BIRZEIT UNIVERSITY

Electrical and Computer Engineering

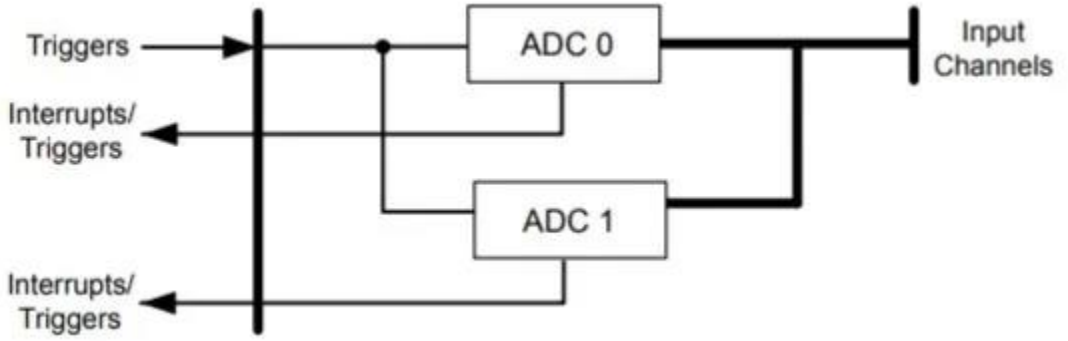
Computer Design Lab – ENCS4110

ADC– Measure Analog Voltage Signal

In this experiment, we will 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.

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:



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.

Analog channel	AN0	AN1	AN2	AN3	AN4	AN5	AN6	AN7	AN8	AN9	AN10	AN11
Pin name	PE3	PE2	PE1	PE0	PD3	PD2	PD1	PD0	PE5	PE4	PB4	PB5
Pin no	6	7	8	9	64	63	62	61	60	59	58	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:

$$\text{Resolution} = 3.3 \text{ volts} / 4095 = 0.8\text{mV}$$

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.

$$\text{Analog voltage} = 2048 \times 0.8 = 1.65\text{V}$$

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

M4C123GH6PM 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.

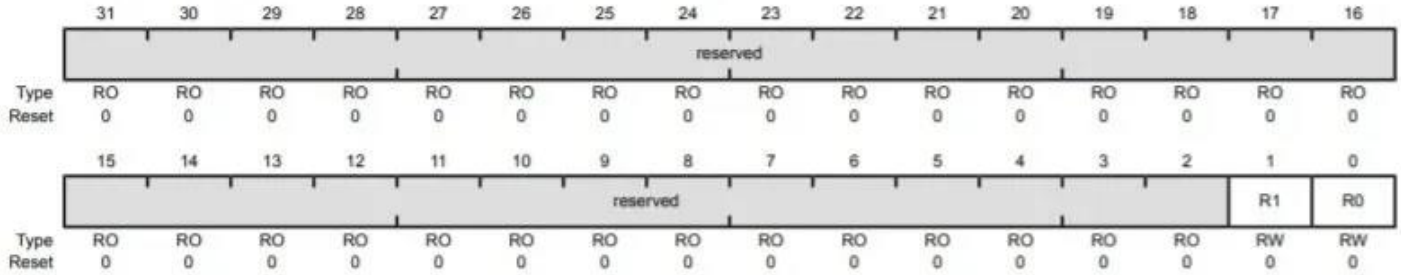
Followings are the steps to enable ADC module of TM4C123 Tiva C Launchpad:

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



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.

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

```
SYSCTL->RCGCADC |= (1<<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.

Run mode clock gating control Reg.

```
SYSCTL->RCGCGPIO |= (1<<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.

analog function select Register

- **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 |= (1<<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 &= ~(1<<3);
  ```

analog mode select Register

- **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 |= (1<<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 experiment. 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 &= ~(1<<3);
```

Sampling Option

event multiplexer select reg.

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. ADC-SSMUXn 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.

sample sequencer mux register

```
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 |= (1<<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.

ADC sample sequencer result FIFO

ADCSSFIFO0,ADC-SSFIFO1, ADC-SSFIFO2 and ADC-SSFIFO3 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.

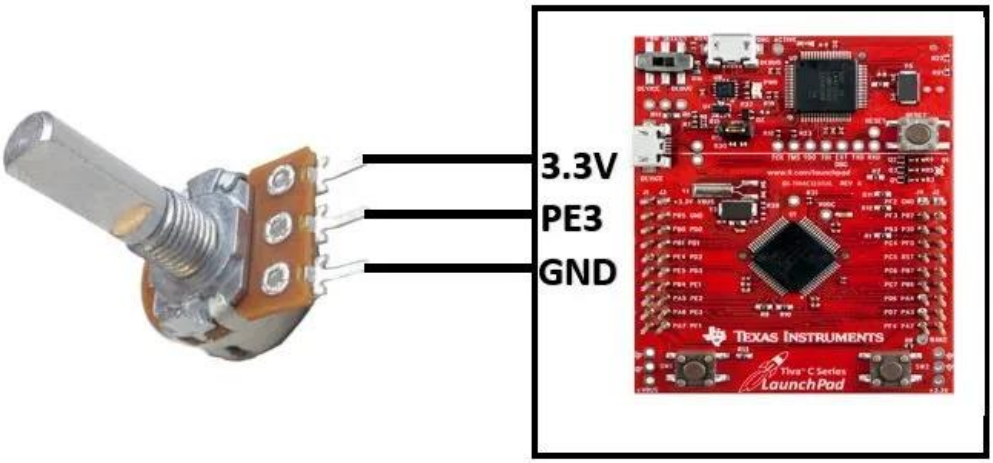
How to use TM4C123G Tiva Launchpad ADC

In the following 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.



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.

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

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 previous example, we used a polling method to measure analog signal value with TM4C123 microcontroller ADC. However, 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 in the previous 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

```
/* 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*/
  }
}

}
```

In-Lab task:

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.

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
```

Keil uvision v5 Instructions

Note: Follow these steps to build above code with Keil uvision version 5.0 or higher:

The SystemInit() function in the new system_TM4C123.c is that it configures the clock generation which results in a different system clock rate than the default 16 MHz frequency. But the above sample code works on the default 16 MHz system clock. Therefore, the above program creates timing issues with Keil v5. You may retain the default system clock rate in Keil v5, by the following steps:

1. Expand the Project->Device to show system_TM4C123.c (startup)
2. Double click to open the file in the editor window
3. Find the line “#define CLOCK_SETUP 1” as the figure below
4. Comment out the line
5. Rebuild the project