



Analógico digital PIC



El CAD conversor analógico digital PIC (no todos los PIC lo tienen, para los ejemplos se utilizará el **16F877A**, permite medir señales analógicas en forma digital, para ello el PIC cuenta con pines por donde le llegará la señal analógica, estos pines deben configurarse como entradas analógicas, el conversor analógico digital PIC cuenta con un circuito que carga un condensador interno al PIC con la tensión analógica que le está llegando a la entrada analógica, luego la tensión almacenada en el condensador lo convierte en un número binario de 10 bits que representará la tensión almacenada en el condensador, este número binario se guarda en sus registros **ADRESH** y **ADRESL** de 8 bits cada uno pero estos actúan como un solo registro de 16 bits, en el registro ADRESH se guardan los bits mas significativos y en el registro ADRESL se guardan los bits menos significativos, el número que representa la tensión almacenada en el condensador y guardado en forma binaria dentro de estos registros será de 10 bits para el PIC16F877A, la cantidad de bits de este número depende del conversor analógico digital PIC del microcontrolador PIC utilizado.

El CAD conversor analógico digital PIC necesita una tensión de referencia para poder trabajar adecuadamente, esta tensión de referencia **Vref** normalmente será la tensión a la cual trabaja el PIC, aunque por programa se puede elegir otra diferente; a la relación que hay entre la tensión de referencia Vref y el máximo número binario de 8 bits $2^8-1=255=11111111_2$ de 10 bits $2^{10}-1=1023=1111111111_2$ que representará la tensión analógica se le conoce como resolución, por ejemplo para el caso del PIC16F877A se tendrá que la resolución del conversor analógico digital PIC será:

$$\text{Resolución} = V_{\text{ref}} / (2^{10} - 1) = V_{\text{ref}} / 1023$$

Si se toma como la $V_{\text{ref}}=5\text{V}$ que es la tensión adecuada a la que trabaja el PIC16F877A se tendrá que la resolución de su convertidor analógico digital PIC será:

$$\text{Resolución} = 5\text{V} / (2^{10} - 1) = 5\text{V} / 1023$$

de donde

Resolución = $0,004887585533\text{V} \approx 0,0049\text{V}$ luego para el caso del conversor de 10 bits con un voltaje de referencia de 5V será:

$$\text{Resolución} = 4,9\text{mV}$$

La resolución indica en este caso que a la tensión de referencia de 5V se le ha partido en 1023 partes iguales y cada una de esas partes equivalen a aproximadamente 4,9mV, al utilizar esto en forma digital indicará que para un 0 será 0V, si el voltaje aumenta desde 0 en 4,9mV se tendrá un 1, si aumenta 2 veces 4,9mV se tendrá un 2, si aumenta 3 veces 4,9mV se tendrá un 3, si aumenta 4 veces 4,9mV se tendrá un 4 etc.

Esto quiere decir en este caso, que al utilizar el conversor analógico digital PIC, cada vez que la tensión analógica que llegue por el pin configurado como entrada analógica aumente en 4,9mV, el número que lo representa y almacenado en los registros **ADRESH** y **ADRESL** aumentará en 1, por ejemplo cuando la tensión analógica es 0, el número binario que lo representa será 000000000, si la tensión analógica aumenta de 0 a 4,9mV el número binario que lo representa será 000000001, si la tensión analógica aumenta de 4,9mV a 9,8=2*4,9mV el número binario que representa este valor será 000000010, si la tensión analógica aumenta de 9,8V a 14,7=3*4,9mV el número binario que representa este valor será 000000011, y así hasta que la tensión analógica se haga igual a la tensión de referencia, lo que ocurrirá cuando la tensión analógica aumente desde 0 de 4,9mV en 4,9mV unas 1023 veces lo cual es 1023*4,9mV que es un poquito mas de 5V porque la resolución se redondeo, el número binario que representa a los 5V será 111111111.

En el siguiente enlace es un vídeo publicado por MrElberni, en el que se hace una introducción al uso del conversor analógico digital PIC.

Youtube mrelberni: [Conversor Analógico Digital CAD o ADC PIC parte 1](#)

El número binario que representará la tensión analógica la cual dependerá de la resolución, puede leerse desde los registros **ADRESH** y **ADRESL**, guardarlo en una variable y luego mediante operaciones matemáticas se puede hacer que represente el valor de la medida analógica que esta llegando a la entrada analógica del PIC, por ejemplo si se crea una variable de 16 bits llamada `medida_analogica` y en esta se guarda el número binario, y suponiendo como se verá líneas abajo que se ha elegido una justificación a la derecha, en el XC8 sería algo así:

`int medida_analogica;`//se declara la variable de 16 bits

`medida_analogica=ADRESH<<8;`//se le asigna el valor almacenado en ADRESH que son los 2 bits de mayor peso de los 10 totales del número binario y se hace un corrimiento de 8 bits a la izquierda.

`medida_analogica=medida_analogica+ADRESL;`//se le suma los 8 bits restantes de los 10 totales del número binario los que se encuentran almacenados en el registro ADRESL.

Luego al valor almacenado en la variable `medida_analogica` se le multiplica por la resolución obteniéndose el valor de la tensión analógica representada por el número binario

`medida_analogica=medida_analogica*Vref/1023;`

Para la utilización del convertidor analógico digital PIC del PIC16F877A se tienen 2 registros para su control, los que son el **ADCON0** y el **ADCON1**, el CAD también puede producir **interrupciones** para lo cual se utilizan los registros **INTCON**, **PIE1** y **PIR1**,

algunos microcontroladores PIC tienen además otros registros relacionados con el convertidor analógico digital PIC.

Para el almacenamiento de la tensión analógica en el condensador es necesario que pase un tiempo, a este tiempo se le llama tiempo de adquisición el cual es de aproximadamente de 20 μ s según la hoja de datos del **16F877A**, por lo que hay que esperar un tiempo mientras se carga el condensador, una vez almacenada la tensión en el condensador, a la conversión del valor analógico a su representación digital también le toma un tiempo al que se le llama tiempo de conversión, el cual depende de la velocidad de la fuente de reloj que se selecciona para la conversión, esta selección se hace por programa mediante una tabla que viene en la hoja de datos como se verá más abajo, la hoja de datos recomienda que se debe seleccionar un tiempo mínimo de conversión de 1,6 μ s, entonces si es un poquito más mejor, por ejemplo 2 μ s o 4 μ s.

En lo que sigue es necesario tener la hoja de datos del microcontrolador que se esté utilizando, lo que se comenta más abajo es para el caso del **PIC16F877A**, la forma en que se procede puede variar un poco en comparación con el conversor analógico digital PIC de otros microcontroladores PIC, pero en general la manera de proceder es muy similar y siempre hay que guiarse con la hoja de datos.

Conversor Analógico Digital PIC: El registro ADCON0

En la siguiente imagen se tiene el registro ADCON0 para el control del conversor analógico digital PIC, se pueden ver los nombres que le corresponden a cada uno de sus bits.

ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7							bit 0

Los bits 7 y 6 de este registro junto con el bit 6 del registro ADCON1 se elige el reloj, esto es entre cuanto se fraccionará la frecuencia del oscilador utilizado para que se tenga un tiempo de conversión adecuado, esto es el tiempo que tardará el PIC para realizar la conversión, además de estas opciones el conversor analógico digital pic cuenta con su propio oscilador formado por un circuito RC que también puede ser elegido mediante estos bits, en la siguiente tabla se tienen los fraccionamientos del oscilador de acuerdo a los valores que tomen estos bits o si se quiere utilizar el oscilador interno del CAD.

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

De acuerdo a la frecuencia del oscilador se obtendrá un tiempo para la conversión, ese tiempo tiene que ser mayor a 1,6us para que al leer los registros **ADRESH** y **ADRESL** el numero obtenido represente en forma adecuada el valor de la señal analógica.

Por ejemplo, si se usa un cristal con una Fosc de 4Mhz, de la tabla se puede ver que si la Fosc se divide entre 2 se tendrá el tiempo de conversión será de 0,5us, lo cual no llega a los 1,6us mínimos, si se divide entre la Fosc 4 se tendrá el tiempo de conversión será de 1us, con lo cual tampoco se llega a los 1,6us mínimos, si se divide la Fosc entre 8 se tendrá el tiempo de conversión será de 2us, con lo cual ya se ha logrado un tiempo de conversión que sobrepasa los 1,6us mínimos que se necesita, por lo que en este caso se elegiría esta opción para el tiempo de conversión y la combinación de bits serian 001, aunque se pueden elegir otros siempre y cuando se obtengan tiempos de conversiones mayores a los 1,6us.

Los bits 5, 4 y 3 son para elegir el canal analógico a utilizar, esto es el pin que previamente mediante los bits 3,2,1 y 0 del registro ADCON1 se ha configurado como entrada analógica, en el cual se leerá la señal analógica, el PIC16F877A cuenta con 8 entradas analógicas, 5 de las cuales están en el puerto A y 3 en el puerto E, los pines de las entradas analógicas se conocen como AN0, AN1, AN2, AN3, AN4, AN5, AN6 y AN7, la elección del canal a leerse se hace de acuerdo a los valores de estos bits como se muestra en la siguiente tabla.

CHS2:CHS0: Analog Channel Select bits

000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

El bit 2 se pondrá a 1 para iniciar la conversión analógica digital PIC, cuando la conversión de analógico a digital termina este bit se pone a 0 en forma automática, lo que indica que la conversión a terminado además de que si está habilitada las interrupciones por el CAD pues se producirá una interrupción.

El bit 1 no es utilizado por lo que pondrá a 0.

El bit 0 es para activar o desactivar el conversor analógico digital PIC, cuando este bit es puesto a 1 el conversor está activo y listo para usarse, si este bit es puesto a 0 el conversor estará apagado no pudiendo utilizarse.

Conversor Analógico Digital PIC: El registro ADCON1

En la siguiente imagen se tiene el registro ADCON1 para el control del conversor analógico digital PIC, se pueden ver los nombres que le corresponden a cada uno de sus bits.

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7				bit 0			

El bit 7 los registros **ADRESH** y **ADRESL** donde se guarda el número binario que representa el valor de la señal analógica convertida hacen un total de 16 bits, pero el número de la conversión solo está formado por 10 bits en este caso, por lo que 6 bits no representan nada, luego mediante este bit se elige si los 10 bits donde se guarda este número son los 10 mas significativos o los 10 menos significativos, si son los 10 bits menos significativos se dice que la justificación es a la derecha y se elige esta opción poniendo este bit a 1, pero si son los 10 bits mas significativos se dice que la justificación es a la izquierda y se elige esta opción poniendo este bit a 0; los 6 bits que no interesan siempre estarán a 0 en forma automática.

000000xxxxxxxxxx donde las x pueden ser 0 o 1, siendo este caso la justificación a la derecha cuando el bit 7 del registro ADCON1 es 1.

xxxxxxxxxx000000 donde las x pueden ser 0 o 1, siendo este caso la justificación a la izquierda cuando el bit 7 del registro ADCON1 es 0.

El bit 6 junto con los bits 7 y 6 del registro ADCON0 se utiliza para obtener el tiempo de conversión adecuado del CAD, el que tiene que ser mayor a 1,6us.

Los bits 5 y 4 no se utilizan por lo que se les pone a 0.

Los bits 3, 2, 1 y 0 son para elegir que pines serán utilizado, como entradas analógicas, se puede elegir entre todas las entradas analógica o solo algunas, también si se quiere utilizar un voltaje de referencia diferente al que trabaja el pic, para esto será necesario configurar los bits de acuerdo a la siguiente tabla.

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O

C/R = # of analog input channels/# of A/D voltage references

Se puede ver que se permiten utilizar todos los pines como entradas salidas digitales o todos como entradas analógicas, son varias opciones que se tienen para el PIC16F877A, esto puede variar para otros microcontroladores PIC.

Conversor Analógico Digital PIC: Pasos a seguir para la conversión sin interrupciones.

Para el proceso de conversión analógico digital PIC sin tomar en cuenta la interrupción que este puede producir, el fabricante recomienda que se sigan una serie de pasos los cuales son:

1. Configurar que pines serán utilizados como entradas analógicas, si se quiere una tensión de referencia diferente a la tensión de trabajo del PIC, lo cual como se ha comentado líneas arriba se hace con los bits 3, 2, 1 y 0 del registro ADCON1.
2. Elegir cual será el reloj a utilizar para obtener el tiempo de conversión adecuado, que tendrá que ser mayor a 1,6us, lo cual se hace mediante los bits 7 y 6 del registro ADCON0 junto con el bit 6 del registro ADCON1.
3. Seleccionar cual será la entrada analógica a leerse, lo cual se hace mediante los bits 5, 4 y 3 del registro ADCON0.
4. Activar el conversor analógico digital PIC poniendo a 1 el bit 0 del registro ADCON0.

5. Esperar el tiempo de adquisición necesario para que el condensador del módulo conversor analógico digital PIC se cargue en forma adecuada.

6. Poner a 1 el bit 2 del registro ADCON0 para que comience la conversión de la tensión almacenada en el condensador del conversor analógico digital PIC y se guarde este valor en forma binaria en los registros **ADRESH** y **ADRESL**.

7. Esperar a que la conversión analógica digital termine mediante la lectura del bit 2 del registro ADCON0, la conversión habrá terminado cuando el valor de este bit se ponga a 0.

8. Leer los registros **ADRESH** y **ADRESL** que es donde se ha guardado el número binario que representa el valor de la señal analógica.

Si se quiera hacer otra lectura se empezará a partir del paso 3.

Conversor Analógico Digital PIC: Pasos a seguir para la conversión con interrupciones.

Para el proceso de conversión analógico digital PIC sin tomar en cuenta la interrupción que este puede producir, el fabricante recomienda que se sigan una serie de pasos los cuales son:

1. Configurar las interrupciones poniendo los bits 7 (GIE) y 6 (PEIE) del registro INTCON y el bit 6 (ADIE) del registro PIE1 a 1 y el bit 6 (ADIF) del registro PIR1 a 0;

2. Configurar que pines serán utilizados como entradas analógicas, si se quiere una tensión de referencia diferente a la tensión de trabajo del PIC, lo cual como se ha comentado líneas arriba se hace con los bits 3, 2, 1 y 0 del registro ADCON1.

3. Elegir cual será el reloj a utilizar para obtener el tiempo de conversión adecuado, que tendrá que ser mayor a 1,6 μ s, lo cual se hace mediante los bits 7 y 6 del registro ADCON0 junto con el bit 6 del registro ADCON1.

4. Seleccionar cual será la entrada analógica a leerse, lo cual se hace mediante los bits 5, 4 y 3 del registro ADCON0.

5. Activar el conversor analógico digital PIC poniendo a 1 el bit 0 del registro ADCON0.

6. Esperar el tiempo de adquisición necesario para que el condensador del módulo conversor analógico digital PIC se cargue en forma adecuada.

7. Poner a 1 el bit 2 del registro ADCON0 para que comience la conversión de la tensión almacenada en el condensador del conversor analógico digital PIC y se guarde este valor en forma binaria en los registros **ADRESH** y **ADRESL**.

8. Esperar a que se produzca la interrupción.

9. Leer los registros **ADRESH** y **ADRESL** que es donde se ha guardado el número binario que representa el valor de la señal analógica y poner a 0 el bit 6 (ADIF) del registro PIR1.

Si se quiera hacer otra lectura se empezará a partir del paso 4.

La forma de proceder con un ejemplo para el uso del CAD con interrupción se encuentra [aquí](#).

En el siguiente vídeo publicado por MrElberni se comenta sobre el circuito que se utilizará en el ejemplo1, el tiempo de adquisición, el tiempo de captura, sobre los bits a programar de los registros ADCON0 y ADCON1.

Youtube mrelberni: [Conversor Analógico Digital CAD o ADC PIC parte 2](#)

Se harán algunos ejemplos para el uso del conversor analógico digital PIC, los 2 primeros serán hechos en el XC8 para ver la forma en que se manipulan los registros de control del CAD, el primero será realizado sin la utilización de interrupciones, mientras que en el segundo ejemplo se recurrirá al uso de las interrupciones para realizar la conversión.

Conversor Analógico Digital PIC: CAD en XC8

Para el uso del CAD en el XC8, se han definido los nombres de los registros a utilizar para el control del conversor analógico digital como ADCON0 y ADCON1, los cuales se pueden configurar en forma binaria, decimal o hexadecimal, por ejemplo en forma binaria sería algo así:

En el caso de usar un cristal de 4Mhz, para obtener un tiempo de conversión mayor a 1,6us se utilizará un reloj que sea $F_{osc}/2$ con lo cual se logra que el tiempo de conversión sea de 2us, para lograr esto se ponen los bits 7 y 6 del registro ADCON0 a 0 y el bit 6 del registro ADCON1 también a 0 según el cuadro visto líneas arriba.

Si se quiere elegir por ejemplo el canal 3 o la entrada analógica 3 AN3, el el registro ADCON0 se tendrá que poner su bit 5 a 0 y sus bits 4 y 3 a 1.

Para iniciar la conversión se pone el bit 2 del registro ADCON0 a 1, el cual luego de la conversión se pondrá automáticamente a 0.

El bit 1 del registro ADCON0 no se usa por lo que se pone a 0.

Para activa el módulo conversor analógico digital pic el bit 0 del registro ADCON0 se pondrá a 1.

ADCON0=0b01011101; //fosc/8 AN3 conversión iniciada CAD activado.

Para la justificación a la derecha del número binario de 10 bits que representa el valor analógico, el bit 7 del registro ADCON1 se pone a 1.

El bit 6 del registro ADCON1 trabaja junto con los bits 7 y 6 del registro ADCON0 para obtener el tiempo de conversión, que para este ejemplo será de 2 us.

Los bits 5 y 4 del registro ADCON1 no se usan por lo que se ponen a 0.

Con los bits 3, 2, 1 y 0 del registro ADCON1 de acuerdo a la tabla vista líneas arriba, se elige alguna de las combinaciones que incluyan el uso de la entrada analógica 3 AN3 por ejemplo 0100, con lo cual también se indica que se usarán las entradas AN1 y AN0, en este caso no hay una opción para utilizar solo la entrada AN3, cosa que en otros microcontroladores PIC si es posible.

ADCON1=0b10000100; //justificación derecha fosc/2 AN3

Si se necesita manipular solo uno de los bits de estos registros, por ejemplo para activar el CAD sería algo así:

ADCON0bits.ADON=1; //CAD activado

y para desactivarlo sería algo así:

ADCON0bits.ADON=0; //CAD desactivado

Para iniciar la conversión sería algo así:

ADCON0bits.GO=1; //Iniciar la conversión, este bit se pone automáticamente a 0 cuando la conversión termina, lo cual si se ha elegido trabajar con interrupciones pues producirá una interrupción.

Y así se pueden manipular en forma independiente cada uno de los bits de estos registros, además el MPLABX tiene una ayuda que hace aparecer estos manipuladores y poder seleccionar el que se desee utilizar.

Para guardar el número de 10 bits que representará el valor analógico, los nombres de los registros que almacenan este valor en el XC8 se han definido como ADRESH y ADRESL de 8 bits cada uno, dependiendo si la justificación se ha elegido a la derecha o a la izquierda el número se guardará de diferente manera, por ejemplo si la justificación es a la derecha, eso indica que los 2 bits mas significativos del número estarán en los bits 1 y 0 del registro ADRESH, mientras que los 8 bits menos significativos del número estarán en el registro ADRESL, luego para la lectura de este número se procede como se indica a continuación.

Se crea una variable entera que en el XC8 estas variable son de 16 bits y se le asigna el valor del registro ADRESH

int numero_cad=ADRESH; // en la variable entera numero_cad se guardan los 2 bits mas significativos del número que representa el valor analógico, estos se ubicarán en los bits 1 y 0 de la variable numero_cad, luego hay que hacer un corrimiento de 8 bits hacia la izquierda en la variable numero_cad para que estos bits se ubiquen en los bits 10 y 9 de la variable numero_cad, sería algo así

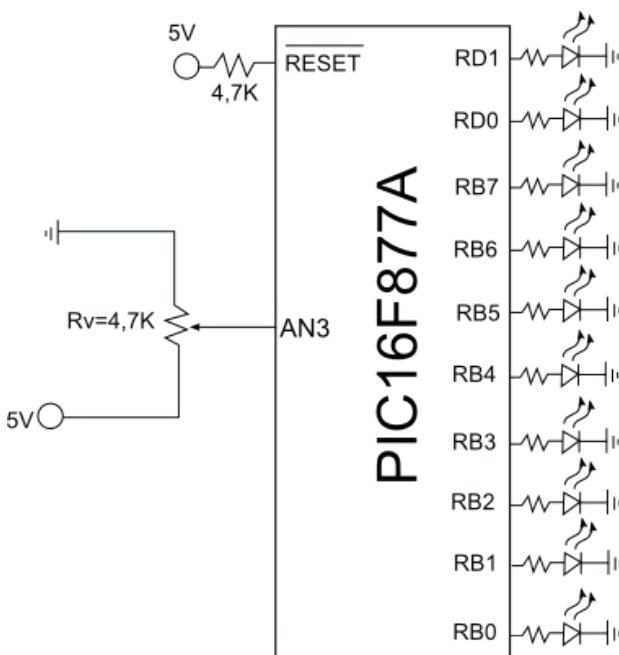
numero_cad=numero_cad<<8; //Ahora los 2 bits mas significativos del número binario que representa el valor analógico están ubicados en los bits 10 y 9 de la variable numero_cad.

Ahora a la variable `numero_cad` se le sumaran los 8 bits menos significativos del numero binario que representa el valor analógico y que está almacenado en el registro `ADRESL` como se indica

`numero_cad=numero_cad+ADRESL;` //de esta manera el número binario de 10 bits que representa el valor analógico queda almacenado o guardado dentro de la variable `numero_cad` la cual puede utilizarse al gusto según las necesidades que se tenga de este número.

Ejemplo1 En el siguiente ejemplo, mediante una resistencia variable, a la entrada analógica `AN3` del PIC16F877A se le hará llegar una señal analógica que va desde 0V a 5V, esa señal será convertida a digital cuyos valores en binario serán 0000000000 para 0V y 1111111111 para 5V los cuales serán visibles por medio de unos leds conectados a los puertos B y D, los 8 bits menos significativos serán visibles en el puerto B y los 2 bits mas significativos serán visibles en el puerto D, cuando todos los leds estén apagados esto es 0 se estará midiendo 0V y cuando todos los leds estén encendidos esto es 1023 se estará midiendo 5V, los valores binarios variarán de 0 a 1023 lo cual será visible por medio de los leds cada vez que se gire la resistencia variable para los diferentes valores de la señal analógica. En este caso no se utilizarán interrupciones.

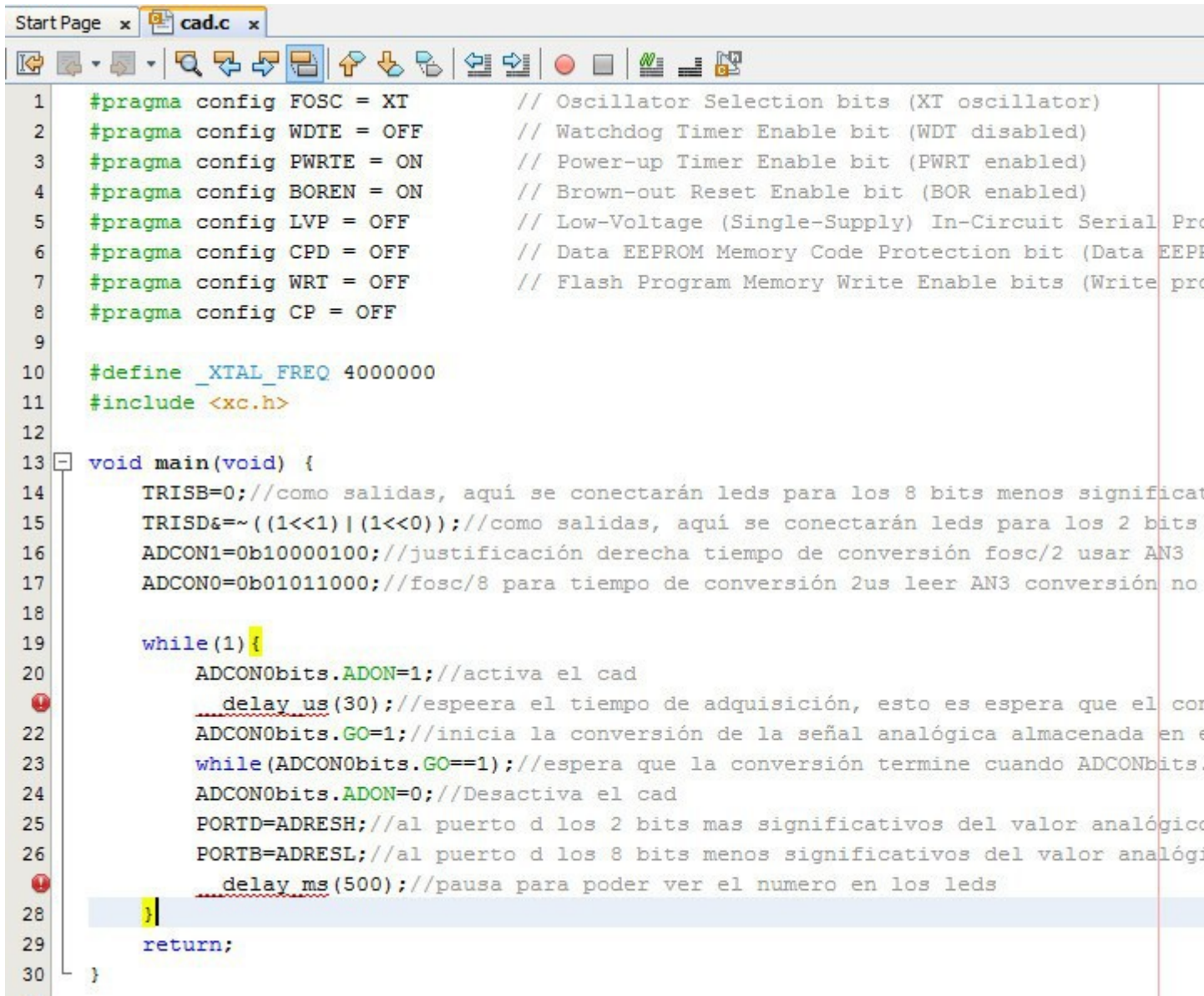
El circuito utilizado es el siguiente:



CONVERSOR ANALÓGICO DIGITAL CAD EN EL PIC 16F877A

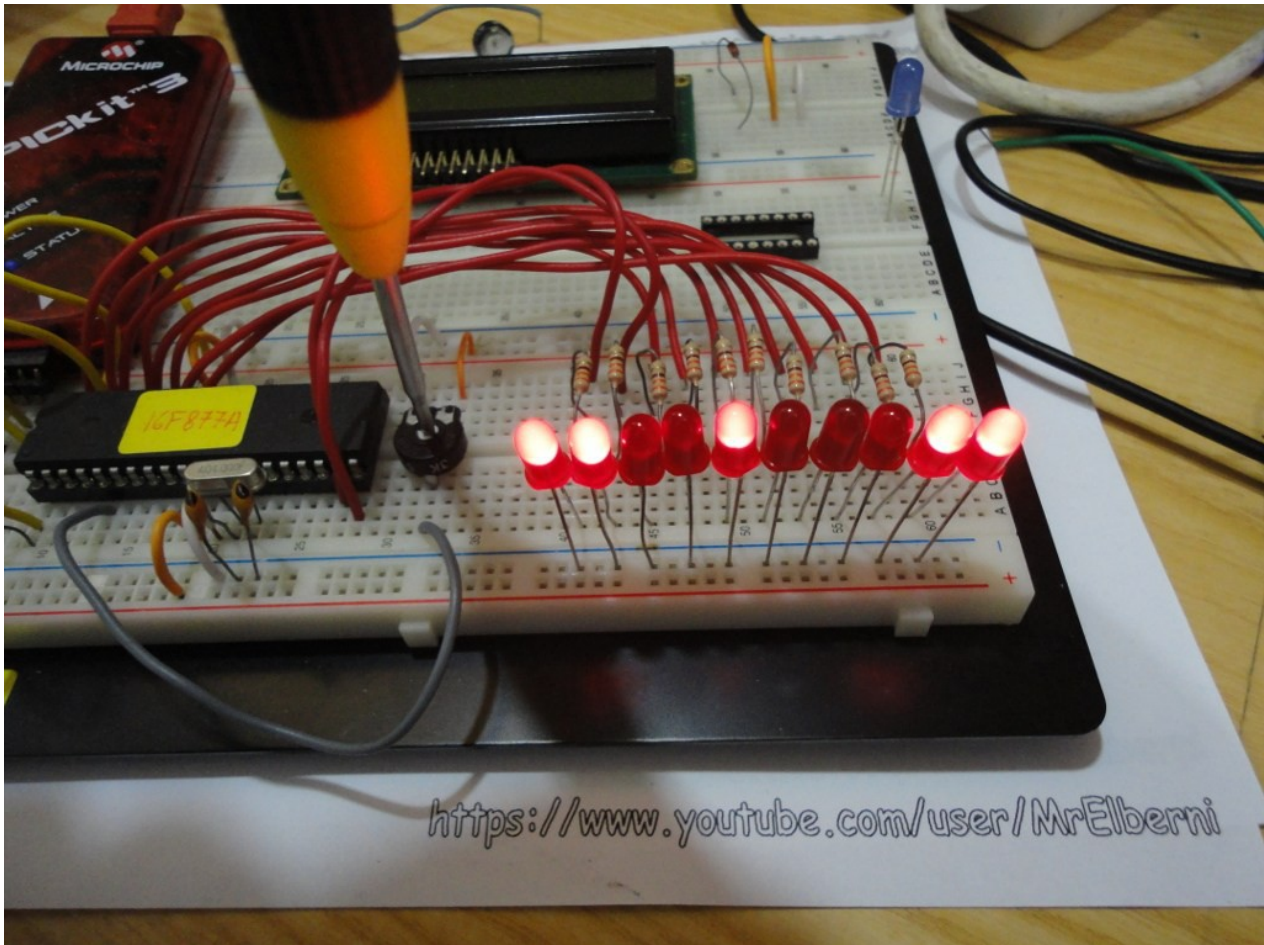
Mediante la resistencia variable R_v , a la entrada analógica `AN3` se le hace llegar una señal analógica que varia de 0V a 5V. Esta señal será convertida a digital, los valores digitales correspondientes de la tensión analógica que está llegando se verán en forma binaria a través del puerto B para los 8 bits menos significativos y através del puerto D los 2 bits mas significativos.

El código en el XC8 es el siguiente:



```
1  #pragma config FOSC = XT          // Oscillator Selection bits (XT oscillator)
2  #pragma config WDTE = OFF         // Watchdog Timer Enable bit (WDT disabled)
3  #pragma config PWRTE = ON         // Power-up Timer Enable bit (PWRT enabled)
4  #pragma config BOREN = ON         // Brown-out Reset Enable bit (BOR enabled)
5  #pragma config LVP = OFF          // Low-Voltage (Single-Supply) In-Circuit Serial Pro
6  #pragma config CPD = OFF          // Data EEPROM Memory Code Protection bit (Data EEPR
7  #pragma config WRT = OFF          // Flash Program Memory Write Enable bits (Write pro
8  #pragma config CP = OFF
9
10 #define _XTAL_FREQ 4000000
11 #include <xc.h>
12
13 void main(void) {
14     TRISB=0; //como salidas, aquí se conectarán leds para los 8 bits menos significat
15     TRISD&=~((1<<1)|(1<<0)); //como salidas, aquí se conectarán leds para los 2 bits
16     ADCON1=0b10000100; //justificación derecha tiempo de conversión fosc/2 usar AN3
17     ADCON0=0b01011000; //fosc/8 para tiempo de conversión 2us leer AN3 conversión no
18
19     while(1) {
20         ADCON0bits.ADON=1; //activa el cad
21         delay_us(30); //espera el tiempo de adquisición, esto es espera que el cor
22         ADCON0bits.GO=1; //inicia la conversión de la señal analógica almacenada en
23         while(ADCON0bits.GO==1); //espera que la conversión termine cuando ADCONbits
24         ADCON0bits.ADON=0; //Desactiva el cad
25         PORTD=ADRESH; //al puerto d los 2 bits mas significativos del valor analógico
26         PORTB=ADRESL; //al puerto d los 8 bits menos significativos del valor analógi
27         delay_ms(500); //pausa para poder ver el numero en los leds
28     }
29     return;
30 }
```

Una imagen del circuito realizado para el uso del conversor analógico digital PIC.

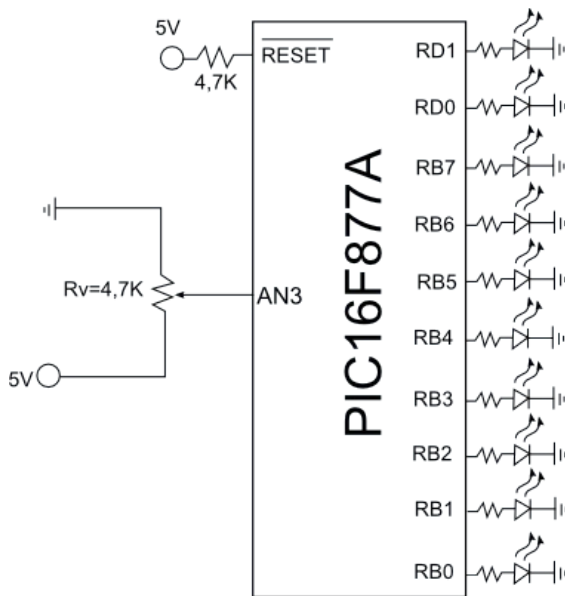


El siguiente vídeo publicado por MrElberni se ve la forma de utilizar el módulo conversor analógico digital PIC sin el uso de interrupciones:

Youtube mrelberni: [Analógico Digital CAD o ADC PIC en XC8 sin interrupciones](#)

Ejemplo 2 En el siguiente ejemplo, mediante una resistencia variable, a la entrada analógica AN3 del PIC16F877A se le hará llegar una señal analógica que va desde 0V a 5V, esa señal será convertida a digital cuyos valores en binario serán 000000000 para 0V y 111111111 para 5V los cuales serán visibles por medio de unos leds conectados a los puertos B y D, cada vez que se produzca una interrupción tras la conversión en la rutina de interrupción los 8 bits menos significativos serán visibles en el puerto B y los 2 bits mas significativos serán visibles en el puerto D, cuando todos los leds estén apagados esto es 0 se estará midiendo 0V y cuando todos los leds estén encendidos esto es 1023 se estará midiendo 5V, los valores binarios variarán de 0 a 1023 lo cual será visible por medio de los leds cada vez que se gire la resistencia variable para los diferentes valores de la señal analógica. En este caso se utilizarán interrupciones por finalización de conversión del modulo CAD.

El circuito utilizado es el siguiente:



CONVERSOR ANALÓGICO DIGITAL CAD EN EL PIC 16F877A

Mediante la resistencia variable Rv, a la entrada analógica AN3 se le hace llegar una señal analógica que varía de 0V a 5V. Esta señal será convertida a digital, los valores digitales correspondientes de la tensión analógica que está llegando se verán en forma binaria a través del puerto B para los 8 bits menos significativos y a través del puerto D los 2 bits más significativos.

El código en el XC8 es el siguiente:

```

Start Page x  cad.c x
//
1  #pragma config FOSC = XT          // Oscillator Selection bits (XT oscillator)
2  #pragma config WDTE = OFF         // Watchdog Timer Enable bit (WDT disabled)
3  #pragma config PWRTE = ON         // Power-up Timer Enable bit (PWRT enabled)
4  #pragma config BOREN = ON         // Brown-out Reset Enable bit (BOR enabled)
5  #pragma config LVP = OFF          // Low-Voltage (Single-Supply) In-Circuit Serial Pr
6  #pragma config CPD = OFF          // Data EEPROM Memory Code Protection bit (Data EEP
7  #pragma config WRT = OFF          // Flash Program Memory Write Enable bits (Write pr
8  #pragma config CP = OFF           // Flash Program Memory Code Protection bit (Code p
9
10 #define _XTAL_FREQ 4000000        //fosc=4Mhz
11 #include <xc.h>
12
13 void main(void) {
14     //pines a utilizar como salidas donde se conectarán los leds
15     TRISB=0; //como calidas ya que aqui se conectaran los leds para ver los 8 bits m
16     TRISD&=~((1<<1)|(1<<0)); //como salidas para los leds de los 2 bits mas signific
17     //para el manejo de interrupciones
18     INTCON=0b11000000; //Se habilitan el uso de las interrupciones
19     PIE1bits.ADIE=1; //El bit 6 del registro PIE1 a 1 para habilitar la interrupción
20     PIR1bits.ADIF=0; //El bit 6 del registro PIR1 a 0 para que ocurran las interrupc
21     //Control del módulo conversor analógico digital PIC
22
23     ADCON0=0b01011000; //tiempo de conversión fosc/8 leer AN3 conversión no inici
24     ADCON1=0b10000100; //justificación derecha tiempo de conversión fosc/8 usar C

```



```
25
26     while(1){
27         ADCON0bits.ADON=1; //Activa el conversor analógico digital PIC
28         delay_us(30);      //espera el tiempo de adquisición, esto es esperar que se
29         ADCON0bits.GO=1;    //Inicia la conversión de la señal analógica almacenada
30     }                      //cuando esta termine se producirá una interrupción al ha
31     return;
32 }
33
34 void interrupt CAD_int(void){ //función de interrupción por finalización del convers
35     if(PIR1bits.ADIF==1){ //ha ocurrido la interrupción por finalización de la cnver
36         ADCON0bits.ADON=0; //Desactiva el conversor analógico digital PIC
37         PORTD=ADRESH; //al puerto D se le asigna los 2 bits mas significativos del v
38         PORTB=ADRESL; //al puerto B se le asigna los 8 bits menos significativos del
39     }
40     PIR1bits.ADIF=0; //El bit 6 del registro PIR1 a 0 para que vuelvan a ocurrir mas
41 }
42
```

El siguiente vídeo publicado por MrElberni se ve la forma de utilizar el módulo conversor analógico digital PIC con el uso de interrupciones:

Youtube mrelberni: [Analógico Digital CAD o ADC PIC en XC8 con interrupciones](#)

Interrupción ADC PIC

En este apartado se comenta como utilizar la interrupción ADC PIC que se produce cuando se termina la conversión de una entrada analógica a digital.

Los pasos a seguir para este caso son los siguientes:

1. Configurar las interrupciones poniendo los bits 7 (GIE) y 6 (PEIE) del registro INTCON y el bit 6 (ADIE) del registro PIE1 a 1 y el bit 6 (ADIF) del registro PIR1 a 0;
2. Configurar que pines serán utilizados como entradas analógicas, si se quiere una tensión de referencia diferente a la tensión de trabajo del PIC, lo cual como se ha comentado líneas arriba se hace con los bits 3, 2, 1 y 0 del registro ADCON1.
3. Elegir cual será el reloj a utilizar para obtener el tiempo de conversión adecuado, que tendrá que ser mayor a 1,6µs, lo cual se hace mediante los bits 7 y 6 del registro ADCON0 junto con el bit 6 del registro ADCON1.
4. Seleccionar cual será la entrada analógica a leerse, lo cual se hace mediante los bits 5, 4 y 3 del registro ADCON0.

5. Activar el conversor analógico digital PIC poniendo a 1 el bit 0 del registro ADCON0.
6. Esperar el tiempo de adquisición necesario para que el condensador del módulo conversor analógico digital PIC se cargue en forma adecuada.
7. Poner a 1 el bit 2 del registro ADCON0 para que comience la conversión de la tensión almacenada en el condensador del conversor analógico digital PIC y se guarde este valor en forma binaria en los registros **ADRESH** y **ADRESL**.
8. Esperar a que se produzca la interrupción.
9. Leer los registros **ADRESH** y **ADRESL** que es donde se ha guardado el número binario que representa el valor de la señal analógica y poner a 0 el bit 6 (ADIF) del registro PIR1.

Interrupción ADC PIC en el XC8

Para habilitar la interrupción ADC PIC los bits 7 y 6 del registro INTCON se pondrán a 1 en el XC8 esto se hace así:

INTCON=0b11000000;//Se habilitan el uso de las interrupciones

Para habilitar la interrupción ADC PIC por la finalización de una conversión analógica digital con el modulo ADC, del registro PIE1 se pone a 1 su bit 6, lo que en el XC8 se hace así

PIE1bits.ADIE=1;//El bit 6 del registro PIE1 a 1 para habilitar la interrupción ADC PIC.

El bit 6 del registro PIR1 se pondrá a 0 para detectar que se ha producido una interrupción ADC PIC, si este bit está a 1 no se detectará la interrupción ADC PIC en el XC8 se hará así.

PIR1bits.ADIF=0;//El bit 6 del registro PIR1 a 0 para que ocurran las interrupciones.

Para el uso del módulo ADC en el XC8, se han definido los nombres de los registros a utilizar para el control del convertidor analógico digital como ADCON0 y ADCON1, los cuales se pueden configurar en forma binaria, decimal o hexadecimal.

En el caso de usar un cristal de 4Mhz, para obtener un tiempo de conversión mayor a 1,6us se utilizará un reloj que sea $F_{osc}/2$ con lo cual se logra que el tiempo de conversión sea de 2us, para lograr esto se ponen los bits 7 y 6 del registro ADCON0 a 0 y el bit 6 del registro ADCON1 también a 0.

Si se quiere elegir por ejemplo el canal 3 o la entrada analógica 3 AN3, el el registro ADCON0 se tendrá que poner su bit 5 a 0 y sus bits 4 y 3 a 1.

Para iniciar la conversión se pone el bit 2 del registro ADCON0 a 1, el cual luego de la conversión se pondrá automáticamente a 0 provocando una interrupción.

El bit 1 del registro ADCON0 no se usa por lo que se pone a 0.

Para activar el módulo conversor analógico digital pic el bit 0 del registro ADCON0 se pondrá a 1.

Para la justificación a la derecha del número binario de 10 bits que representa el valor analógico, el bit 7 del registro ADCON1 se pone a 1.

El bit 6 del registro ADCON1 trabaja junto con los bits 7 y 6 del registro ADCON0 para obtener el tiempo de conversión, que para este ejemplo será de 2 us.

Los bits 5 y 4 del registro ADCON1 no se usan por lo que se ponen a 0.

Con los bits 3, 2, 1 y 0 del registro ADCON1 de acuerdo a la tabla vista líneas arriba, se elige alguna de las combinaciones que incluyan el uso de la entrada analógica 3 AN3 por ejemplo 0100, con lo cual también se indica que se usarán las entradas AN1 y AN0, en este caso no hay una opción para utilizar solo la entrada AN3, cosa que en otros microcontroladores PIC si es posible.

Luego se tendrá:

ADCON0=0b00011101; //fosc/2 AN3 conversión iniciada ADC activado.

ADCON1=0b10000100; //justificación derecha fosc/2 AN3

Si se necesita manipular solo uno de los bits de estos registros, por ejemplo para activar el ADC será así:

ADCON0bits.ADON=1; //ADC activado

y para desactivarlo sería algo así:

ADCON0bits.ADON=0; //ADC desactivado

Para iniciar la conversión sería algo así:

ADCON0bits.GO=1; //Iniciar la conversión, este bit se pone automáticamente a 0 cuando la conversión termina, a la vez que se producirá una interrupción.

Y así se pueden manipular en forma independiente cada uno de los bits de estos registros, además el MPLABX tiene una ayuda que hace aparecer estos manipuladores y poder seleccionar el que se desee utilizar.

Para guardar el número de 10 bits que representará el valor analógico, los nombres de los registros que almacenan este valor en el XC8 se han definido como ADRESH y ADRESL de 8 bits cada uno, dependiendo si la justificación se ha elegido a la derecha o a la izquierda el número se guardará de diferente manera, por ejemplo si la justificación es a la derecha, eso indica que los 2 bits más significativos del número estarán en los bits 1 y 0 del registro ADRESH, mientras que los 8 bits menos significativos del número estarán en el registro ADRESL, luego para la lectura de este número se procede como se indica a continuación.

Se crea una variable entera que en el XC8 estas variables son de 16 bits y se le asigna el valor del registro ADRESH

int numero_adc=ADRESH; // en la variable entera numero_adc se guardan los 2 bits más significativos del número que representa el valor analógico, estos se ubicarán en los

bits 1 y 0 de la variable `numero_adc`, luego hay que hacer un corrimiento de 8 bits hacia la izquierda en la variable `numero_adc` para que estos bits se ubiquen en los bits 10 y 9 de la variable `numero_adc`, sería algo así

`numero_cad=numero_adc<<8;` //Ahora los 2 bits mas significativos del número binario que representa el valor analógico están ubicados en los bits 10 y 9 de la variable `numero_adc`.

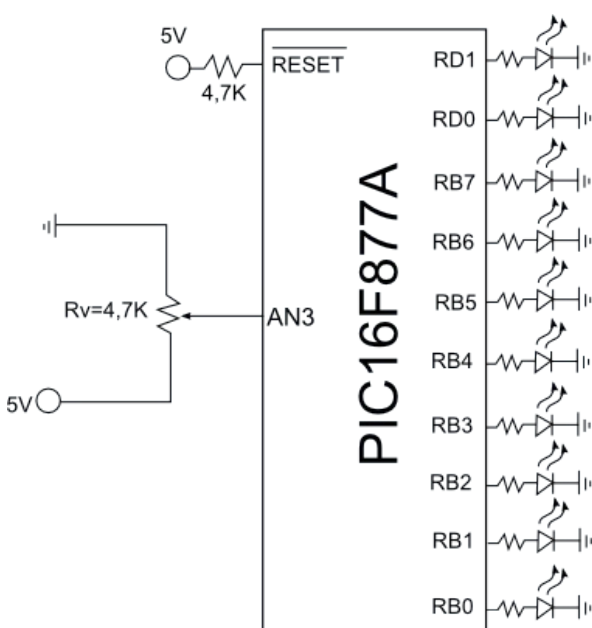
Ahora a la variable `numero_adc` se le sumaran los 8 bits menos significativos del número binario que representa el valor analógico y que está almacenado en el registro `ADRESL` como se indica

`numero_cad=numero_adc+ADRESL;` //de esta manera el número binario de 10 bits que representa el valor analógico queda almacenado o guardado dentro de la variable `numero_adc` la cual puede utilizarse al gusto según las necesidades que se tenga de este número.

Ejemplo interrupción ADC PIC

En el siguiente ejemplo, mediante una resistencia variable, a la entrada analógica AN3 del PIC16F877A se le hará llegar una señal analógica que va desde 0V a 5V, esa señal será convertida a digital cuyos valores en binario serán 0000000000 para 0V y 1111111111 para 5V los cuales serán visibles por medio de unos leds conectados a los puertos B y D, cada vez que se produzca una interrupción ADC PIC, tras la conversión en la rutina de interrupción ADC PIC los 8 bits menos significativos serán visibles en el puerto B y los 2 bits mas significativos serán visibles en el puerto D, cuando todos los leds estén apagados esto es 0 se estará midiendo 0V y cuando todos los leds estén encendidos esto es 1023 se estará midiendo 5V, los valores binarios variarán de 0 a 1023 lo cual será visible por medio de los leds cada vez que se gire la resistencia variable para los diferentes valores de la señal analógica. En este caso se utilizará la interrupción ADC PIC por finalización de conversión del módulo ADC.

El circuito utilizado es el siguiente:



CONVERSOR ANALÓGICO DIGITAL CAD EN EL PIC 16F877A

Mediante la resistencia variable R_v , a la entrada analógica AN3 se le hace llegar una señal analógica que varía de 0V a 5V. Esta señal será convertida a digital, los valores digitales correspondientes de la tensión analógica que está llegando se verán en forma binaria a través del puerto B para los 8 bits menos significativos y a través del puerto D los 2 bits mas significativos.

El código en el XC8 es el siguiente:

```
//////*****  
//***microcontroladores-mrelberni.com***//  
//////*****  
//////**** Interrupción ADC PIC ****//  
//////*****  
  
//bits de configuración del PIC16F877A  
#pragma config FOSC = XT      // Oscillator Selection bits (XT oscillator)  
#pragma config WDTE = OFF     // Watchdog Timer Enable bit (WDT disabled)  
#pragma config PWRTE = ON     // Power-up Timer Enable bit (PWRT enabled)  
#pragma config BOREN = OFF    // Brown-out Reset Enable bit (BOR disabled)  
#pragma config LVP = OFF     // Low-Voltage (Single-Supply) In-Circuit  
#pragma config CPD = OFF     // Data EEPROM Memory Code Protection bit  
#pragma config WRT = OFF     // Flash Program Memory Write Enable bits  
#pragma config CP = OFF      // Flash Program Memory Code Protection bit  
  
#define _XTAL_FREQ 4000000  
#include <xc.h>  
  
void main(void) {  
    //pines a utilizar como salidas donde se conectarán los leds  
    TRISB=0;//como salida, aquí se conectarán los leds para los bits menos significativos  
    TRISD&=~((1<<1)|(1<<0));//como salidas para los leds de los bits mas significativos  
  
    //para el manejo de las interrupciones  
    INTCON=0b11000000;//se habilitan el uso de las interrupciones  
    PIE1bits.ADIE=1;//habilita la interrupción ADC PIC  
    PIR1bits.ADIF=0;//para detectar la interrupción ADC PIC  
  
    //control del módulo convertidor analógico digital ADC  
    ADCON0=0b01011000;//tiempo de conversión fosc/8, leer AN3 conversión no iniciada  
        //ADC desactivado  
    ADCON1=0b10000100;//justificación derecha, tiempo de conversión fosc/8 usar AN3  
  
    while(1){//ciclo del programa  
        ADCON0bits.ADON=1;//activa el módulo ADC  
        __delay_us(30);//tiempo de adquisición, esto es espera que el condensador se cargue  
        ADCON0bits.GO=1;//inicia la conversión de la señal analógica almacenada en el condensador  
        //cuando esta termine se producirá una interrupción ADC PIC  
    }  
    return;  
}  
  
void interrupt ADC_int(void){//rutina de atención a las interrupciones  
    if(PIR1bits.ADIF==1){//ha ocurrido la interrupción por finalización de la conversión  
        ADCON0bits.ADON=0;//desactiva el ADC  
        PORTD=ADRESH;//al puerto D se le asigna los 2 bits mas significativos  
        PORTB=ADRESL;//al puerto B se le asignan los 8 bits menos significativos  
  
        PIR1bits.ADIF=0;//Para que se vuelvan a detectar interrupción ADC PIC  
    }  
}
```


<http://microcontroladores-mrelberni.com/>