Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Computer Design Lab – ENCS4110

Experiment No. 10 Report

ADC– Measure Analog Voltage Signal with TM4C123

Prepared by:

Dana Ghnimat

Partners:

Jana Sawalmeh

Shahd Shreteh

Instructor: Dr. Abualseoud Hanani
Assistant: Eng. Hanan Awawdeh

Section: 2

Date: 03/09/2023

Abstract

In this experiment, our focus is on learning how to utilize the analog to digital module

(ADC) of the TM4C123GH6PM Microcontroller using the TM4C123G Tiva C Launchpad. Our initial step includes designing the ADC modules and test sequencer through the utilize of setup registers. To demonstrate its functionality, we are going measure analog voltage using one of the analog input pins on the TM4C123GH6PM microcontroller. The experiment also dives into two particular approaches for utilizing the TM4C123 MCU ADC:

the polling approach and the interrupt-based approach. The polling approach includes continuously checking the ADC status to decide when a change is complete, in this way recovering the converted information. In contrast, the interrupt-based approach involves configuring interrupts to trigger when a conversion finishes, streamlining the ADC data recovery handler. By gaining proficiency in these methods, we are able successfully configure and utilize the ADC module for analog voltage measurements and choose the most suitable approach based on our specific requirements.

Table of Contents

Table of figures:

Table of tables:

Theory

LCD Interfacing with TM4C123 Tiva LaunchPad

16×2 LCD Introduction

This LCD can display 32 ASCII characters. It consists of two rows and one column. Each row can display one ASCII character. Hence, the position of each character is defined in terms of rows and columns order pairs (x,y). For example, (0,0) means the first row and first column, and (1,15) means the second row and 15th column.

Pinout Diagram

Following diagram shows the pinout diagram of 16×2 LCD.

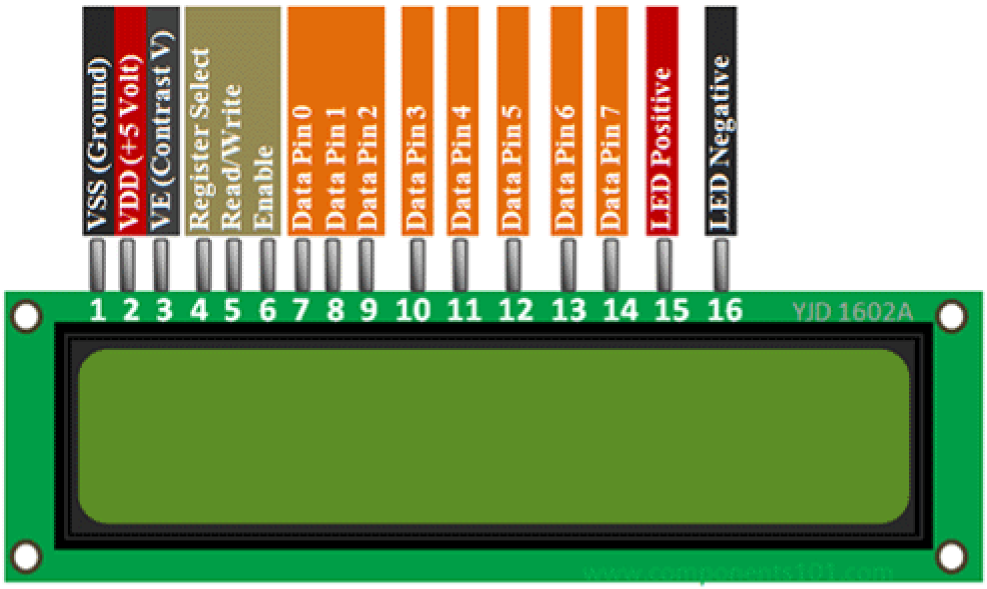


Figure 1 LCD pins

16×2 LCD Pin Details

Let’s first discuss the pin details of LCD. It consists of 16 pins such as data, control, power supply, and backlight LED pins.

Data Pins

Pins D0 to D7 are data pins which are used to send data to be displayed or commands to LCD controller. Hence, these lines will be connected with TM4C123 GPIO pins to transfer data. But this LCD can be used either in 4-bit (only D0 to D3 pins are used) or 8-bit mode (all D0 to D7 pins are used).

Control Pins

LCD Contrast (Vo) : It is used to adjust the contrast of LCD with respect to text display. This contrast can be set through by using a potential divider circuit with a variable resistor.

Register Select (RS) : With the help of this pin, TM4C123 microcontroller informs the LCD controller either we are sending commands or data to LCD. In other words, it helps to differentiate between data and commands.

For example, when we want to send commands to LCD from TM4C123 microcontroller, we set this pin active low by sending an active signal from the GPIO pin of Tiva Launchpad. These commands are setting cursor position, cursor on or off, scrolling text left or right, clearing text from LCD, etc. On the contrary, when we want to send data to LCD, we provide active high signal to this pin from TM4C123 microcontroller

Read/Write (R/W)
As its name suggests, R/W is used to select read or write mode of LCD. When this pin is set to active high, LCD will be in read mode which means we can read data from the LCD. Similarly, when this pin is set to active low, we will be able to write data to the LCD. We will connect this pin to the ground. Because we are using a 16×2 LCD as an output device only.

Enable (E)
This pin is used to enable and disable LCD. When this pin is active low, LCD controller will be disabled. That means control pins and data pins will not have any effect on the display. On the other hand, when the enable pin is set to active high, the LCD will work normally and process all data and control instructions.

Backlight LED Pins
These pins are cathode and anode pins of back light LED. They are used to provide +5 volts and ground to anode and cathode pins.

Power Supply Pins
+5 volt power supply pins.

Difference Between 4-bit and 8-bit mode
HD47780 LCDs can be interfaced with TM4C123 Tiva Launchpad either in 4-bit or 8-bit mode. For a 4-bit interface, we need to use 6 GPIO pins of Launchpad. In 4-bit mode, data transfers from microcontroller to the LCD in two consecutive half bytes. On the contrary for 8-bit mode, one byte data transfers to the LCD in one go. Consequently, we need to use 10 GPIO pins of TM4C123 microcontroller. Therefore, to save GPIO pins, we will use 4-bit mode of 16×2 LCD. Because, displaying data on LCD is not a time-critical action. Hence. 4-bit use will save us 4 GPIO pins of the Tiva launchpad. [1]

ADC TM4C123G Tiva C Launchpad – Measure Analog Voltage Signal

TM4C123 Microcontroller ADC Introduction

It has two identical successive Approximation Register (SAR) architecture-based ADC modules such as ADC0 and ADC1 which share 12 analog input channels as shown in the figure below:

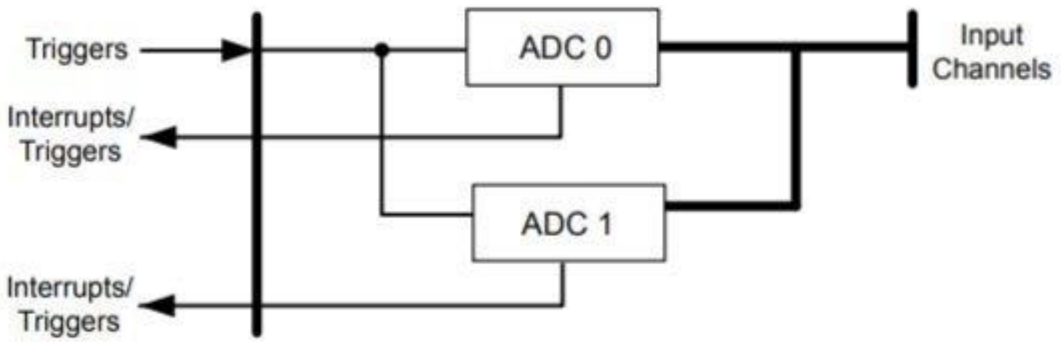


Figure 2 Microcontroller ADC

Each ADC input channel supports 12-bit conversion resolution. Furthermore, it has a built-in temperature sensor and a programmable sequencer to sample multiple input analog sources. The following table shows the GPIO pins which are used for alternate functions with analog input channels.

Table 1 ADC input channel

Analog Channel	Pin Name	Pin Number
AN0	PE3	6
AN1	PE2	7
AN2	PE1	8
AN3	PE0	9
AN4	PD3	64
AN5	PD2	63
AN6	PD1	62
AN7	PD0	61
AN8	PE5	60
AN9	PE4	59
AN10	PB4	58
AN11	PB5	57

ADC Resolution

12-bit resolution means the ADC converts the analog values between 0 to 3.3 volts into discrete digital values in the range of 0 to (2^12) – 1 or 0 to 4095. That means, if digital value is 0, the analog input to ADC channel is zero and if digital value is 4095, the analog input to ADC channel is 3.3 volts. This formula is used to calculate the minimum analog voltage ADC can measure:

Resolution = 3.3 volts / 4095 = 0.8mV

Here 0.8mV means the discrete digital value after conversion shows the 0.8 millivolts. For example, if the digital value measured by ADC is 2048, we can calculate analog voltage by multiplying digital value with 0.8 millivolts.

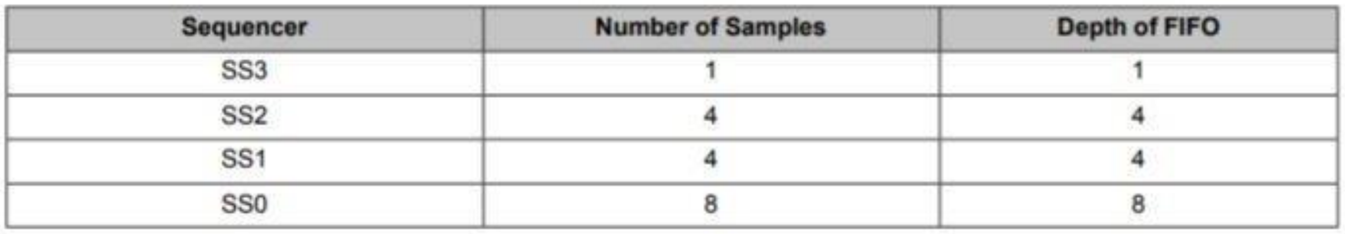
Analog voltage = 2048 x 0.8 = 1.65V

In short, the maximum voltage ADC can measure directly is 3.3 volts and the minimum is zero volts.

Note: We can measure higher voltages such as high DC voltage, AC voltage also. But we have to use a signal conditioning circuit.

Sample Sequencers

As we mentioned earlier, TM4C123 microcontroller has two ADC modules that are ADC0 and ADC1. Each analog to digital converter has a separate sample sequencer (SS0, SS1, SS2 and SS3) which can sample different input channels. All these sequencers offer different numbers of samples that they can capture and the length of FIFO. The following table shows the number of samples and the length of FIFO for each sequencer circuit.



Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

Table 2 Sample Sequencers

TM4C123GH6PM microcontroller supports sampling rate from 125 KSPS to 1 MSPS.

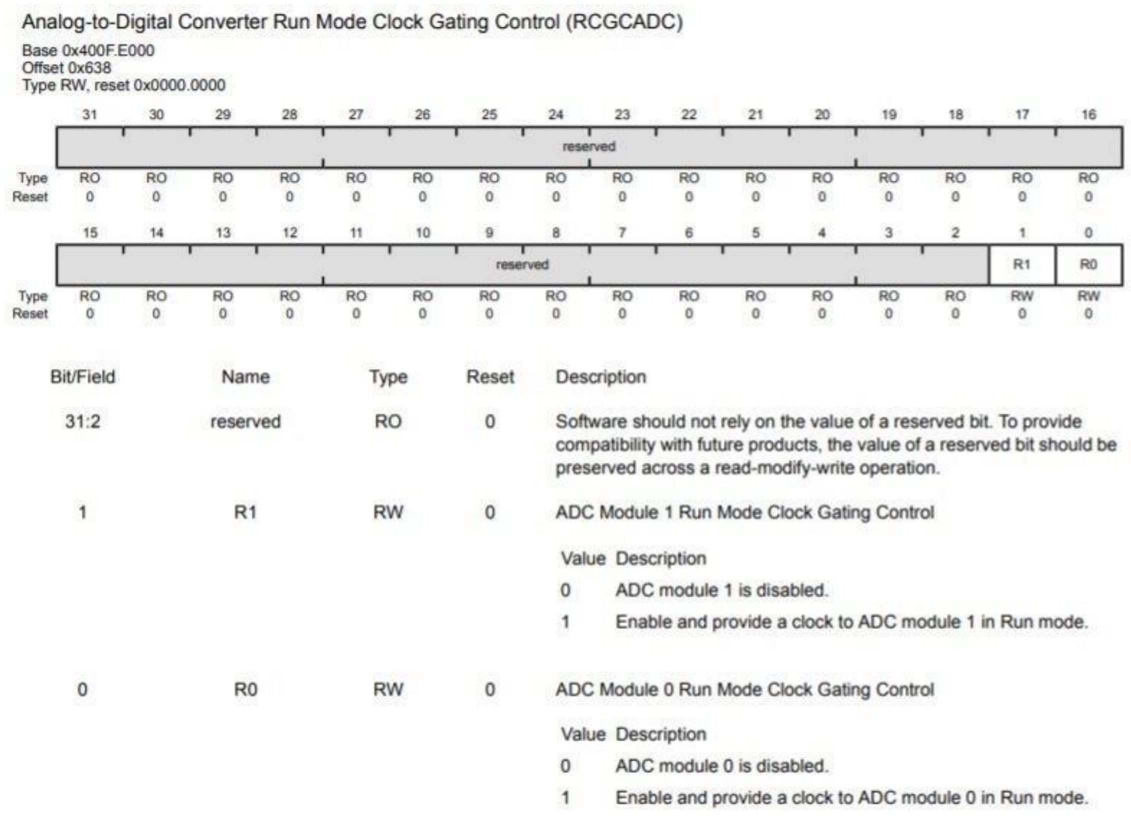
TM4C123G ADC Configuration and Initialization steps

In this section, we will discuss the different registers that are used to configure ADC input channels and sample sequencers of TM4C123 Tiva C Launchpad.

Followings are the steps to enable ADC module of TM4C123 Tiva C Launchpad:

Enable Clock to ADC

First, we enable the clock to ADC module and to the GPIO pin which is used as an analog input. RCGCADC register is used to enable the clock to ADC0 and ADC1 modules. Bit0 of RCGCADC enables ADC0 and bit1 of RCGCADC register enables ADC1. Setting the corresponding bit enables and clearing disables the clock to the corresponding analog to digital converter.



Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)

Base 0x400F.E000
Offset 0x638
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RW	0	ADC Module 1 Run Mode Clock Gating Control Value Description 0 ADC module 1 is disabled. 1 Enable and provide a clock to ADC module 1 in Run mode.
0	R0	RW	0	ADC Module 0 Run Mode Clock Gating Control Value Description 0 ADC module 0 is disabled. 1 Enable and provide a clock to ADC module 0 in Run mode.

Figure 3 Enable Clock to ADC

For example, we want to use ADC0, this line will enable clock to ADC0:

```
SYSCTL->RCGCADC |= (1UL<<0);
```

RCGCGPIO register is used to enable clock to the related GPIO port pin which will be used to measure analog input. For example, we are using analog channel zero or AN0. As we mentioned earlier, AN0 takes analog signals from the PE3 pin of PORTE. Setting the 5th bit of RCGCGPIO register enables the clock to PORTE of TM4C123 microcontroller.

```
SYSCTL->RCGCGPIO |= (1UL<<5);
```

Configure PORT as an Analog Pin
The next step is to configure the analog channel corresponding GPIO pin as an analog pin. To enable alternate function of PE3 pin as an analog input, three registers are used.

- AFSEL: This register selects alternate functions of each GPIO pin. Because each pin is multiplexed with different peripherals. This line enables the alternate function of PORTE pin 3 or PE3.

```
GPIOE->AFSEL |= (1UL<<3);
```

- DEN : This register enables or disables the digital functionality of the corresponding GPIO port. This line disables the digital functionality of PE3 pin.

```
GPIOE->DEN &= ~(1UL<<3);
```

- AMSEL: This register is used to enable analog function of GPIO pins. We are using PE3 pin for AN0.Therefore, we must enable PE3 analog function by setting the 3rd bit of AMSEL register.

```
GPIOE->AMSEL |= (1UL<<3);
```

Sample Sequencer Configuration
As we mentioned earlier, the sample sequencer is part of ADC modules of TM4C123 microcontroller. It is used to get ADC samples and stores the conversion results in FIFO.

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);
```

Sampling Option
ADCEMUX register selects the trigger option to start sampling of an input signal for each sample sequencer. Trigger event sources are processor, PWM, analog comparators, external interrupts, and software. The default trigger option is by software. This line selects a software event as a start conversion trigger.

```
ADC0->EMUX &= ~0xF000;
```

Analog Channel Selection
TM4C123G microcontroller provides 12 analog channels. ADCSSMUXn registers (where n=0, 1,2,3) select analog channels for sample sequencers. For example, if we want to use SS3 and analog channel A0, this line will select AN0 analog channel for sample sequencer 3 or SS3.

```
ADC0->SSMUX3 = 0;
```

ADC Sampling Rate Setting

ADCPC register is used to set the sampling rate of ADC. The Analog to digital converter module of the TM4C123G microcontroller supports a sampling rate from 125 KSPS to 1 MSPS. First four bits of the ADCPC register are used to set the sampling rate. This line sets the sampling rate to 250ksps.

```
ADC0->PC = 0x3;
```

After ADC initialization and configuration, set the SS3 bit of ADCPSSI register to start ADC conversion for sample sequencer 3.

```
ADC0->PSSI |= (1UL<<3);
```

If you are using ADC1 or other sample sequencers, select the corresponding bit of ADCPSSI register.

ADCRIS register provides raw interrupt signal for each sample sequencer on sample conversion completion. INR3 bit of ADCRIS register raw interrupt status of SS3. If you are using a polling method to get ADC value, you can keep polling this bit and read the data whenever the INR3 bit becomes one.

ADCSSFIFO0,ADCSSFIFO1, ADCSSFIFO2 and ADCSSFIFO3 registers contain the conversion result of ADC sample sequencers for SS0, SS1, SS2 and SS3 respectively. 12 least significant bits of these registers store conversion results.

How to use TM4C123G Tiva Launchpad ADC

we will see a demo code of TM4C123G microcontroller ADC. In this code, we configure and initialize the AN0 channel with the ADC0 module, and sample sequencer 3 is used to take input signal samples.

Schematic Diagram

We are using Analog channel zero or AN0 which takes input from PE3 pin. PE3 is a pin number three of PORTE of TM4C123 microcontroller.

In this schematic diagram, we are using a variable resistor to provide variable voltage input signal to AN0. Connect the middle terminal of a potentiometer with PE3/AN0 and other two terminals with 3.3 volts and ground terminal respectively.

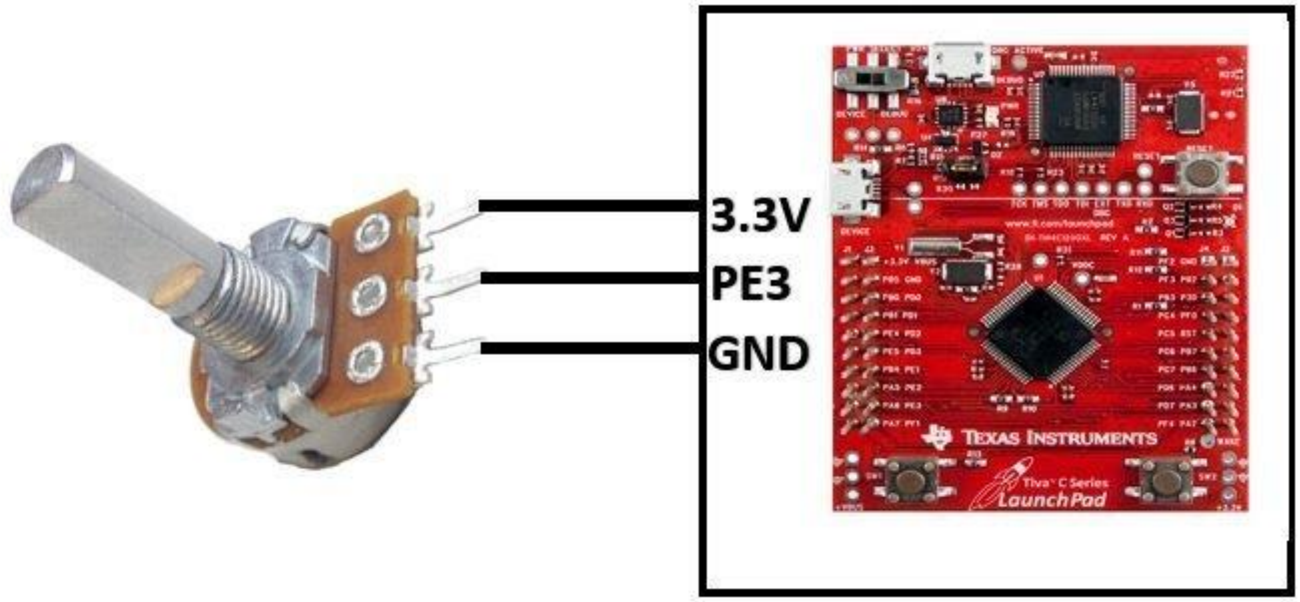


Figure 4 Schematic Diagram

ADC Code TM4C123G

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.[2]

```
/* 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*/
}
}
```

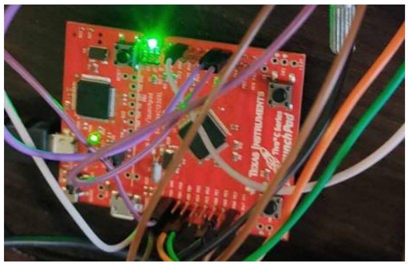


Figure 5 Example 1 output

Now create a new project in Keil uvision IDE and upload this ADC code to TM4C123G Tiva Launchpad.

Rotate potentiometer to change the input signal value. You will notice that the green LED of TM4C123G Tiva Launchpad turns on when the digital value crosses 2047.

Enable TM4C123G ADC Interrupt

In the last example, we used a polling method to measure analog signal value with TM4C123 microcontroller ADC. But the polling method is not an efficient method and it's a wastage of microcontroller processing time and resources. Instead of using a polling method where we keep polling for ADC conversion to complete using bit3 of ADCRIS register, we can use the ADC interrupt handler function to read the ADC conversion value.

As we have seen inn last example, we keep polling for ADC conversion to complete using this line:

```
while((ADC0->RIS & 8);
```

TM4C123 microcontroller keeps executing this loop and will not perform any useful function until ADC conversion is not completed. Hence, to avoid this wastage of microcontroller processing time, we can enable ADC data reception using ADC0 handler function.

Enable TM4C123 ADC Interrupt Mask

In order to enable ADC0 interrupt, ADC Interrupt Mask (ADCIM) is used. Setting bit3 of the ADCIM register enables the SS3 Interrupt Mask.

As you know that in ARM Cortex-M4 microcontroller, NVIC or nested vectored interrupt controller manages all interrupts. TM4C123 microcontroller supports 78 peripheral interrupts which are also known as IRQs. Before using each peripheral interrupt handler, we should enable each peripheral interrupt request to each NVIC using NVIC interrupt control register. The interrupt number of ADC0SS3_IRQn is 17. This line enables ADC0 sequencer 3 interrupt in NVIC.

```
NVIC->ISER[0] |= 0x00010001; /* enable IRQ17 for ADC0SS3*/
```

Exception numbers are defined inside the header file of TM4C123GH6PM microcontroller and interrupt vector table.

The next step is to find the name of the interrupt handler function of ADC0SS3_IRQn inside the startup file of TM4C123G microcontroller. If you check startup file, you will find its name as shown in figure below:

```
 ADC0Seq3_Handler ()
```

Now receive data from the FIFO register of SS3 inside this function. This ADC0Seq3_Handler () function will execute whenever interrupt occurs due to sample sequencer 3.

TM4C123G ADC Interrupt Code [3]

```
/* 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*/
}
}
}
```

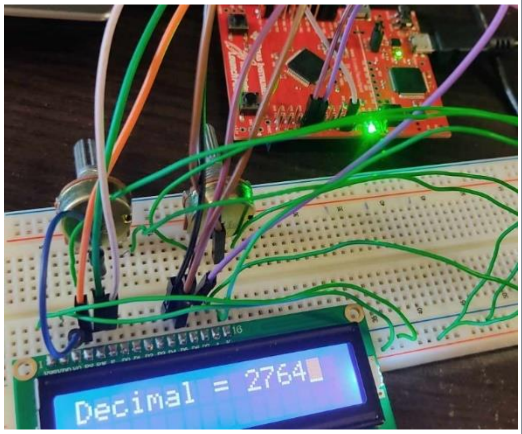


Figure 6 output for second example

Lab work

Part 1

Modify 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.

```
/* 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>
#include <stdlib.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 |= 0x0A; //set red AND green pin as a digital output pin
 GPIOF->DEN |= 0x0A; // Enable red AND green pins 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 >= 1000)
 GPIOF->DATA = 0x02; /* turn on red LED*/

 else if(adc_value < 1000)
 GPIOF->DATA = 0x08; /* turn green LED*/
 }
}
```

What this code does is that it checks the adc value and depend on it the green will be on when its less than 1000, and the red only is on when its more than 1000.

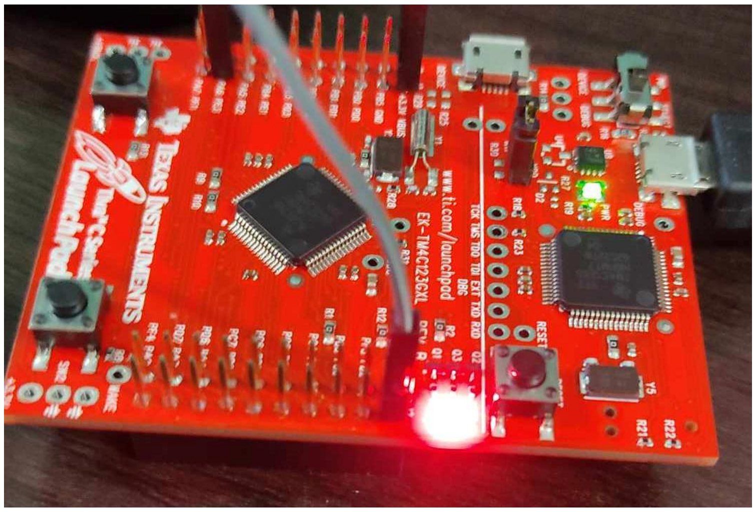


Figure 7 output 1 for labwork 1

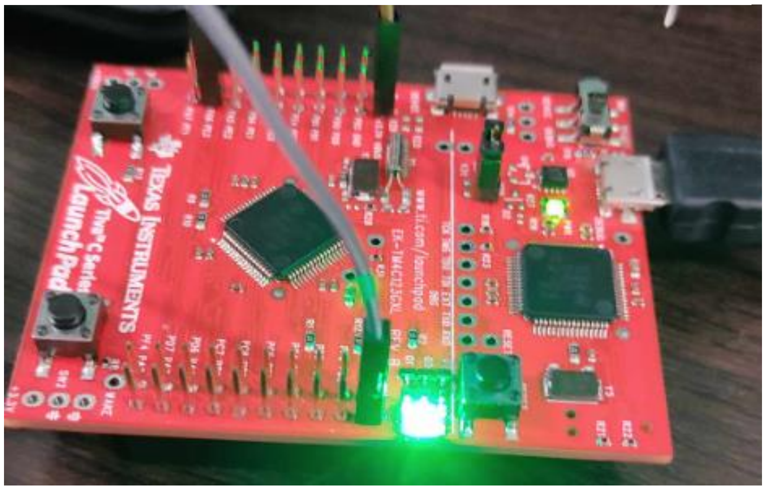


Figure 8 output 2 for labwork 1

Part 2

Modify 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 volts
```

```
/* 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>
#include <stdlib.h>


#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)
//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 = "Tiva C starting"; //Write any string you want to display on LCD
char s[20];
char de[20];
volatile float voltage;

            LCD4bits_Init(); //Initialization of LCD
 delayMs(500); //delay 500ms for LCD initialization
 LCD4bits_Cmd(0x01); //Clear the display
 delayMs(500);
 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
 /* 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)
 {
         voltage = (adc_value * 0.0008);

/*control Green PF3->LED */
 /* convert digital value back into voltage */
 sprintf(de, "Decimal = %d", adc_value);
 LCD4bits_Cmd(0x80);
 LCD_WriteString(de);
          delayMs(100);
 sprintf(s, "Voltage = %0.3f", voltage);
 LCD4bits_Cmd(0xC0);
 LCD_WriteString(s);
 if(adc_value >= 2048)
 GPIOF->DATA = 0x08; /* turn on green LED*/
 else if(adc_value < 2048)
 GPIOF->DATA = 0x00; /* turn off green LED*/
 delayMs(2000);
 }
}

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)
 delayMs(100);
LCD4bits_Cmd(0x06); //Automatic Increment cursor (shift cursor to right)
 delayMs(100);
LCD4bits_Cmd(0x01); //Clear display screen
 delayMs(100);
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(10); //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
{}
}
```

This code will erase the old screen data when interrupt to write new data in each line, the curser will be forced to move between the two lines when we need to write the voltage and decimal value, as will it will check if the value higher or lower than 2048 to turn on and off the green led.

The outputs:

1-

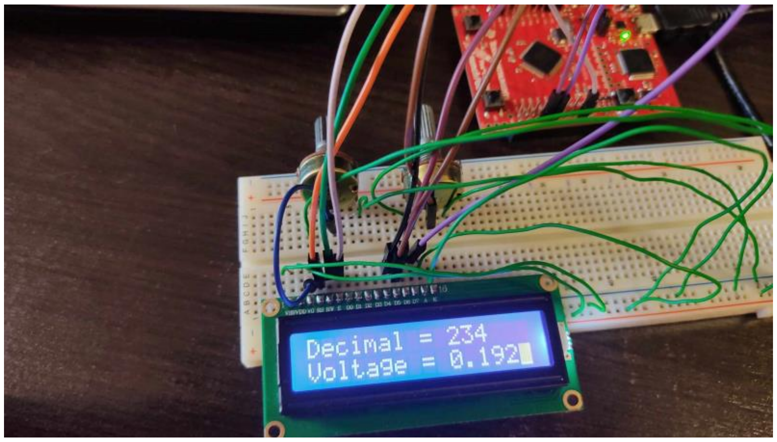


Figure 9 first output labwork 2

2-

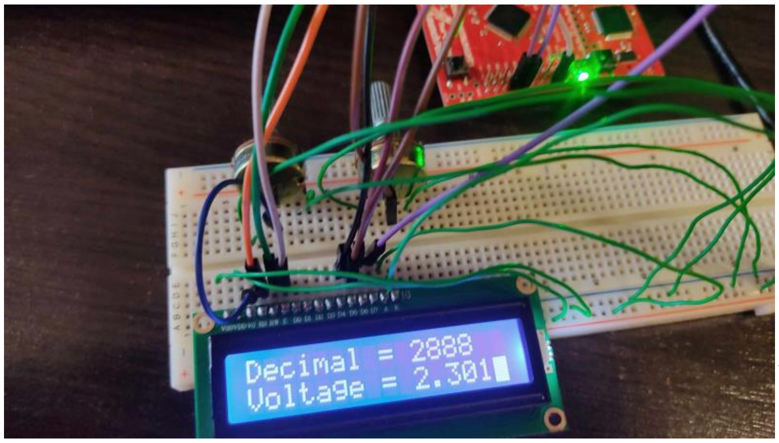


Figure 10 second output labwork 2

Conclusion

In conclusion, this experiment has taught us of how to effectively utilize the Analog to Digital Converter (ADC) module within the TM4C123GH6PM microcontroller, utilizing the Tiva C Launchpad as our hardware platform.

One of the important features of the TM4C123GH6PM microcontroller is its inclusion of two SAR-based ADC modules, namely ADC0 and ADC1, each equipped with an array of 12 analog input channels.

The experiment guided us through steps of configuring the ADC module, which included the essential task of enabling clock sources and selecting the desired analog input channels.

As well highlighted the role of the ADC module in microcontroller-based projects, reinforcing its position as a flexible and irreplaceable tool for the conversion of analog signals into digital data and its application in real-world scenarios.

References

[1]https://microcontrollerslab.com/16x2-lcd-interfacing-with-tm4c123-tiva-launchpad-keil-uvision/

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

[3]Lab manual for experiment 10