Lab Work1 outputs

## **Lab Work 2:**

**Description:**

We modified the code above (ADC with interrupt), so the input value and the voltage value are both displayed on the LCD as follow:

The input value: 767

The voltage: 2.474 volt

**Code:**

#include "TM4C123.h" // Device header

#define LCD GPIOB //LCD port with Tiva C

#define RS 0x01 //RS -> PB0 (0x01)

#define RW 0x02 //RW -> PB1 (0x02)

#define EN 0x04 //EN -> PB2 (0x04)

#include "TM4C123GH6PM.h"

#include <stdio.h> //Functions Declaration void delayUs(int); //Delay in Micro Seconds void delayMs(int); //Delay in Milli Seconds

void LCD4bits\_Init(void); //Initialization of LCD Dispaly

void LCD\_Write4bits(unsigned char, unsigned char); //Write data as (4 bits) on LCD void LCD\_WriteString(char\*); //Write a string on LCD void LCD4bits\_Cmd(unsigned char); //Write command void LCD4bits\_Data(unsigned char); //Write a character

//Functions Declaration

void delayUs(int); //Delay in Micro Seconds volatile unsigned int adc\_value;

void ADC0SS3\_Handler(void){

adc\_value = ADC0->SSFIFO3; /\* read adc coversion result from SS3 FIFO\*/

ADC0->ISC = 8; /\* clear coversion clear flag bit\*/

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/ }

int main(void)

{ /\* char\* str1 = "ENCS4110…"; //Write any string you want to display on the first row of LCD char\* str2 = "Lab …"; //Write any string you want to display on the second row of LCD

LCD4bits\_Init(); //Initialization of LCD

LCD4bits\_Cmd(0x01); //Clear the display

LCD4bits\_Cmd(0x80); //Force the cursor to beginning of 1st line delayMs(500); //delay 500ms for LCD (MCU is faster than LCD) LCD\_WriteString(str1); //Write the string on LCD delayMs(500); //Delay 500 ms to let the LCD diplays the data LCD4bits\_Cmd(0xC0); //Force the cursor to beginning of 2

delayMs(500); //delay 500ms for LCD (MCU is faster than LCD) LCD\_WriteString(str2); //Write the string on LCD delayMs(500); //Delay 500 ms to let the LCD diplays the data

\*/

char s[20];

char s1[20]; volatile float voltage;

char\* str = "Tiva C starting"; //Write any string you want to display on LCD

LCD4bits\_Init(); //Initialization of LCD

LCD4bits\_Cmd(0x01); //Clear the display

LCD4bits\_Cmd(0x80); //Force the cursor to beginning of 1st line delayMs(500); //delay 500ms for LCD (MCU is faster than LCD) LCD\_WriteString(str); //Write the string on LCD delayMs(500); //Delay 500 ms to let the LCD diplays the data

/\* Enable Clock to ADC0 and GPIO pins\*/

SYSCTL->RCGCGPIO |= (1<<4); /\* Enable Clock to GPIOE or PE3/AN0 \*/

SYSCTL->RCGCADC |= (1<<0); /\* AD0 clock enable\*/

/\* initialize PE3 for AIN0 input \*/

GPIOE->AFSEL |= (1<<3); /\* enable alternate function \*/

GPIOE->DEN &= ~(1<<3); /\* disable digital function \*/

GPIOE->AMSEL |= (1<<3); /\* enable analog function \*/

/\* initialize sample sequencer3 \*/

ADC0->ACTSS &= ~(1<<3); /\* disable SS3 during configuration \*/

ADC0->EMUX &= ~0xF000; /\* software trigger conversion \*/

ADC0->SSMUX3 = 0; /\* get input from channel 0 \*/

ADC0->SSCTL3 |= (1<<1)|(1<<2); /\* take one sample at a time, set flag at 1st sample \*/

ADC0->ACTSS |= (1<<3); /\* enable ADC0 sequencer 3 \*/

/\*Iniitialize PF3 as a digital output pin \*/

SYSCTL->RCGCGPIO |= 0x20; // turn on bus clock for GPIOF

GPIOF->DIR |= 0x08; //set GREEN pin as a digital output pin

GPIOF->DEN |= 0x08; // Enable PF3 pin as a digital pin

/\* Enable ADC Interrupt \*/

ADC0->IM |= (1<<3); /\* Unmask ADC0 sequence 3 interrupt\*/

NVIC->ISER[0] |= 0x00020000; /\* enable IRQ17 for ADC0SS3\*/

ADC0->ACTSS |= (1<<3); /\* enable ADC0 sequencer 3 \*/

ADC0->PSSI |= (1<<3); /\* Enable SS3 conversion or start sampling data from AN0 \*/

while(1)

{

/\*control Green PF3->LED \*/

/\* convert digital value back into voltage \*/ voltage = (adc\_value \* 0.0008);

LCD4bits\_Init(); //Initialization of LCD

LCD4bits\_Cmd(0x01); //Clear the display

LCD4bits\_Cmd(0x80); //Force the cursor to beginning of 1st line delayMs(500); //delay 500ms for LCD (MCU is faster than LCD)

sprintf(s, "Voltage = %0.2f", voltage);

sprintf(s1, "inval = %d", adc\_value);

LCD\_WriteString(s1); //Write the string on LCD delayMs(500); //Delay 500 ms to let the LCD diplays the data LCD4bits\_Cmd(0xC0); //Force the cursor to beginning of 2

delayMs(500); //delay 500ms for LCD (MCU is faster than LCD) LCD\_WriteString(s); //Write the string on LCD delayMs(500); //Delay 500 ms to let the LCD diplays the data

if(adc\_value >= 2048)

GPIOF->DATA = 0x08; /\* turn on green LED\*/ else if(adc\_value < 2048)

GPIOF->DATA = 0x00; /\* turn off green LED\*/

}

}

void LCD4bits\_Init(void)

{

SYSCTL->RCGCGPIO |= 0x02; //enable clock for PORTB delayMs(10); //delay 10 ms for enable the clock of PORTB LCD->DIR = 0xFF; //let PORTB as output pins

LCD->DEN = 0xFF; //enable PORTB digital IO pins

LCD4bits\_Cmd(0x28); //2 lines and 5x7 character (4-bit data, D4 to D7)

LCD4bits\_Cmd(0x06); //Automatic Increment cursor (shift cursor to right)

LCD4bits\_Cmd(0x01); //Clear display screen

LCD4bits\_Cmd(0x0F); //Display on, cursor blinking

}

void LCD\_Write4bits(unsigned char data, unsigned char control)

{ data &= 0xF0; //clear lower nibble for control control &= 0x0F; //clear upper nibble for data

LCD->DATA = data | control; //Include RS value (command or data ) with data

LCD->DATA = data | control | EN; //pulse EN delayUs(0); //delay for pulsing EN

LCD->DATA = data | control; //Turn off the pulse EN

LCD->DATA = 0; //Clear the Data

}

void LCD\_WriteString(char \* str)

{ volatile int i = 0; //volatile is important while(\*(str+i) != '\0') //until the end of the string

{

LCD4bits\_Data(\*(str+i)); //Write each character of string i++; //increment for next character

}

}

void LCD4bits\_Cmd(unsigned char command)

{

LCD\_Write4bits(command & 0xF0 , 0); //upper nibble first LCD\_Write4bits(command << 4 , 0); //then lower nibble if(command < 4)

delayMs(2); //commands 1 and 2 need up to 1.64ms else

delayUs(40); //all others 40 us

}

void LCD4bits\_Data(unsigned char data)

{

LCD\_Write4bits(data & 0xF0 , RS); //upper nibble first

LCD\_Write4bits(data << 4 , RS); //then lower nibble

delayUs(40); //delay for LCD (MCU is faster than LCD)

}

void delayMs(int n)

{

volatile int i,j; //volatile is important for variables incremented in code for(i=0;i<n;i++)

for(j=0;j<3180;j++) //delay for 1 msec

{} }

void delayUs(int n)

{

volatile int i,j; //volatile is important for variables incremented in code for(i=0;i<n;i++)

for(j=0;j<3;j++) //delay for 1 micro second

{} }

**Notes:**

After modifying ADC with interrupt code, and adding string to appear on the LCD screen, we added a variable resistance. The code will read the resistance value and print the input Value and the voltage on the LCD screen.

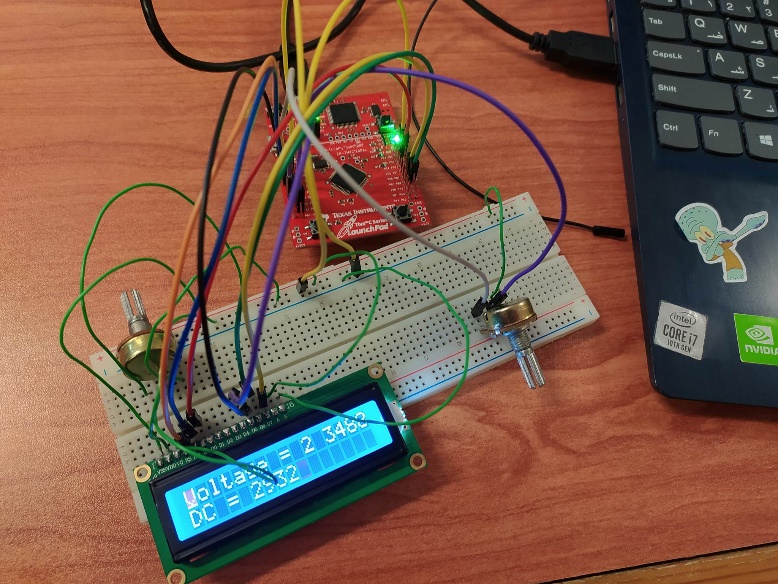


Figure 5 Lab Work 2 output

# **Conclusion:**

After getting done with these experiments and executing each program, we have fully understood how to interact with the TM4C123GH6PM Microcontroller using TM4C123G Tiva C Launchpad. We learned using TM4C123 MCU ADC both polling and interrupt-based approach, and also get to know more about ADC different modules.

# **References:**

[1] <https://www.mccdaq.com/PDFs/specs/Analog-to-Digital.pdf>

[2] <https://microcontrollerslab.com/adc-tm4c123g-tiva-c-launchpad-measure-analog-voltage-signal/>

[3] <http://shukra.cedt.iisc.ernet.in/edwiki/EmSys:ADC_Programming_with_the_Tiva_TM4C123G>