Práctica 4 ADC

Fecha: 29/01/22

Alumno:

Sosa Zepeda Yessica

PRÁCTICA 4.

ADC

INTRODUCCIÓN

ADC es el acrónimo de conversión analógica-digital. Es un dispositivo electrónico que convierte una señal analógica (continua) en una señal digital (discreta).

La señal analógica puede ser una tensión, una corriente o una señal de cualquier otro tipo que varía continuamente con el tiempo. El ADC muestrea la señal analógica en intervalos regulares y la convierte en una secuencia de números digitales.

El ADC es ampliamente utilizado en sistemas electrónicos para convertir señales analógicas en un formato que puede ser procesado y almacenado por un microcontrolador o una computadora. Esto permite que los sistemas electrónicos interactúen con el mundo real y reciban información de sensores o controlen actuadores y dispositivos externos.

ADC es un dispositivo electrónico que convierte una señal analógica en una señal digital. Es ampliamente utilizado en sistemas electrónicos para permitir que los sistemas interactúen con el mundo real y reciban información de sensores o controlen actuadores y dispositivos externos.

MARCO TEÓRICO

TIVA TM4C123GH6PM

Es un microcontrolador de 32 bits basado en el procesador ARM Cortex-M4 y ofrece una amplia gama de características para aplicaciones embebidas, incluyendo una memoria flash de 256 KB y una memoria RAM de 32 KB.

El TM4C123GH6PM es un microcontrolador de bajo costo y de baja potencia que

es adecuado para una amplia gama de aplicaciones, incluyendo sistemas de control de motores, sistemas de automatización industrial, sistemas de monitoreo de sensores y sistemas de control de procesos.

El microcontrolador incluye una amplia gama de periféricos integrados, como UART, SPI, I2C, ADC, PWM, timers, entre otros. Además, también ofrece soporte para una amplia gama de sistemas operativos embebidos, como FreeRTOS, embOS y Keil RTX5.

TM4C123GH6PM es un microcontrolador versátil y de bajo costo que ofrece una amplia gama de características y funcionalidades para aplicaciones embebidas.

METODOLOGÍA

Se implementó el siguiente código de programación en Visual Studio:

```
extern void Configura_Reg_ADC0(void)
{
    /*
Usando el modulo 0 y 1 , configurar la tarjeta a la frecuencia asignada,
para adquirir 6 señales analógicas a una velocidad de 1 millón de muestras
por segundo, por los canales asignados  y guardar los valores en un arreglo
para ser enviados con un botones  externos asociado al gpio D a través del
protocolo de comunicación asíncrona a una velocidad de 115200 todo esto usando
interrupciones.


    */
    //pag352 Habilitar y proporcionar un reloj al Modulo ADC0 Y ADC1
    SYSCTL->RCGCADC |= (1<<0) | (1<<1) ; //Modulo 0 y 1

    //Pag 406 (RGCGPIO) Habilitar Puertos a utilizar
    //                    F     E     D     C     B     A
    SYSCTL->RCGCGPIO |= (0<<5)|(1<<4)|(1<<3)|(0<<2)|(1<<1)|(0<<0);

    //Pag 663 (GPIODIR) Habilta los pines como I/O un cero para entrada y un uno para s
    //PARA ADC TIENEN QUE SER ENTRADAS
    //EL GPIO -DIR se hace para cada puerto
    GPIOB_AHB->DIR = (0<<5) ; //PB5
    GPIOD_AHB->DIR = (0<<0) | (0<<1) | (0<<3); //PD0 PD1 PD3
    GPIOE_AHB->DIR = (0<<1) | (0<<4); //PE1 PE4

    //(GPIOAFSEL) pag.672 Enable alternate función para que el modulo analógico tenga c
    //EL GPIO -AFSEL se hace para cada puerto
    GPIOB_AHB->AFSEL =  (1<<5); //PB5
    GPIOD_AHB->AFSEL =  (1<<0) | (1<<1 ) | (1<<3 );//PD0 PD1 PD3
    GPIOE_AHB->AFSEL =  (1<<1) | (1<<4 ); //PE1 PE4
```

```
GPIOE_AHB->AFSEL = (1<<1) | (1<<4 ); //PE1 PE4

//(GPIODEN) pag.682 desabilita el modo digital
GPIOB_AHB->DEN = (0<<5) ; //PB5
GPIOD_AHB->DEN = (0<<0) | (0<<1) | (0<<3); //PD0 PD1 PD3
GPIOE_AHB->DEN = (0<<1) | (0<<4); //PE1 PE4

//*************Pag 787 GPIOPCTL registro combinado con el GPIOAFSEL y la
GPIOE_AHB->PCTL = GPIOE_AHB->PCTL & (0xFF00FFFF);

//(GPIOAMSEL) pag.786 Con el 1 habilitamos la función analogica
GPIOB_AHB->AMSEL = (1<<5); //PB5
GPIOD_AHB->AMSEL = (1<<0) | (1<<1 ) | (1<<3 );//PD0 PD1 PD3
GPIOE_AHB->AMSEL = (1<<1) | (1<<4 ); //PE1 PE4


-------------------------------------------------------------------------
    //Configuración MODULO 0, Utilizaré el sec0 y el sec2

/Pag 891 El registro (ADCPC) establece la velocidad de conversión por segu
//Voy a utilizar ADC0, Y ADC1, será a 1 millon de muestras por segundo =
ADC0->PC = (1<<2)|(1<<1)|(1<<0);// 1msps

/Pag 841 Este registro (ADCSSPRI) configura la prioridad de los secuenciac

//ADC0->SSPRI = 0x3210; //Por default
ADC0->SSPRI = 0x0000;


Pag 821 (ADCACTSS) Este registro controla la activación de los secuenciado
//Primero deshabilito, para poder configurar
ADC0->ACTSS  =  (0<<3) | (0<<2) | (0<<1) | (0<<0);
```

```
  ADC0->ACTSS  =   (0<<3) | (0<<2) | (0<<1) | (0<<0);

  //Pag 1091 Este registro (ADCEMUX) selecciona el evento que activa la conversión (trigger) EL TRI
  ADC0->EMUX  |= (0x0000);



  //Sec0 voy a mandar 3 señales (11,9,6)
  //Sec2 voy a mandar 2 señales (7,4)

  //-------------------------------------------------------------------------
  //Selector 0
  //Pag 1129 Este registro (ADCSSMUX0) define las entradas analógicas con el canal y secuenciador s
  ADC0->SSMUX0 = 0x0000069B; // SEÑALES 11, 9, 6

  //pag 868 Este registro (ADCSSCTL2), configura el bit de control de muestreo y la interrupción //
  ADC0->SSCTL0 =  0X00000644;   //EL 6 se pone porque ES EL ULTIMO QUE VAS A LEER

  //-----------------------------------------------------------------------------------
//Selector 2
    //Pag 1129 Este registro (ADCSSMUX0) define las entradas analógicas con el canal y secuenciador
  ADC0->SSMUX2 = 0x0047; // SEÑALES 11, 9, 6

  //pag 868 Este registro (ADCSSCTL2), configura el bit de control de muestreo y la interrupción //
  ADC0->SSCTL2 =  0X0064;   //EL 6 se pone porque ES EL ULTIMO QUE VAS A LEER

  /* Enable ADC Interrupt */ //PONER 1
  ADC0->IM |= (1<<0)|(1<<2); /* Unmask ADC0 sequence 2 interrupt pag 1082*/
  //NVIC_PRI4_R = (NVIC_PRI4_R & 0xFFFFFF00) | 0x00000020;
  //NVIC_EN0_R = 0x00010000;
  //Pag 1077 (ADCACTSS) Este registro controla la activación de los secuenciadores
```

```
    ADC0->ACTSS = (0<<3) | (1<<2) | (0<<1) | (1<<0); //SOLO ACTIVO EL SEC QUE ME TOCA
    ADC0->PSSI |= (1<<0) | (1<<2); //Para inicializar el muestreo en el secuenciador

//------------------------------------------------------------------------------
            //Configuración MODULO 1, Utilizaré el sec3

    //Pag 891 El registro (ADCPC) establece la velocidad de conversión por segundo
    //Voy a utilizar ADC0, Y ADC1, será a 1 millon de muestras por segundo = 0x7 = 0111
    ADC1->PC = (1<<2)|(1<<1)|(1<<0);// 1msps

    //Pag 841 Este registro (ADCSSPRI) configura la prioridad de los secuenciadores
    ADC1->SSPRI = 0x000;  //Por default

    //Pag 821 (ADCACTSS) Este registro controla la activación de los secuenciadores, oc
    //Primero deshabilito, para poder configurar
    ADC1->ACTSS  =  (0<<3) | (0<<2) | (0<<1) | (0<<0);

    //Pag 1091 Este registro (ADCEMUX) selecciona el evento que activa la conversión (t
    ADC1->EMUX  |= (0x0000);
//-----------------------------------------------------------------------
//SEC3 SEÑAL 2
    //Pag 1129 Este registro (ADCSSMUX2) define las entradas analógicas con el canal y
    ADC1->SSMUX3 = 0x0002; //PRIMER MUESTREO AN9 SEGUNDO AN8 0X0654

    //pag 868 Este registro (ADCSSCTL2), configura el bit de control de muestreo y la i
    ADC1->SSCTL3 = 0x0006 ; //  EL 6 ES EL ULTIMO QUE VAS A LEER

    /* Enable ADC Interrupt */ //PONER 1
    ADC1->IM |= (1<<3); /* Unmask ADC0 sequence 2 interrupt pag 1082*/

    //NVIC_PRI4_R = (NVIC_PRI4_R & 0xFFFFFF00) | 0x00000020;
```

```
    //NVIC_EN0_R = 0x00010000;
    //Pag 1077 (ADCACTSS) Este registro controla la activación de los secue
    ADC1->ACTSS = (1<<3) | (0<<2) | (0<<1) | (0<<0); //SOLO ACTIVO EL SEC 0
    ADC1->PSSI |= (1<<3); //Para inicializar el muestreo en el secuenciador
}
/*
extern void ADC0_InSeq2(uint16_t *Result){

    //ADC Processor Sample Sequence Initiate (ADCPSSI)
        ADC0->PSSI = 0x00000004;
        while((ADC0->RIS&0x04)==0){}; // espera al convertidor
        Result[1] = ADC0->SSFIFO2&0xFFF; // Leer el resultado almacenado e
        Result[0] = ADC0->SSFIFO2&0xFFF;
        printChar('A');
        ADC0->ISC = 0x0004; //Conversion finalizada

}*/
//MOD0 SEC0 Y SEC2 MOD1 SEC3
extern void Lec_ADC(uint16_t data[])
{
    //Selector 0
    //ADC0->PSSI |= (1<<0);
    delay_ms(1);

    while((ADC0->RIS&0x01)==0); // espera al convertidor
    delay_ms(1);
    while(ADC0->SSOP0&(1<<0)==(1<<0))
    data[0] = ADC0->SSFIFO0&0xFFF; // Leer el resultado almacenado en la
    delay_ms(1);
    while(ADC0->SSOP0&(1<<4)==(1<<4))
    data[1] = ADC0->SSFIFO0&0xFFF;
```

```
delay_ms(1);
while(ADC0->SSOP0&(1<<8)==(1<<8))
data[2] = ADC0->SSFIFO0&0xFFF;
delay_ms(1);

ADC0->ISC |= (1<<0); //Conversion finalizada
delay_ms(1);

//Selector 2
//ADC0->PSSI |= (1<<2);
delay_ms(1);

while((ADC0->RIS&0x02)==0); // espera al convertidor
delay_ms(1);
while(ADC0->SSOP2&(1<<0)==(1<<0))
data[3] = ADC0->SSFIFO2&0xFFF; // Leer el resultado al
delay_ms(1);
while(ADC0->SSOP2&(1<<4)==(1<<4))
data[4] = ADC0->SSFIFO2&0xFFF;
delay_ms(1);
while(ADC0->SSOP2&(1<<8)==(1<<8))
data[5] = ADC0->SSFIFO2&0xFFF;
delay_ms(1);

ADC0->ISC = (1<<2); //Conversion finalizada
delay_ms(1);

//Selector 3
//ADC1->PSSI = (1<<3);
delay_ms(1);

while((ADC1->RIS&0x04)==0); // espera al convertidor
```

```
  delay_ms(1);
  while(ADC1->SSOP3&(1<<0)==(1<<0))
  data[0] = ADC1->SSFIFO3&0xFFF; //  Leer  el resultado almacenad
  delay_ms(1);
  while(ADC1->SSOP3&(1<<4)==(1<<4))
  data[1] = ADC1->SSFIFO3&0xFFF;
  delay_ms(1);
  while(ADC1->SSOP3&(1<<8)==(1<<8))
  data[2] = ADC1->SSFIFO3&0xFFF;
  delay_ms(1);

  ADC1->ISC = (1<<3);  //Conversion finalizada
  delay_ms(1);
```

Figura 1. Código en Visual Stud

```
2
3   char data_str[32] = " ";
4   uint16_t adc_data[adc_canales] = {0};
5   uint8_t = 0;
6
7   int main(void)
8   {
9
10      uint16_t Result[2];
11      float valor;
12      float valor1;
13      Configurar_PLL();  //Confiuracion de velocidad de reloj
14      Configura_Reg_ADC0();
15      Configura_Reg_ADC1();
16      Configurar_UART0();
17      Configurar_ADCleercanal();
18      //printString("3");
19      while(1)
20      {
21          PWM0
22          ADC0_InSeq2(Result); //llamada a la conversion por proces
23          valor=(float)(((Result[0]))*3.3)/4096;
24          valor1=(float)(((Result[1]))*3.3)/4096;
25
26
27      }
28
```

Figura 2. Código en Visual Studio

CONCLUSIÓN

El ADC forma el corazón de muchos instrumentos digitales comunes, como voltímetros, osciloscopios y analizadores de espectro. También se incorporan en el front end de los circuitos digitales que procesan señales analógicas provenientes de dispositivos como micrófonos, acelerómetros, y otros transductores que necesitan convertir su salida al dominio digital para que un microprocesador pueda trabajar con los datos.

Por ello, su aplicación es de gran útilidad.