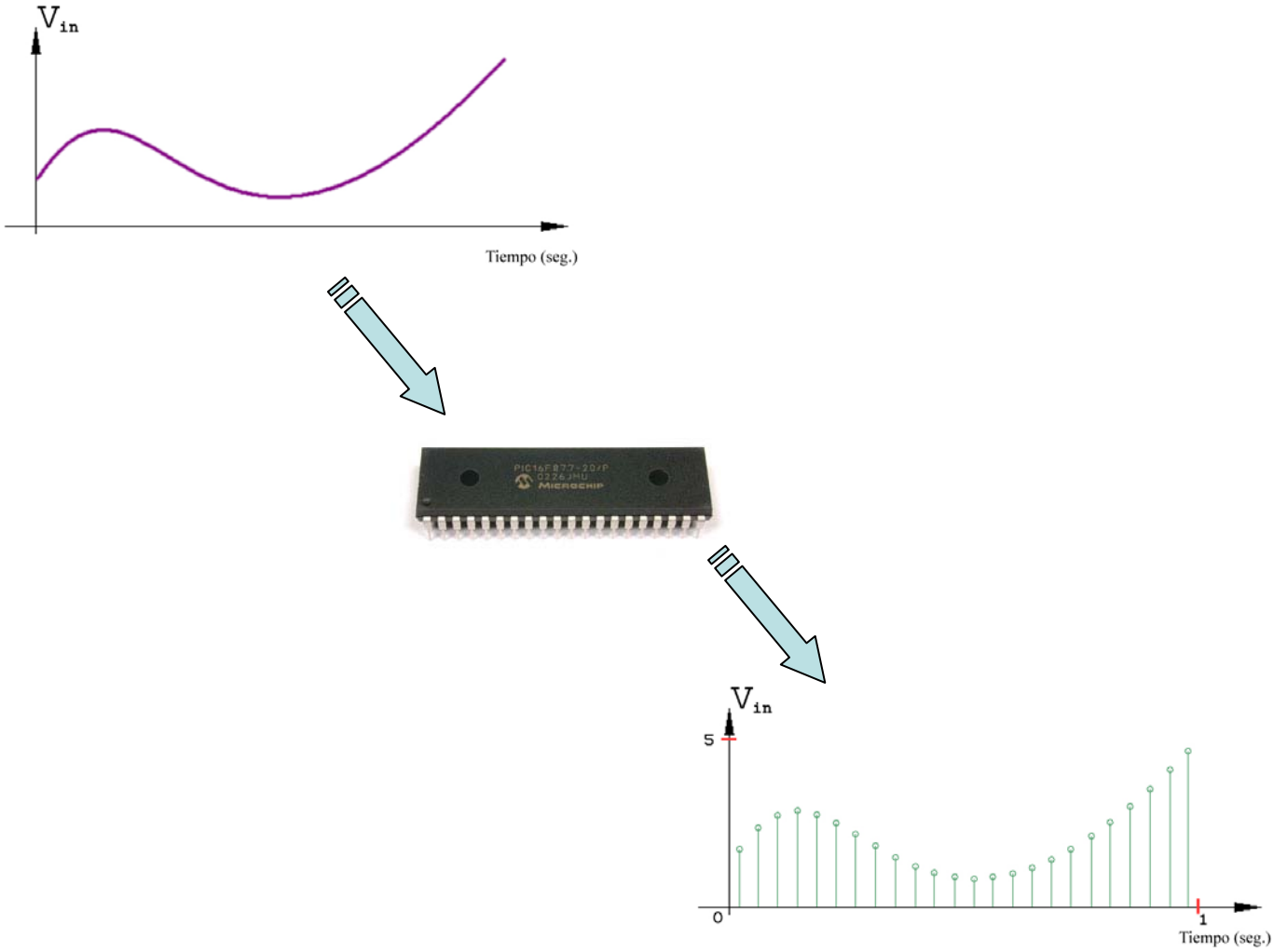
CONVERSOR ANALÓGICO DIGITAL DEL PIC16F877



GRUPO A02-A03

Proyecto PAEEES 04/993. U.P.V.
Escuela Politécnica Superior de Alcoy
Marzo 2005

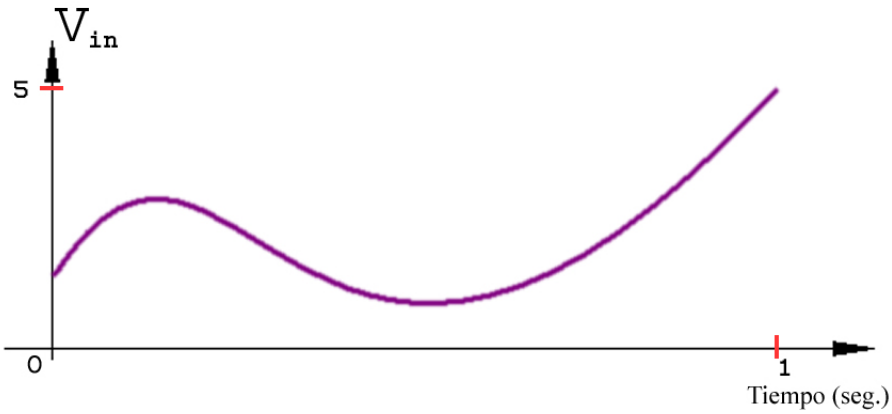
Cantero Siñuela, Iván Saúl
Gil Hernández, Diego
Ponsoda Hernández, Iván
Richart Sanchis, Lucía
Sala Gisbert, Héctor
Seguí Richart, Santiago

INDICE

1. INTRODUCCIÓN

El microcontrolador PIC16F877 de Microchip puede desempeñar muchas funciones pero en la que nos vamos a centrar aquí es la de su conversor analógico-digital. Antes de meternos de pleno en su funcionamiento vamos a comentar los conceptos básicos de una conversión de una señal analógica a digital.

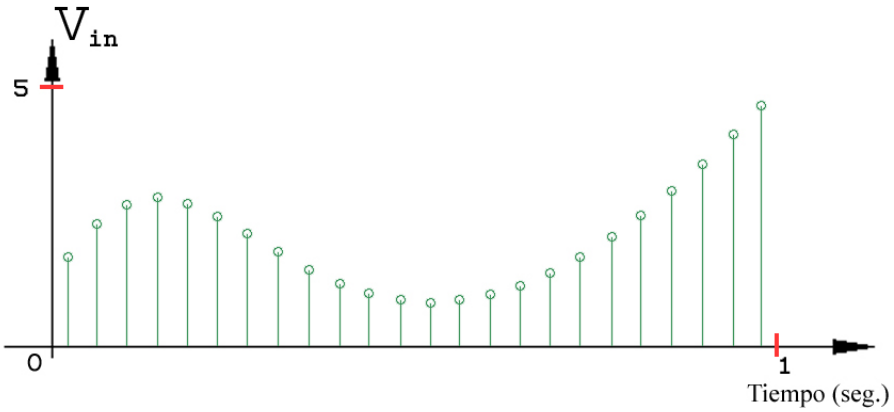
Tanto nuestra voz como muchas de las señales que se envían a través de un medio guiado como un cable o no guiado como es el aire son de tipo continuo y pueden tomar infinitos valores a lo largo del tiempo. Por ejemplo podemos decir que la señal eléctrica que se transmite de la tarjeta de sonido al altavoz es continua y puede tomar cualquier tensión entre los dos hilos.



El interés en digitalizar una señal puede surgir por varios motivos: el hecho de querer almacenarla en un soporte digital o transmitirla digitalmente para poder reconstruirla, poder tratar con programas los valores analógicos que dé un sensor, etc.

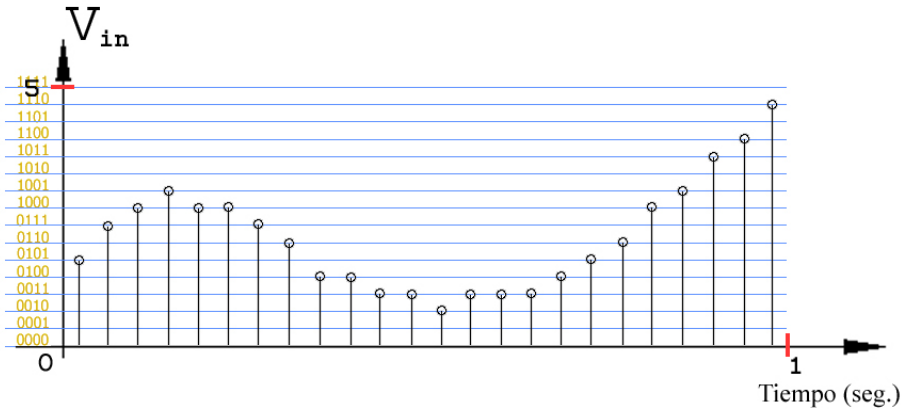
Acotando la señal en un intervalo de tiempo y unos valores mínimos y máximos de tensión tenemos que tener en cuenta dos factores fundamentales a la hora de almacenar dicha señal en un formato adecuado que pueda almacenarse digitalmente (con ceros y unos): se tiene que muestrear y cuantificar.

El muestreo implica que tenemos que coger una muestra de la señal cada T segundos ya que no hay memoria suficiente capaz de almacenar los infinitos puntos de una señal en un intervalo cualquiera de tiempo. En el ejemplo de las figuras se ha acotado un segundo de tiempo y 5 V de tensión de entrada analógica del PIC.



En la figura se han tomado 24 muestras en un segundo, esto es, la frecuencia de muestreo.

La cuantificación surge por el mismo motivo que el muestreo pero para el eje de ordenadas: una vez tenemos una muestra su amplitud puede tomar infinitos valores, debemos redondear entre unos valores fijos a lo largo de ese eje. Estos valores van a depender del número de bits que vayamos a almacenar para cada muestra, por ejemplo, en la imagen se cogen 4 bits y con ellos se pueden formar 16 combinaciones y por lo tanto 16 distintos niveles en los que se puede dividir el eje. El PIC cuantifica con 10 bits luego son 1023 niveles.

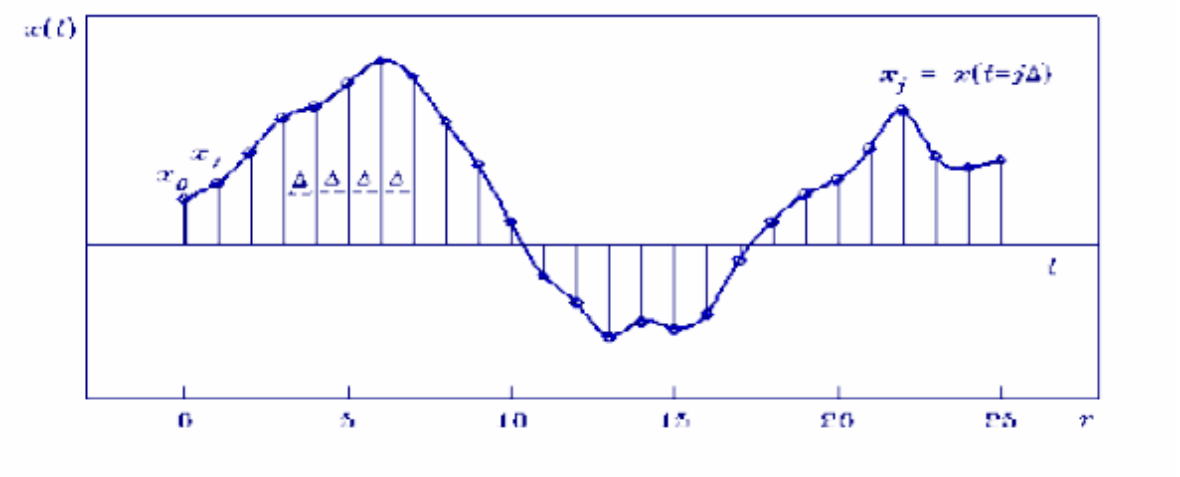


Tanto en un proceso como en el otro, se va a provocar una pérdida de la forma de la señal original y eso implica que se va a cometer un error de aproximación cuya magnitud se puede calcular y se comenta con más detalle en el siguiente apartado.

2. ERROR DE CUANTIFICACIÓN

2.1. MUESTREO

El parámetro fundamental del muestreo digital es el *intervalo de muestreo* ▲seg., o su equivalente *frecuencia de muestreo 1/*▲ Hz. Lógicamente, cuanto menor sea ▲, mayor número de valores obtendremos de la señal, y viceversa. El resultado de dicho muestreo es la obtención de una serie discreta ordenada {*xr*} = {*x*0, *x*1, *x*3,..., *xr*,...}, en la que el índice *r* indica la posición de orden temporal del valor *xr*. Así, el valor de la señal original, en el tiempo t = ▲*r*, x (t), se representa por *xr*.



Si aumentamos el número de muestras por unidad de tiempo, la señal muestreada se parecerá más a la señal continua. Respecto a esto, el criterio de Nyquist asegura que para que la señal muestreada contenga la misma información que la continua, la separación mínima entre dos instantes de muestreo debe ser 1/(2 W), siendo W el ancho de banda de la señal. Dicho de otra forma, que la frecuencia de muestreo debe ser mayor o igual que 2 W.

2.2. CUANTIFICACIÓN

Se conoce como error de cuantificación (o *ruido*), a la diferencia entre la señal de entrada (sin cuantificar) y la señal de salida (ya cuantificada), interesa que el ruido sea lo más bajo posible.

Ej. Si tenemos una señal cualquiera, y la queremos guardar en un archivo de 4 bits, tendremos 2^4 niveles para cuantificarla, (8 para valores positivos, y 8 mas para los negativos).

▲ = *(1/2^{N-1})= Intervalo de cuantificación = 1/2^3 = 0.125*

Error: -▲/2 < *error* >▲/2

Ej. A partir de la potencia de una señal, podemos saber que error estamos cometiendo.

Potencia: $<error^2>= \blacktriangle^2/12$

Si tenemos una tarjeta de sonido, y trabaja a 16 bits por muestra:

Cometerá un error de: $(1/2^{15})^2/12 = 1/(2^{30} * 12)$

2.3. TÉCNICAS DE CUANTIFICACIÓN:

Como información complementaria, decir que existen diferentes tipos de cuantificación, cada uno de las cuales se amoldara a los datos que estemos enviando.

- Cuantificación uniforme**:** la distancia entre los niveles de reconstrucción es siempre la misma. No hacen ninguna suposición acerca de la naturaleza de la señal a cuantificar, de ahí que no proporcionen los mejores resultados. Sin embargo, tienen como ventaja que son los más fáciles y menos costosos de implementar.

- Cuantificación logarítmica**:** incrementa la distancia entre los niveles de reconstrucción conforme aumenta la amplitud de la señal. Muy usada en señales de voz.

- Cuantificación no uniforme**:** si conocemos la función de la distribución de probabilidad, podemos ajustar los niveles de reconstrucción a la distribución de forma que se minimice el error cuadrático medio. Esto significa que la mayoría de los niveles de reconstrucción se den en la vecindad de las entradas más frecuentes y, consecuentemente, se minimice el error (ruido).

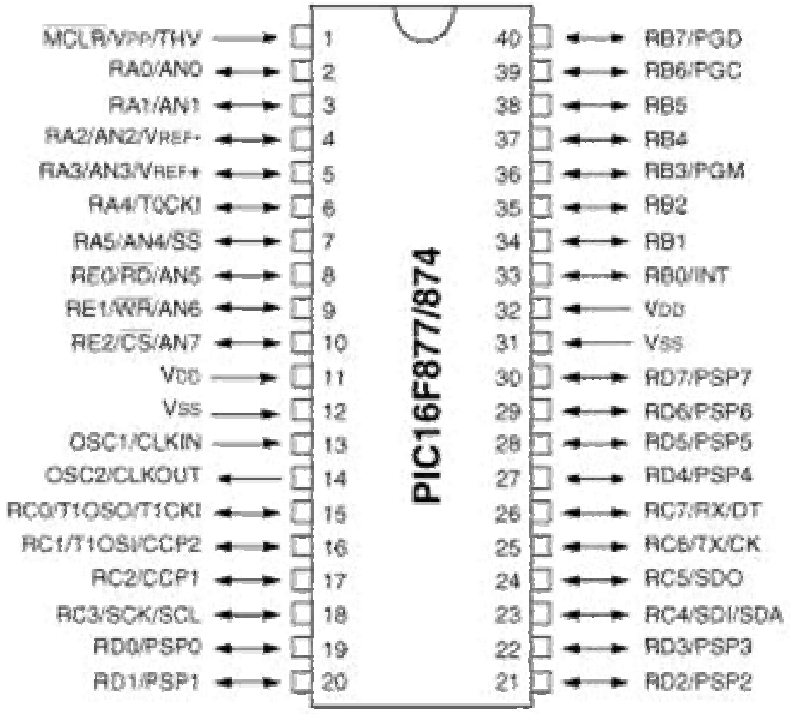
- Cuantificación vectorial**:** se basa en cuantificar según las muestras vecinas, resulta más eficiente cuantificar los datos en bloques de N muestras.

- Cuantificadores de Vecino Más Cercano (Voronoi)**:** el proceso de codificación no necesita almacenar una descripción geométrica de las celdas, la codificación es mediante una comparación de distancias.

3. REGISTROS DEL MODULO DE A/D.

El módulo de A/D tiene cuatro registros. Estos registros son:

- **ADRESH** : Parte alta del resultado de la conversión
- **ADRESL**: Parte baja del resultado de la conversión
- **ADCON0**: Registro de Control 0 ;control del funcionamiento del conversor
- **ADCON1**, Registro de Control 1; configuración de los pines del puerto



3.1.REGISTRO ADCON0 (DIRECCIÓN LFH)

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/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6:**ADCS1:ADCS0**: En estos dos bits se hace la selección de la frecuencia de reloj para el Convertidor A/D.

00 Fosc/2
01 Fosc /8
10 Fosc/32
11 F_{RC} (Procede del oscilador RC interno)

bit 5-3:**CH2:CH0:** Aquí se selecciona el canal analógico por donde entrará la señal a digitalizar. En este microcontrolador tenemos 8 canales de entrada al Conversor A/D
000 = Canal 0, (RA0/AN0)
001 = Canal 1, (RA1/AN1)
010 = Canal 2, (RA2/AN2)
011 = Canal 3, (RA3/AN3)
100 = Canal 4, (RA4/AN4)
101 = Canal 5, (RA5/AN5)
110 = Canal 6, (RA6/AN6)
111 = Canal 7, (RA7/AN7)

bit 2: **GO/#DONE**. bit de estado de la conversión A/D
Si ADON=1
1= La conversión A/D está en marcha (mientras está a 1 se está realizando la conversión)
0 = La conversión ha finalizado. (el bit se pone a cero automáticamente por hardware cuando la conversión A/D finaliza) el resultado de la conversión aparece en ADRESH:ADRESL

bit 1: **No implementado**: Se lee como "0"

bit 0: **ADON**: bit de puesta en marcha
1 = El convertidor A/D está operativo
0 = El convertidor A/D está apagado y no consume corriente.

3.2. EL REGISTRO ADCON1

El registro ADCON1 es uno de los registros del convertidor A/D del PIC16F877, se trata de un registro de configuración de los pines del puerto, este registro se compone de 8 bits, los cuales describamos su función a continuación:

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

Bit 7: **ADFM**: Selecciona el formato del resultado de la conversión A/D

1 = >Pone en el registro **ARDESH** los seis bits de mayor peso a "0"
0 = >Pone los 6 bits de menor peso del registro **ADRESL** a "0"

Bits 6-4: No implementados: Se leen como cero

Bit 3-0: **PCFG3:PCFG0**: bits de configuración de los canales de entrada del convertidor A/D. Se utilizan para configurar las patillas como E/S digital o como entrada analógica de acuerdo con la siguiente tabla:

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs(2)
0000	A	A	A	A	A	A	A	A	VDD	Vss	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	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	RA3	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	RA3	Vss	2/1
011x	D	D	D	D	D	D	D	D	VDD	Vss	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	Vss	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	Vss	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	Vss	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

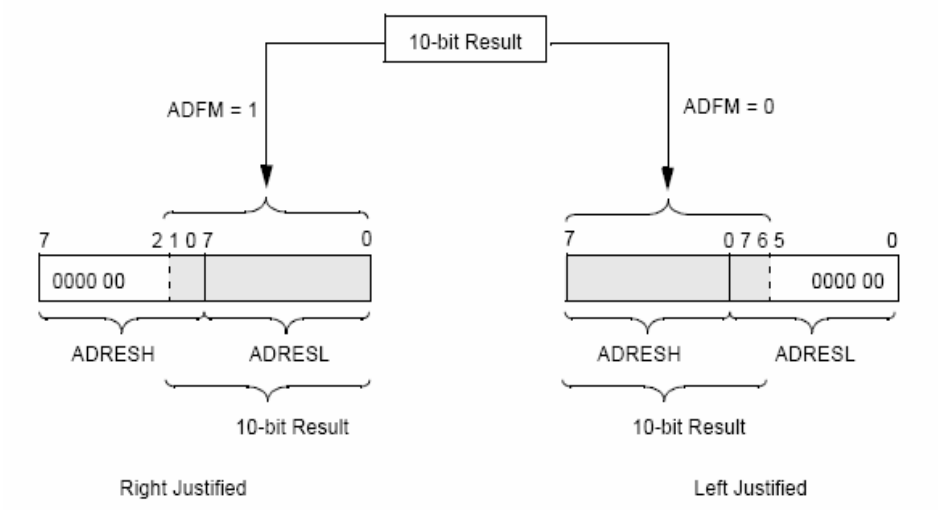
A= Entrada Analógica; D = E/S Digital

3.3. LOS REGISTROS ADRESH Y ADRESL

El par de registros **ADRESH:ADRESL** se carga con el resultado de 10 bits de la conversión A/D. Este par de registros se extienden hasta 16 bits. El módulo A/D tiene la posibilidad de justificar el resultado de 10 bits dentro de los 16 bits de la pareja de registros. La selección del formato de justificación a la izquierda o derecha se realiza con el bit **ADFM** (**ADCON1**). Los bits restantes (a los 10 de la conversión) se llenan con ceros.

Estos dos registros cuando el convertidor A/D está en OFF y no se utiliza, pueden utilizarse como dos registros de 8 bits de propósito general.

Cuando se completa la conversión A/D, el resultado se guarda en los registros y se pone a cero el bit **GO/DONE**



Por lo tanto, los 16 bits que forman el registro **ARDESH-ARDESL** con **ADFM**=1 tiene los 6 bits de mayor peso a cero y con **ADFM**=0 los 6 bit de menor peso están a cero, en los 10 bits restantes se almacena el resultado de la conversión.

4. TEMPORIZACIÓN

Para introducirnos vamos a llamar a 'Tad' como el tiempo de conversión por bit.
En la figura 3.1 tenemos un esquema de lo que seria el proceso medido en tiempo para estar seguros de que se ha realizado la conversión.

Este comienza a funcionar en el tiempo de adquisición cuando activamos el bit GO/#DONE(ADCON0). El tiempo de adquisición es el tiempo que el modulo A/D esta conectado a un voltaje externo.

Pero tenemos que tener en cuenta que para una nueva conversión podemos preparar la configuración del módulo conversor A/D y si se desea realizar una interrupción del modulo conversor A/D, se exige un mínimo de $2*T_{AD}$ para realizar una nueva conversión.

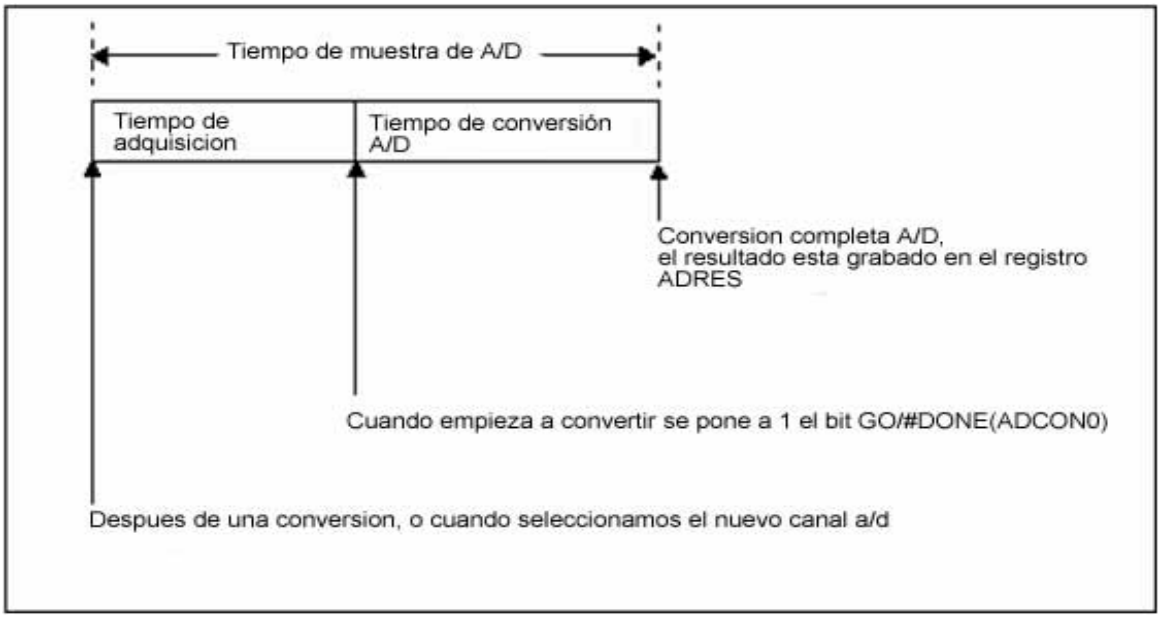


Figura 3.1

En la figura 3.2 tenemos una ecuación que nos resolverá el tiempo de adquisición asumiendo un error explicado anteriormente, para 1024 pasos del convertidor A/D. Estos cálculos han sido basados en el esquema de uso de la figura 3.21. TACQ seria el tiempo de respuesta del amplificador, TC seria el tiempo de carga del condensador (figura 3.22) que guarda el dato y TCOFF seria el coeficiente de temperatura (que este solo se utilizaría para temperaturas > 25º).

TACQ = Amplifier Settling Time + Holding Capacitor Charging Time Amplifier Settling Time Holding Capacitor Charging Time +Temperature Coefficient=

$$= TAMP + TC + TCOFF \approx 19{,}72\mu s$$

Figura 3.2

CHOLD	=	120 pF
Rs	=	10 kΩ
Conversion Error	≤	1/2 LSb
VDD	=	5V → Rss = 7 kΩ
Temperature	=	50°C (system max.)
VHOLD	=	0V @ time = 0

Figura 3.21

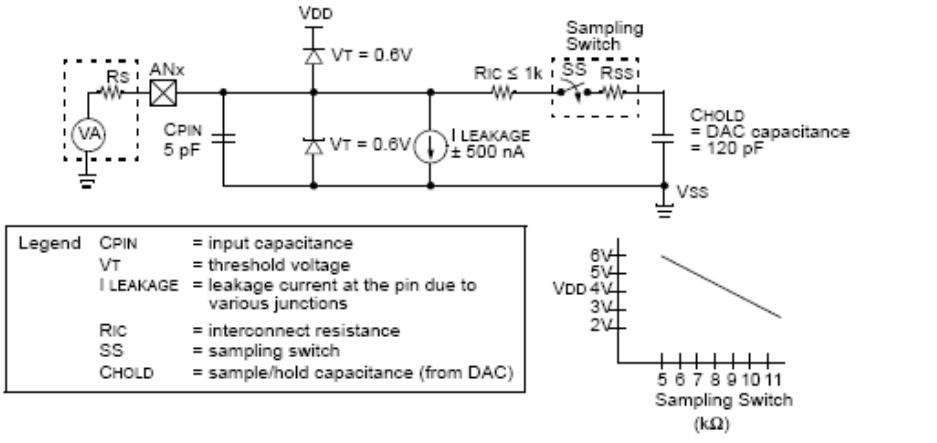


Figura 3.22

4.1. SELECCIÓN DEL RELOJ DEL CONVERTIDOR A/D

El convertidor A/D requiere un mínimo de 12 T_{AD} para la conversión de los 10 bits, La señal de reloj para la conversión A/D se selecciona por software mediante los bits **ADCS1:ADCS0**

- 2 T_{OSC}
- 8 T_{OSC}
- 32 T_{OSC}
- Oscilador interno RC (2-6 μs)

Para realizar conversiones correctas el reloj del convertidor A/D debe seleccionarse para asegurar un tiempo mínimo de T_{AD} de 1,6 mS. La figura 3.3 siguiente muestra los tiempos de T_{AD} dependiendo de la señal de reloj del micro.

Fuente del Reloj A/D		**Frecuencia máxima del dispositivo**
Operación	**ADCS1:ADCS0**	**Máx**
$F_{OSC}/2$	0 0	1.25 MHz
$F_{OSC}/8$	0 1	5 MHz
$F_{OSC}/32$	1 0	20 MHz
RC (1,2,3)	1 1	Nota 1

Nota.-

1. Si la fuente es el oscilador RC tiene un T_{AD} típico de 4 μs, pero puede variar entre 2 –6 μs.
2. Cuando la frecuencia del dispositivo es mayor de 1MHz, la fuente del oscilador RC para la conversión A/D se recomienda solo si se trabaja en el modo Sleep.
3. Para dispositivos de tensión mayores (LC), comprobar las características eléctricas

Sugerencia:

Una cosa a tener en cuenta, seria mantener las frecuencias máximas que se indican en la tabla, ya que, si por ejemplo utilizamos una frecuencia de 10 MHz entonces tendríamos que dividir por el mismo numero que si utilizamos una frecuencia de 20 MHz, por lo que estaríamos haciendo que el conversor trabajara mas despacio de lo que podría trabajar.

4.2. TIEMPOS DE FUNCIONAMIENTO

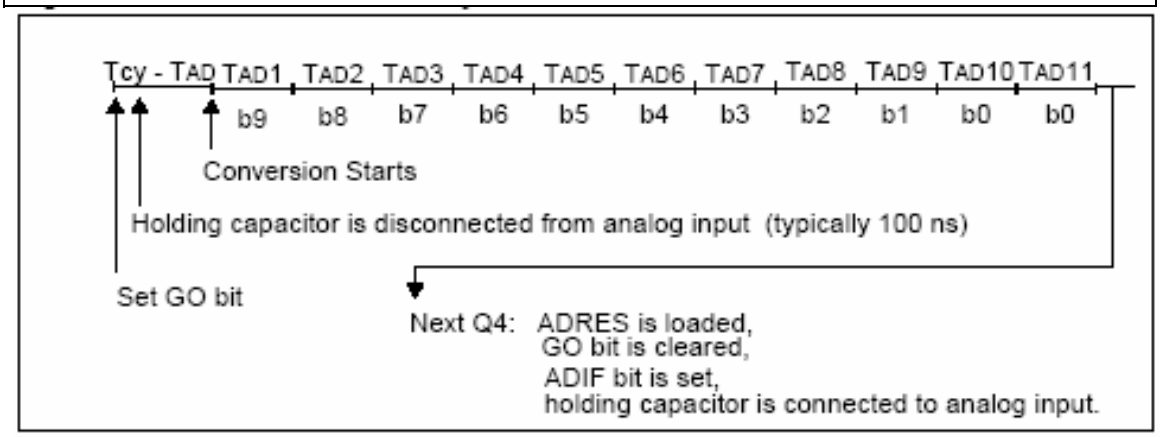
Si se pone a cero el bit **GO/#DONE** durante la conversión, se aborta la conversión actual.

El par de registros no se modificarán parcialmente con los bits que se hayan completado hasta el momento. Es decir, los registros **ADRESH:ADRESL** seguirán conteniendo el valor de la última conversión completa (o el último valor que se haya escrito en **ADRESH:ADRESL**) después de abortar la conversión A/D, es requerido el T_{AD} de espera para realizar la próxima adquisición comience. Después de 2 T_{AD} de espera, la adquisición en cauce se comienza automáticamente.

En la Figura 3.3, después de poner el bit **GO** a uno, la primera vez el segmento tiene un T_{CY} mínimo y un T_{AD} máximo.

Nota:

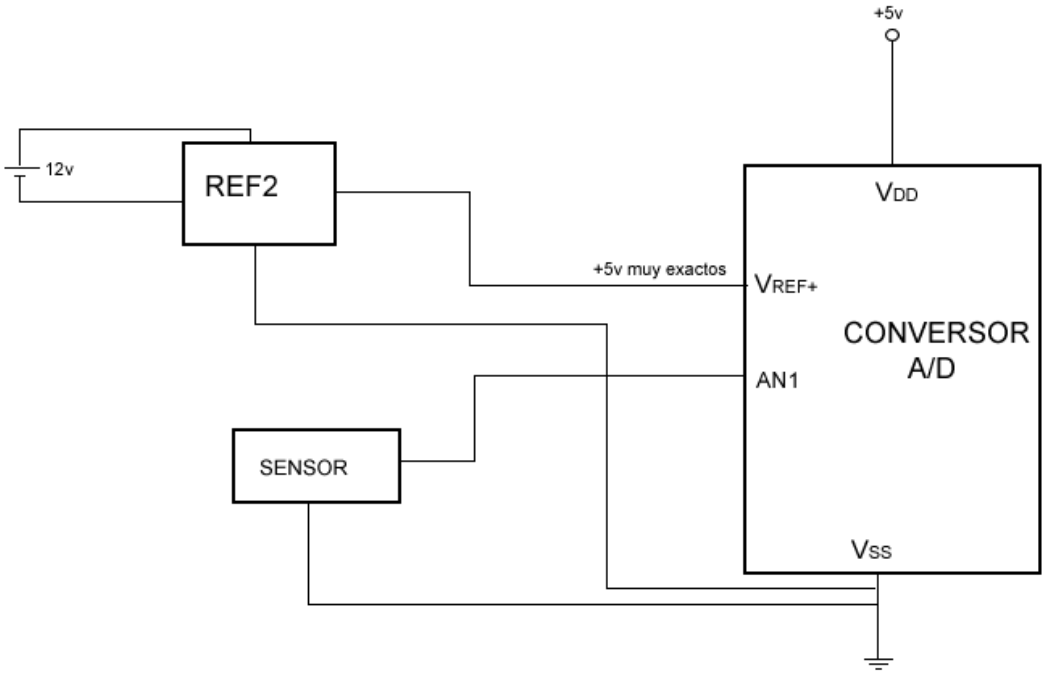
El bit **GO/#DONE** no debe ponerse a uno en la misma instrucción que se pone en **ON** el convertidor A/D.



5. EJEMPLO DE CONVERSIÓN UTILIZANDO REF02

El conversor A/D necesita 2 tensiones de referencia para su funcionamiento, Vref+ y Vref- .Dichas tensiones serán proporcionadas por el circuito REF02, ya que este nos da tensiones muy exactas. En este ejemplo hemos decidido que no vamos a utilizar Vref-, entonces conectamos los 0V de REF02 directamente a tierra, lo más cerca posible a Vss para evitar ruidos. (El sensor también lo pondremos lo más cerca posible de Vss)

El conversor lo utilizamos para comparar los niveles de tensión con la salida analógica del sensor y así poder muestrearla y cuantificarla. La tensión proporcionada por el REF02 la conectamos al conversor A/D por la patilla AN3.



5.1PROGRAMA EN C, DE CONFIGURACION DEL CONVERSOR A/D

Hemos estructurado el programa en cuatro funciones:

- Inicia_AD(): Esta función nos inicializa los registros ADCON0 y ADCON1.En ADCON0 definimos una frecuencia de Fosc/32 , ya que vamos a utilizar una frecuencia de 20Mhz., el canal por defecto será el 0 y ponemos el conversor en marcha. En ADCON1 definimos que el bit de mayor peso sea ADRESSH y decimos que dos entradas sean analógicas.

- Selecciona _A/D():Esta función sirve para seleccionar el canal por donde queramos hacer la conversión (hay 8 canales posibles), en este caso hemos elegido el canal 1 que es donde hemos conectado el sensor.

- Arranca_A/D(): Esta función pone el bit GO/DONE en 1 para que comience a convertir.

- Leer_A/D: Esta función me devuelve el resultado de la conversión, en este caso me lo devuelve en la variable resultado

```
#include<16f877.h>
#include<reg.h>
#fuses HS, NOWDT,NOPROTECT,NOPUT,NOBROWNOUT,NOLVP
#use delay(clock=20000000)

void inicia_AD(void);
void selecciona_AD(unsigned num);
void arranca_AD(void);
void leer_AD(void);


main()
  {
        unsigned canal;
        unsigned long resultado;

       inicia_AD();                    //llamamos a la funcion inicia_AD
       while(1)
       {
              canal=0b00000001;          //indicamos el canal de entrada
              selecciona_AD(canal);      //llamamos a la función selecciona_AD dandole como
                                         parametro el canal
              arranca_AD();              //llamamos a dicha función
              resultado=leer_AD();       //leer_AD nos devuelve el valor de la conversión y  lo
                                         guarda en resultado
       }
}
```

```
void inicia_AD(void)
   {
        ADCON0=0b10000001;          //Fosc/32 ; channel 0
        ADCON1=0b10000101;          // 4 ultimos bits dependiendo de la tabla 1
   }


void selecciona_AD(unsigned num)
   {
        char aux;
        aux=ADCON0 & 0b11000111;          //pongo a 0 el canal de conversion
        ADCON0=aux | (num<<3);            //activamos el canal que se pasa
   }


void arranca_AD(void)
   {
   ADGO=1;                          //ponemos en marcha el conversor
   }


void leer_AD(void)
   {
   char alto,bajo;
   long total;

   while (ADGO);                    //aquí esperamos  a que acabe de convertir
   bajo= ADRESL;
   alto = ADRESH;
   total= (long)alto<<8 | bajo;     //Hacemos una or de bajo y de alto convertido a long  y
                                    desplazado 8 bits.

   return(total);

 }
```

tabla 1:tabla de configuraciones de la entrada del A/D.

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs(2)
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	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	RA3	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	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog input D = Digital I/O

6.-BILBIOGRAFÍA

- PIC16F877 Datasheet. MICROCHIP.

- Apuntes de Teoría de la Comunicación. CAMACHO GARCÍA, Andrés.