UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

MICROCONTROLADORES

REPORTE
PRÁCTICA 4
ADC

SONIA PAOLA AGUILLÓN OLAMENDI
261520

PRÁCTICA ADC

1. OBJETIVO

El alumno conocerá y aprenderá a implementar el código de programación para la configuración y uso del ADC utilizando el microcontrolador Tiva TM4C123GH6PM.

2. MARCO TEÓRICO

ADC significa Convertidor analógico a digital, permite que una MCU, que funciona con señales digitales, y cualquier otro circuito digital lea señales analógicas. Existen varios tipos de ADC, con diferentes métodos para transformar una señal analógica en digital, y con diferentes interfaces, desde salida binaria paralela hasta i2c.

Un ser MCU del mundo digital funciona solo con 1 y 0. Una señal analógica varía en voltaje con múltiples valores, por ello es importante el uso de un ADC.

El tipo de ADC en el Tiva tiene una referencia interna de 3V y ese es el voltaje máximo que puede medir, el mínimo será igual a GND.

Los ADC también tienen una resolución máxima de 12 bits, como el de Tiva, tendrá la capacidad de distinguir 4096 valores diferentes entre GND y la referencia, con una referencia de 3V, lo que significa que cada valor es de aproximadamente 0,73 mV.

Cuanto mayor sea la resolución del ADC, mejor se leerá y almacenará una señal analógica.

Características de los módulos ADC

El microcontrolador TM4C123GH6PM proporciona dos módulos ADC, cada uno con las siguientes características:

- 12 canales de entrada analógica compartidos
- ADC de precisión de 12 bits
- Configuraciones de entrada diferencial y de un solo extremo
- Sensor de temperatura interno en chip
- Tasa de muestreo máxima de un millón de muestras/segundo
- Cambio de fase opcional en el tiempo de muestra programable de 22,5º a 337,5º
- Cuatro secuenciadores programables de conversión de muestras de una a ocho entradas de largo, con FIFO de resultado de conversión correspondientes
- Control de disparo flexible

    - Controlador (software)
    - Temporizadores
    - Comparadores analógicos
    - PWM
    - GPIO
- Promedio de hardware de hasta 64 muestras
- Ocho comparadores digitales
- La alimentación y la tierra para el circuito analógico están separadas de la alimentación y la tierra digitales.
- Transferencias eficientes mediante el controlador de acceso a memoria Micro Direct (µDMA)
- Canal dedicado para cada secuenciador de muestra
- El módulo ADC usa solicitudes de ráfaga para DMA

Hay 2 módulos ADC y cada uno de 4 secuenciadores de muestra que permiten el muestreo de múltiples fuentes sin la intervención del procesador. La mayoría de los periféricos tienen pines específicos para cada entrada/salida. El ADC tiene pines específicos conectados a él, pero puede usar cualquiera de ellos para cualquier secuenciador o módulo ADC, por ejemplo, puede usar AIN9 para ADC0 o ADC1 y cualquiera de los 4 secuenciadores.

El muestreo ADC puede ser activado por varias fuentes, simplemente por software o cualquiera de los siguientes periféricos: temporizador, módulo PWM, GPIO o comparadores analógicos. El gatillo inicia una lectura preconfigurada para comenzar.

Puede disparar los diferentes módulos al mismo tiempo, pero no diferentes secuenciadores de muestra del mismo módulo. Eso es porque cada módulo solo tiene 1 convertidor. Los diferentes secuenciadores son útiles para configurar diferentes configuraciones y luego dispararlas según sea necesario, sin tener que reconfigurar todo cada vez.

Secuenciadores de muestra

El control de muestreo y la captura de datos está a cargo de los secuenciadores de muestras. Todos los secuenciadores tienen una implementación idéntica, excepto por el número de muestras que se pueden capturar y la profundidad del FIFO. Cada muestra que se captura se almacena en el FIFO. En esta implementación, cada entrada FIFO es una palabra de 32 bits, y los 12 bits inferiores contienen el resultado de la conversión.

3. ANTEDECENTES

Estudiando la datasheet se extrajeron los puntos más importantes para lograr llevar a cabo una configuración correcta del ADC del microcontrolador Tiva TM4C123GH6PM.

Características UART

Cuenta con selectores y control de trigger, así que se debe configurar que evento es el que iniciara. La más fácil es por software. Tiene 12 canales independientes. Se toman las muestras que se configuren y se reinicia la cuenta

SECUENCIADOR PAG.802

Configuración ADC

La secuencia de inicialización del ADC es la siguiente:

1. Habilitar el reloj ADC utilizando el registro RCGCADC.
2. Habilitar el reloj para los módulos GPIO apropiados a través del registro RCGCGPIO. Para averiguar qué puertos GPIO habilitar (pag.801)
3. Establecer los bits GPIO AFSEL para los pines de entrada ADC. Para determinar qué GPIO configurar, (pag.1344)
4. Configurar las señales AINx para que sean entradas analógicas borrando el bit DEN correspondiente en el registro de habilitación digital de GPIO (GPIODEN) (pag.682)
5. Deshabilitar el circuito de aislamiento analógico para todos los pines de entrada ADC que se van a utilizar escribiendo un 1 en los bits correspondientes del registro GPIOAMSEL (pag.687) en el bloque GPIO asociado.
6. Si la aplicación lo requiere, volver a configurar las prioridades del secuenciador de muestras en el registro ADCSSPRI. La configuración predeterminada tiene Sample Sequencer 0 con la prioridad más alta y Sample Sequencer 3 como la prioridad más baja.

4. METODOLOGÍA

Usando el módulo 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 botón externo asociado al gpio D a través del protocolo de comunicación asíncrona a una velocidad de 115200 todo esto usando interrupciones.

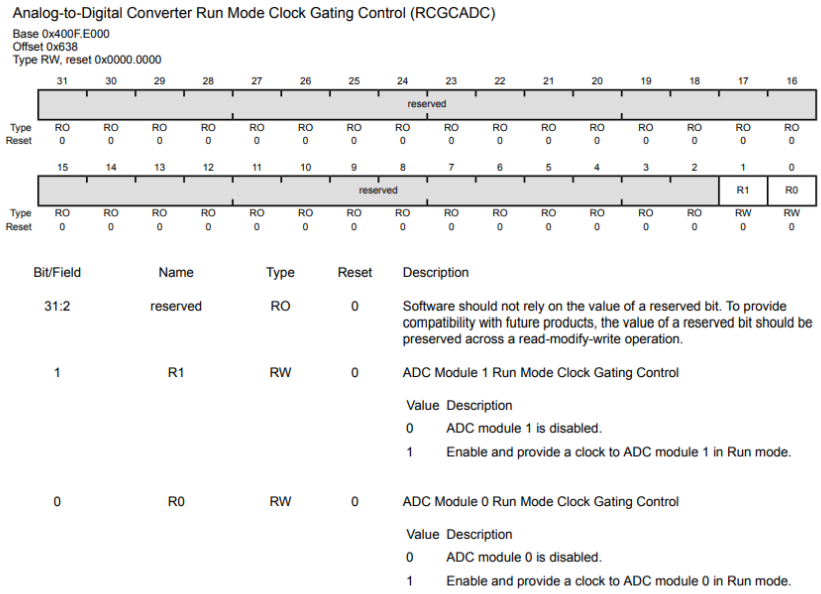
- **1,3,4,5,11,9- 55Mhz- 115200 -sec 2 ,sec 1**

Para comenzar a llevar a cabo la práctica, se inició configurando el ADC.

Primero fue necesario inicializarlo, se buscaba leer los 6 canales utilizando 2 selectores, ya que me toco el selector 1 y 2, pondré 3 señales en cada uno.

Mediante RCGCADC se habilito la señal del ADC con el registro de la figura 1, de esta manera, se puso un 1 en el bit 0 y en el bit 1.

|Register 116: Analog-to-Digital Converter Peripheral Ready PAG. 418



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.

Fig1. *Registro RCGCADC*

```
17      //Pag 396 para inicializar el modulo de reloj del adc RCGCADC
18      SYSCTL->RCGCADC = (1<<0) | (1<<1); //Se inicializa el modulo 0 y 1 del ADC
```

Fig2. *Inicialización módulo 0 y 1 ADC*

Posteriormente mediante RGCGPIO, se habilitaron los puertos que corresponden a los pines que me tocaron, para ello, fue necesario buscar en la datasheet a que puertos correspondían.

|13.2 Signal Description PAG. 801-802

Table 13-1. ADC Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	6	PE3	I	Analog	Analog-to-digital converter input 0.
AIN1	7	PE2	I	Analog	Analog-to-digital converter input 1.
AIN2	8	PE1	I	Analog	Analog-to-digital converter input 2.
AIN3	9	PE0	I	Analog	Analog-to-digital converter input 3.
AIN4	64	PD3	I	Analog	Analog-to-digital converter input 4.
AIN5	63	PD2	I	Analog	Analog-to-digital converter input 5.
AIN6	62	PD1	I	Analog	Analog-to-digital converter input 6.
AIN7	61	PD0	I	Analog	Analog-to-digital converter input 7.
AIN8	60	PE5	I	Analog	Analog-to-digital converter input 8.

Table 13-1. ADC Signals (64LQFP) ***(continued)***

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN9	59	PE4	I	Analog	Analog-to-digital converter input 9.
AIN10	58	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	57	PB5	I	Analog	Analog-to-digital converter input 11.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

Fig3. *Pines – Puertos*

Los pines que me tocaron fueron el 1,3,4,5,11,9. Por medio de la figura 1, podemos obtener los puertos a los que corresponde cada pin

PIN	PUERTO
PIN 1	PE2
PIN 3	PE0
PIN 4	PD3
PIN 5	PD2
PIN 9	PE4
PIN 11	PB5

Tabla1. *Puertos correspondientes a pines seleccionados*

Una vez sabido esto, se continuó haciendo la configuración RCGCGPIO, en la cual se habilito el reloj para los módulos GPIO del puerto B, E y D

```
//Pag 382 (RGCGPIO) Se habilitan los puertos que corresponden - Dependiendo de los pines que te tocaron ( me toco 1,3,4,5,11,9 - PUERTOS E, D Y
//                  F     E     D     C     B     A
SYSCTL->RCGCGPIO |= (0<<5)|(1<<4)|(1<<3)|(0<<2)|(1<<1)|(0<<0);
```

Fig4. *Registro RCGCGPIO*

Después se habilitaron los pines como I/O mediante GPIODIR, se utiliza un cero para entrada y un uno para salida, debido a que los ADC son entradas, se ponen los pines que nos tocaron como entradas, indicando los puertos a los que corresponden, siendo:

PUERTO E	Pin 0	Entrada
	Pin 2	
	Pin 4	
PUERTO D	Pin 2	Entrada
	Pin 3	
PUERTO B	Pin 5	Entrada

Tabla2. *Pines ADC como entradas*

```
23    IOE_AHB->DIR = (0<<0) | (0<<2) | (0<<4); //PIN 0, 2,4  - PUERTO E COMO ENTRADAS
24    IOD_AHB->DIR = (0<<3) | (0<<2); //PIN 3 Y 2 - PUERTO D COMO ENTRADAS
25    IOB_AHB->DIR = (0<<5); //PIN 5 - PUERTO B COMO ENTRADAS
```

Fig5. *Registro DIR*

Para continuar, se habilitaron las funciones alternativas para que el módulo analógico tenga control de esos pines los cuales están ubicados en el puerto E, D y B, esto se hizo mediante el GPIOAFSEL por ello se pone un 1 en los pines correspondientes a cada puerto, ya que esto indica que el pin asociado funciona como una señal periférica y está controlado por la función de hardware alternativa.

```
28    //(GPIOAFSEL) pag.770 Habilitar funciones alternativas para que el modulo analógico tenga control de esos pines (PUERTO E, D, B)
29    GPIOE_AHB->AFSEL =  (1<<0) | (1<<2) | (1<<4);
30    GPIOD_AHB->AFSEL =  (1<<2) | (1<<3);
31    GPIOB_AHB->AFSEL =  (1<<5);
```

Fig6. *Registro AFSEL*

Posteriormente se deshabilita el modo digital por medio del GPIODEN para los pines correspondientes a los puertos E, D y B, quedando de la siguiente manera:

```
33    //(GPIODEN) pag.781 desabilita el modo digital para los puertos E, D y B
34    GPIOE_AHB->DEN = (0<<4) | (0<<3) | (0<<1);
35    GPIOD_AHB->DEN =  (0<<2) | (0<<3);
36    GPIOB_AHB->DEN =  (0<<5);
```

Fig7. *Registro DEN*

Una vez deshabilitado el modo digital, se procede a habilitar la función analógica para los mismos pines por medio de GPIOAMSEL, para este registro, se tienen los dos valores siguientes.

Valor	Indica
0	Función analógica deshabilitada
1	Función analógica habilitada

Tabla3. *Valores GPIOAMSEL*

Por ello, en la configuración se tomo el valor como 1, pues se busca habilitar esta función.

```
41    //(GPIOAMSEL) pag.786 Se habilita función analogica para cada uno de los puertos
42    GPIOE_AHB->AMSEL |= (1<<0) | (1<<2) | (1<<4); //PIN 0, 2,4  - PUERTO E COMO ENTRADAS
43    GPIOD_AHB->AMSEL |= (1<<3) | (1<<2); //PIN 3 Y 2 - PUERTO D COMO ENTRADAS
44    GPIOB_AHB->AMSEL |= (1<<5); //PIN 5 - PUERTO B COMO ENTRADAS
45
```

Fig8. *Registro AMSEL*

Después por medio del registro ADCPC se estableció la velocidad de conversión de un millón por segundo para cada ADC y para cada puerto. Se continúo habilitando los secuenciadores y se le estableció la prioridad que se tiene por default, la cual es 0, es decir ira de 0,1,2,3.

El siguiente paso fue el registro ADCACTSS, lo que haremos es desactivar todos los secuenciadores para poder iniciar con la configuración.

Una vez que se desactivaron los secuenciadores, se continuó eligiendo cual será el evento que haga la conversión, para ello se eligió el trigger.

Por medio del registro ADCSSMUXn se define las entradas analógicas con el canal y secuenciador que se seleccionó, en este caso, se utiliza el secuenciador 1 y 2.

Utilizando el registro ADCSSCTLn, se configuran los bits de muestra, para esto es importante recordar que cuando se configura el ultimo nibble, hay que asegurarse de que cada interrupción tiene 4 bits, por ello cada bit debe ser configurado.

***Nota:** Nibble - medio bite

Bit 0	0

Bit 1	0 Si es el último secuenciador que vas a leer se pone 1 | Solo este se modifica en los otros escuchadores dependiendo cual es el ultimo
Bit 2	1
Bit 3 - 0 - se lee el canal que se eligió	D0

Tabla4. *Bits ADCSSCTLn*

***Nota:** Para esto, primero se especifica canales y luego se dice que se va a hacer

```
41    //(GPIOAMSEL) pag.786 Se habilita función analogica para cada uno de los puertos
42    GPIOE_AHB->AMSEL |= (1<<0) | (1<<2) | (1<<4); //PIN 0, 2,4  - PUERTO E COMO ENTRADAS
43    GPIOD_AHB->AMSEL |= (1<<3) | (1<<2); //PIN 3 Y 2 - PUERTO D COMO ENTRADAS
44    GPIOB_AHB->AMSEL |= (1<<5); //PIN 5 - PUERTO B COMO ENTRADAS
45
46    //Pag 1159 El registro (ADCPC) establece la velocidad de conversión de un millon por segundo para cada ADC y para cada puerto
47    ADC0->PC = (0<<3)|(1<<2)|(0<<1)|(1<<0);//1 Msps
48    ADC1->PC = (0<<3)|(0<<2)|(1<<1)|(1<<0);//1 Msps
49
50    //Pag 1099 Este registro (ADCSSPRI) configura la prioridad de los secuenciadores
51    ADC0->SSPRI = 0x0000;   //Se pone por desault así que la prioridad sera 0, 1,2,3
52    ADC1->SSPRI = 0x0000;   //Se pone por desault así que la prioridad sera 0, 1,2,3
53
54    //Pag 1077 (ADCACTSS) Este registro controla la activación de los secuenciadores
55    ADC0->ACTSS  =   (0<<3) | (0<<2) | (0<<1) | (0<<0); //SE DESACTIVAN PARA INICIAR LA CONFIGURACIÓN
56    ADC1->ACTSS  =   (0<<3) | (0<<2) | (0<<1) | (0<<0); //SE DESACTIVAN PARA INICIAR LA CONFIGURACIÓN
57
58    //Pag 1091 Este registro (ADCEMUX) selecciona el evento que activa la conversión (trigger)
59    ADC0->EMUX  = (0x0000);  //Se eligio trigger
60    ADC1->EMUX  = (0x0000);  //Se eligio trigger
61
62    //Pag 1129 Este registro (ADCSSMUX2) define las entradas analógicas con el canal y secuenciador seleccionado
63    ADC1->SSMUX1 = 0x00000931; //       que canales se van a leer de cada selector
64    ADC0->SSMUX2 = 0x00000B54; //       que canales se van a leer de cada selector -primer muestreo es el ultimo numero (9)
65
66    //pag 868 Este registro (ADCSSCTL2), configura el bit de control de muestreo y la interrupción
67    ADC1->SSCTL1 = 0x0644;    //AQUI SE LEE EL CANAL Y SE DICE Q
68    ADC0->SSCTL2 = 0x0644;    //AQUI SE LEE EL CANAL Y SE DICE Q
```

Fig9. *Registros AMSEL, PC, SSPRI, ACTSS, EMUX, SSMUXn y SSCTLn*

Una vez realizado esto, se habilitan las interrupciones del ADC correspondientes, en mi caso la para el ADC1 el secuenciador 1 y para el ADC0 el secuenciador 2 utilizando una operación bit a bit OR.

Con la configuración ADCACTSS se controla la activación de los secuenciadores, por ello para cada ADC se habilita el secuenciador que le corresponde.

ADC 1	Secuenciador 1
ADC 0	Secuenciador 2

Tabla5. *ADC correspondiente a secuenciador*

Por medio de la configuración PSSI se inicializa el muestreo en el secuenciador que se va a utilizar acorde al ADC y secuenciador que le corresponde a cada uno. Después por medio de PTT se establecer la cantidad de muestras, el cual tiene por default 1Maps. Con PC se seleccionó la velocidad, para ello se usó un arreglo para ver cuando se tomaran las muestras y guardarlas, ya que lo pusimos para cuando se ponga en alto se tome la muestras, al estar en alto, ósea en 1 se inicializa el muestro. En cuanto a la configuración de RIS, este nos dice que, si es un 1, es porque al menos un bit del registro vale 1, lo que significa que ya se hizo el comparador digital, este nos ayuda pues verifica si esta en 0 o si al menos 1 está trabajando aún.

La memoria FIFO (el que entra primero sale primero), está esperando a que las 8 muestras valgan 0, es decir, que ya estén guardadas las muestras en fifo para poder leerlas, posteriormente se lee cada sample del secuenciador.

Un punto muy importante es que se debe darle una pausa a los temas digitales para que todos los pasos ocurran correctamente. Cuando una operación ocurre, se le da un tiempo para que continue el siguiente paso y se pueda leer el siguiente canal.

***NOTA:** En ADC no se configura nunca la velocidad, ya que sus cuentas se deben hacer internamente

Es importante tener en cuenta, que cada uno de estos registros se hizo para ambos ADC, tanto para el ADC1 y el ADC0; y para cada selector, cada uno con las configuraciones que les correspondes. A continuación, podemos ver como es que quedo el código de configuración de ADC.

```c
#include "lib/include.h"

extern void Configura_Reg_ADC0(void)
{
    /*
    Habilitar el modulo 0 del ADC con dos canales analogicos
    en el puerto E a una velocidad de conversion de 250ksps
    dandole la mayor prioridad al secuenciador 2 con evento
    de procesador
    */
     //Pag 396 para inicializar el modulo de reloj del adc RCGCADC
    SYSCTL->RCGCADC = (1<<0) | (1<<1); //Se inicializa el modulo 0 y 1 del ADC
    //Pag 382 (RGCGPIO) Se habilitan los puertos que corresponden - Dependiendo de los pines que te tocaron ( me toco 1,3,4,5,11,9 - PUERTOS E, D Y
    //                      F     E      D       C      B     A
    SYSCTL->RCGCGPIO |= (0<<5)|(1<<4)|(1<<3)|(0<<2)|(1<<1)|(0<<0);
    //Pag 760 (GPIODIR) Habilta los pines como I/O un cero para entrada y un uno para salida (los ADC SON ENTRADAS - SE PONEN LOS PINES QUE NOS TOCA
    GPIOE_AHB->DIR = (0<<0) | (0<<2) | (0<<4); //PIN 0, 2,4  - PUERTO E COMO ENTRADAS
    GPIOD_AHB->DIR = (0<<3) | (0<<2); //PIN 3 Y 2 - PUERTO D COMO ENTRADAS
    GPIOB_AHB->DIR = (0<<5); //PIN 5 - PUERTO B COMO ENTRADAS


    //(GPIOAFSEL) pag.770 Habilitar funciones alternativas para que el modulo analógico tenga control de esos pines (PUERTO E, D, B) - SE PONEN EN 1
    GPIOE_AHB->AFSEL =  (1<<0) | (1<<2) | (1<<4);
    GPIOD_AHB->AFSEL =  (1<<2) | (1<<3);
    GPIOB_AHB->AFSEL =  (1<<5);

    //(GPIODEN) pag.781 desabilita el modo digital para los puertos E, D y B
    GPIOE_AHB->DEN = (0<<4) | (0<<3) | (0<<1);
    GPIOD_AHB->DEN =  (0<<2) | (0<<3);
    GPIOB_AHB->DEN =  (0<<5);

    //Pag 787 GPIOPCTL registro combinado con el GPIOAFSEL y la tabla pag 1808
    //GPIOE_AHB->PCTL = GPIOE_AHB->PCTL & (0xFF00FFFF);

    //(GPIOAMSEL) pag.786 Se habilita función analogica para cada uno de los puertos
    GPIOE_AHB->AMSEL |= (1<<0) | (1<<2) | (1<<4); //PIN 0, 2,4  - PUERTO E COMO ENTRADAS
    GPIOD_AHB->AMSEL |= (1<<3) | (1<<2); //PIN 3 Y 2 - PUERTO D COMO ENTRADAS
```

Fig10. *Código final ADC.c - Configuración*

```c
    GPIOB_AHB->AMSEL |= (1<<5); //PIN 5 - PUERTO B COMO ENTRADAS

    //Pag 1159 El registro (ADCPC) establece la velocidad de conversión de un millon por segundo para cada ADC y para cada puerto
    ADC0->PC = (0<<3)|(1<<2)|(0<<1)|(1<<0);//1 Msps
    ADC1->PC = (0<<3)|(0<<2)|(1<<1)|(1<<0);//1 Msps

    //Pag 1099 Este registro (ADCSSPRI) configura la prioridad de los secuenciadores
    ADC0->SSPRI = 0x0000;   //Se pone por desault así que la prioridad sera 0, 1,2,3
    ADC1->SSPRI = 0x0000;   //Se pone por desault así que la prioridad sera 0, 1,2,3

    //Pag 1077 (ADCACTSS) Este registro controla la activación de los secuenciadores
    ADC0->ACTSS  =   (0<<3) | (0<<2) | (0<<1) | (0<<0); //SE DESACTIVAN PARA INICIAR LA CONFIGURACIÓN
    ADC1->ACTSS  =   (0<<3) | (0<<2) | (0<<1) | (0<<0); //SE DESACTIVAN PARA INICIAR LA CONFIGURACIÓN

    //Pag 1091 Este registro (ADCEMUX) selecciona el evento que activa la conversión (trigger)
    ADC0->EMUX  = (0x0000);  //Se eligio trigger
    ADC1->EMUX  = (0x0000);  //Se eligio trigger

    //Pag 1129 Este registro (ADCSSMUX2) define las entradas analógicas con el canal y secuenciador seleccionado
    ADC1->SSMUX1 = 0x00000931; //       que canales se van a leer de cada selector
    ADC0->SSMUX2 = 0x00000B54; //       que canales se van a leer de cada selector -primer muestreo es el ultimo numero (9)

    //pag 868 Este registro (ADCSSCTL2), configura el bit de control de muestreo y la interrupción
    ADC1->SSCTL1 = 0x0644;    //AQUI SE LEE EL CANAL Y SE DICE Q
    ADC0->SSCTL2 = 0x0644;    //AQUI SE LEE EL CANAL Y SE DICE Q

    /* Enable ADC Interrupt */
    ADC1->IM |= (1<<1); /* Unmask ADC1 sequence 1 interrupt pag 1082*/
    ADC0->IM |= (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
    ADC1->ACTSS |= (0<<3) | (0<<2) | (1<<1) | (0<<0);
    ADC0->ACTSS |= (0<<3) | (1<<2) | (0<<1) | (0<<0);
```

Fig11. *Código final ADC.c - Configuración*

```
81      ADC0->PSSI |= (1<<2); //Se inicializa el muestreo en el secuenciador que se va a utilizar
82      ADC1->PSSI |= (1<<1);
83  }
84
85  extern void ADC_leer_canal(uint16_t data[])
86  {
87      ADC0 -> PSSI |= (1<<0); //Se inicializa el muestreo en el secuenciador que se va a utilizar
88      delay_ms(1);
89
90      while (ADC0 -> RIS & 0x01 == 0);
91      delay_ms(1);
92
93      while(ADC0  -> SSOP0 & (1<<0) == (1<<0));
94      data[0] = ADC0 -> SSFIFO0 & 0xfff;
95      delay_ms(1);
96
97      while(ADC0 -> SSOP0 & (1<<4) == (1<<4));
98      data[1] = ADC0 -> SSFIFO0 & 0xfff;
99      delay_ms(1);
100
101     while(ADC0 -> SSOP0 & (1<<8) == (1<<8));
102     data[2] = ADC0->SSFIFO0 & 0xfff;
103     delay_ms(1);
104
105     while(ADC0 -> SSOP0 & (1<<12) == (1<<12));
106     data[3] = ADC0 -> SSFIFO0 & 0xfff;
107     delay_ms(1);
108     while(ADC0 -> SSOP0 & (1<<16) == (1<<16));
109     data[4] = ADC0 -> SSFIFO0 & 0xfff;
110     delay_ms(1);
111     while(ADC0 -> SSOP0 & (1<<20) == (1<<20));
112     data[5] = ADC0 -> SSFIFO0 & 0xfff;
113     delay_ms(1);
114
115     ADC0->ISC |= (1<<0);
116     delay_ms(1);
117 }
```

Fig12. *Código final ADC.c - Configuración*

```
C ADC.c > ...
119 ////////////////////////////////////////////////////////////////////////////////////////////////////// RECIBIR CHAR UART
120 extern void ADC_leer_canal(uint16_t data[])
121 {
122     // -------------------------------------selector 0
123     ADC0 -> PSSI |= (1<<0);
124     delay_ms(1);
125
126     while (ADC0 -> RIS & 0x01 == 0);
127     delay_ms(1);
128     while(ADC0 -> SSOP0 & (1<<0) == (1<<0));
129     data[0] = ADC0 -> SSFIFO0 & 0xfff;
130     delay_ms(1);
131     while(ADC0->SSOP0 & (1<<4) == (1<<4));
132     data[1] = ADC0->SSFIFO0 & 0xfff;
133     delay_ms(1);
134     while(ADC0->SSOP0 & (1<<8) == (1<<8));
135     data[2] = ADC0->SSFIFO0 & 0xfff;
136     delay_ms(1);
137
138     ADC0->ISC |= (1<<0);
139     delay_ms(1);
140
141     // --------------------------------selector 1
142     ADC0->PSSI |= (1<<1);
143     delay_ms(1);
144
145     while (ADC0->RIS & 0x02 == 0);
146     delay_ms(1);
147     while(ADC0->SSOP1 & (1<<0) == (1<<0));
148     data[3] = ADC0->SSFIFO1 & 0xfff;
149     delay_ms(1);
150     while(ADC0->SSOP1 & (1<<4) == (1<<4));
151     data[4] = ADC0->SSFIFO1 & 0xfff;
152     delay_ms(1);
153     while(ADC0->SSOP1 & (1<<8) == (1<<8));
154     data[5] = ADC0->SSFIFO1 & 0xfff;
155     delay_ms(1);
```

Fig13. *Código final ADC.c – Selectores*

```
156
157       ADC0->ISC |= (1<<1);
158       delay_ms(1);
159   }
160   //*/
161
162
163
164
165   //*/
166
167
168
169   extern void ADC0_InSeq2(uint16_t *Result){
170
171       //ADC Processor Sample Sequence Initiate (ADCPSSI)
172         ADC0->PSSI = 0x00000004;
173         while((ADC0->RIS&0x04)==0){}; // espera al convertidor
174         Result[1] = ADC0->SSFIFO2&0xFFF; //  Leer  el resultado almacenado en la pila2
175         Result[0] = ADC0->SSFIFO2&0xFFF;
176         printChar('A');
177         ADC0->ISC = 0x0004;  //Conversion finalizada
178
179   }
180
```

Fig14. *Código final ADC.c – Selectores*

De esta manera, la configuración del ADC, quedo terminada.