Apunte de microcontroladores

Módulos de entrada/salida, temporización y conversión Analógica/digital de PIC 16F84A y PIC 16F877

Introducción a los PIC

Introducción

Este documento es una introducción a los microcontroladores PIC de Microchip. En particular está enfocado en los modelos PIC 16F84A y PIC 16F877 pero los conceptos y principios de funcionamiento se aplican, en general, a todos los microcontroladores PIC y pueden extenderse a otros microcontroladores de otras marcas adaptándose a su implementación en particular.

El objetivo que persigue este documento es ser una guía práctica para la implementación de algunos conceptos básicos sobre microcontroladores en lenguaje C, para principiantes. La descripción del hardware que se encuentra en esta guía es parcial en función del objetivo planteado. Los microcontroladores que se utilizan como referenciatienenmás dispositivos de los que se describen en detalle. Solo se hace énfasis en los dispositivos de entrada/salida, temporización, conversión analógica/digital y uso básico de interrupción.

Arquitectura

A diferencia de los microprocesadores tradicionales que se basan en la arquitectura Von Neumann, los microcontroladores PIC de Microchip se basan en la arquitectura Harvard. Esta dispone de bloques separados de memoria de datos y programas, lo que simplifica el diseño y permite el acceso simultáneo.

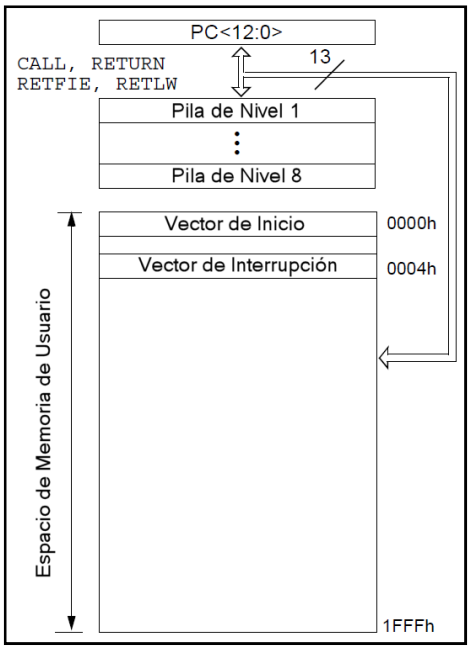
Son microcontroladores con un conjunto de instrucciones reducido (RISC), ortogonal y de un operando (operando y acumulador). Cuentan con alrededor de unas 35 instrucciones y en general no tienen soporte para multiplicación y división, salvo en las líneas de altas prestaciones.

Todos los microcontroladores poseen un cauce segmentado (pipe-line) de cuatro etapas: *fetch, decode, execute, write-back*. Todas las instrucciones son ejecutadas en un ciclo de instrucción (cuatro ciclos de reloj) salvo las instrucciones de salto que requieren 2 ciclos.

La memoria de datos está basada en bancos de registros, lo que implica que los dispositivos del sistema (temporizadores, puertos de entrada/salida, posiciones de memoria, etc.) están implementados físicamente como registros.

Organización de la memoria

La organización de memoria de los microcontroladores se divide en tres diferentes bloques. Dos de los bloques son la memoria de programa (normalmente flash) y la memoria de datos (RAM de arreglo de registros) que tienen los buses separados para poder permitir el acceso simultáneo. El tercer bloque es la Memoria de datos EEPROM (flash) que se puede utilizar para guardar valores de configuración, de estado o datos generados que podrán ser recuperados luego de apagar del microcontrolador.



En la siguiente figura se muestra un esquema con la organización de la memoria de programa. Puede observarse el vector de inicio (reset) se almacena en la dirección 0000h y un único vector de interrupción que se encuentra en la dirección 0004h. La memoria de programa de usuario propiamente dicha, comienza en la posición 0005h y su extensión varía dependiendo del modelo del microcontrolador. También puede observarse una pequeña pila de

8 niveles de profundidad para salvaguardar las direcciones de llamados a subrutinas.

Como la memoria de datos está basada en bancos de registros, el funcionamiento de todos los dispositivos del microcontrolador se implementan físicamente sobre registros. Esta memoria se estructura en varias páginas o bancos que contienen registros de propósitos generales y registros de funciones especiales (ver tabla siguiente de ejemplo para 16F84A).Las posiciones más bajas están reservadas para los registros de funciones especiales, mientras que por encima de estos se encuentran los registros de propósito general, que se utilizan como posiciones de memoria RAM estática.

Como solo una página o bloque de RAM puede estaractivo en un instante de tiempo, la localización de un registro o celda de memoria fuera de la página actual requiere una instrucción de cambio de página. En consecuencia algunos registros de funciones especiales que se utilizan con frecuencia están reflejados en varios bancos, es decir que celdas de memorias con direcciones diferentes se mapean físicamente en una única dirección.Esto permite reducir el código del programa y realizar un acceso rápido al evitar el cambiode banco.

BANCO DE REGISTROS 0			**BANCO DE REGISTROS 1**		
Dir	**Nombre**	**Descripción**	**Dir**	**Nombre**	**Descripción**
00h	INDF*	Contenido apuntado por el registro FSR (no es un registro físico)	80h	INDF*	Contenido apuntado por el registro FSR (no es un registro físico)
01h	TMR0	Contador de 8 bit	81h	OPTION	Configuración asociada a TMR0
02h	PCL*	8 bit bajos del Contador de Programa	82h	PCL*	8 bit bajos del Contador de Programa
03h	STATUS*	Estado de operaciones aritméticas	83h	STATUS*	Estado de operaciones aritméticas
04h	FSR*	Puntero indirecto de direccionamiento de datos	84h	FSR*	Puntero indirecto de direccionamiento de datos
05h	PORTA	Puerto de entrada/salida A	85h	TRISA	Configuración del puerto A
06h	PORTB	Puerto de entrada/salida B	86h	TRISB	Configuración del puerto B
07h	--	No implementada, se lee como "00"	87h	--	No implementada, se lee como "00"
08h	EEDATA	Registro de datos EEPROM	88h	EECON1	Registro de control de EEPROM
09h	EEADR	Registro de direcciones EEPROM	89h	EECON2	Registro de control de EEPROM
0Ah	PCLATH*	Bits más significativos del PC	8Ah	PCLATH*	Bits más significativos del PC
0Bh	INTCON*	Registro de control de interrupciones	8Bh	INTCON*	Registro de control de interrupciones
0Ch a 4Fh	--	68 registros de propósito general (SDRAM)*	8Ch	--*	Mapeados al Banco 0, es decir que la lectura/escritura de estos registros se hace sobre los registros del Banco 0
50h A 7Fh	--	No implementados	D0h A FFh	--	No implementados

Tabla 1. Mapa de bancos del 16F84A con la descripción, nombre y posición de registros.
* registros de bancos diferentes mapeados físicamente a un mismo registro.

Finalmente esta el bloque de memoria de datos EEPROMque se puede utilizar para guardar valores de configuración, de estado o datos generados para recuperarlos luego de apagar del microcontrolador. El acceso para lectura o escritura de esta memoria no se puede realizar de forma directa como se realiza con la RAM, sino indirectamente a través de registros especiales. Es necesario asignar a un registro especial, la dirección de memoria que se quiere acceder y luego utilizar otro registro especial donde se leerá o escribirá el valor de la celda en cuestión. Esta memoria puede tener valores previamente cargados en la etapa de escritura del código del microcontrolador o se puede utilizar valores generados por el programa. El

microcontrolador PIC 16F84A tiene 64 bytes disponibles mientras que el PIC 16F877 tiene 256 bytes, otras líneas de microcontroladores PIC tienen hasta 1024 o 2048 bytes.

Características y prestaciones

A continuación se enumeran las características, prestaciones y dispositivos de los microcontroladores PIC 16F84A y PIC 16F877. También se muestran los integrados con la respectiva distribución de los pines que es una característica muy importante para tener en cuenta al momento de diseñar un sistema que incluya cualquier microcontrolador.

Microcontrolador 16F877	
Procesador	- Núcleo RISC, Arq. Harvard, reloj de hasta 20 MHz (5 MIPS). - 14 Fuentes posibles de interrupción. - Master Clear, Brown Out, Watchdog, Power On. - Set de 35 instrucciones de 14 bits.
Memoria	- Memoria de programa de 8 K palabras de 14 bits. - Memoria de datos de 368 registros de 8 bits. - Memoria EEPROM de 256 registros de 8 bits. - Pila de programa de 8 palabras de 13 bits
Periféricos	- Puertos de entrada/salida programable de hasta 33 bits, que pueden ser usados por otros periféricos. - Dos temporizadores/Contadores de 8 bits y uno de 16 bits. - Dos Puertos de captura/comparación de datos de 16 bits. - Dos Moduladores de ancho de pulso (PWM) de 8 bits. - Conversor Analógico/Digital de 10 bits con un multiplexor de canales para 8 entradas. - Puerto serie síncrono configurable en modo SPI e I2C. - USART, Para conexiones RS 232. - Puerto Paralelo esclavo con 8 bits de datos y 3 bits de control.

Tabla 2. Características y prestaciones del microcontrolador PIC 16F877.

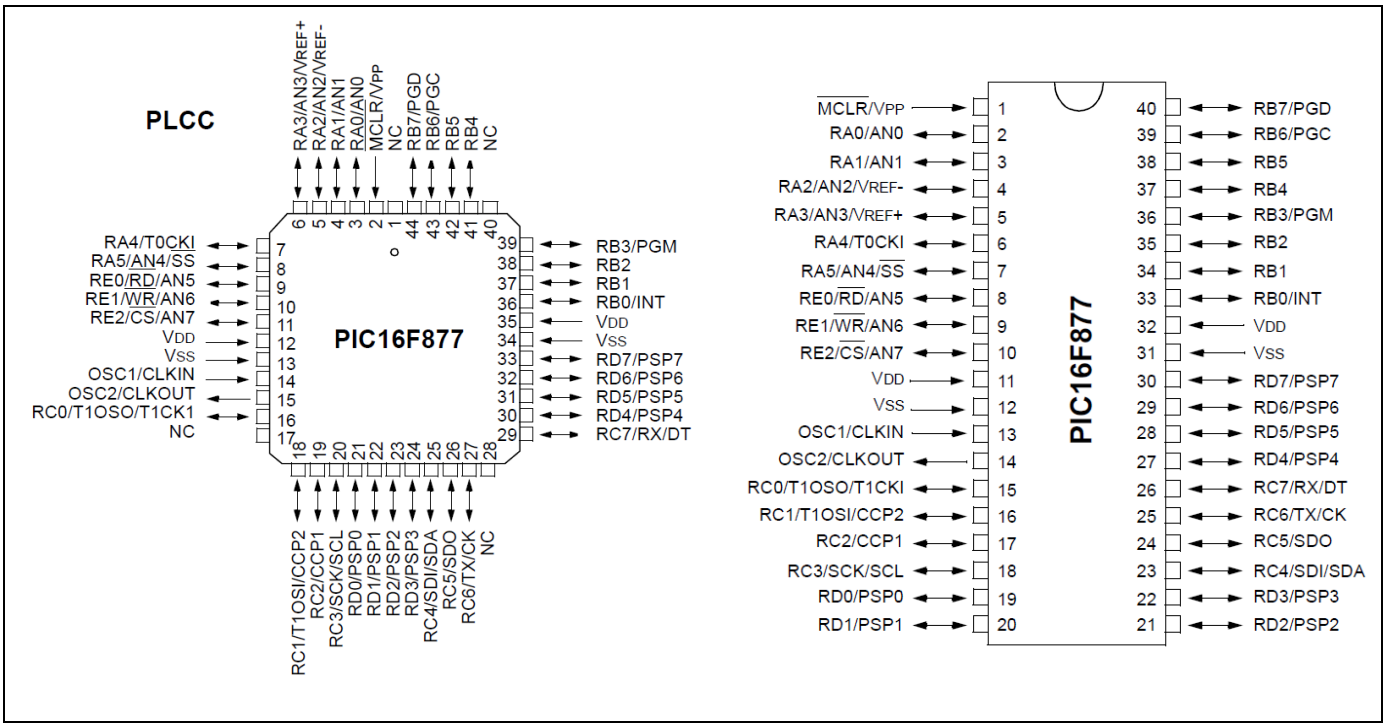


Figura 2. Diagrama con distintos modelos de integrados y la distribución con de pines del PIC 16F877

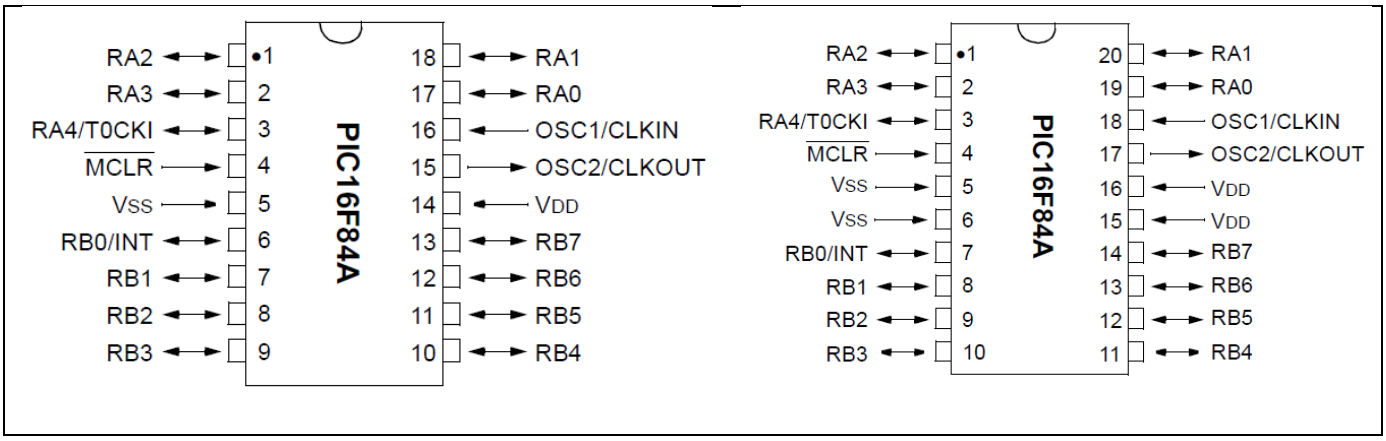


Figura 3. Diagrama con distintos modelos de integrados y su distribución con de pines de PIC 16F84A

Las principales características con que cuenta el 16F84A son:

Microcontrolador 16F84A	
Procesador	- Núcleo RISC, Arq. Harvard, reloj de hasta 20 MHz (5 MIPS). - 4 Fuentes posibles de interrupción. - Master Clear, Brown Out, Watchdog, Power On. - Set de 35 instrucciones de 14 bits.
Memoria	- Memoria de programa de 1 K palabras de 14 bits. - Memoria de datos de 68 registros de 8 bits. - Memoria EEPROM de 64 registros de 8 bits. - Pila de programa de 8 palabras de 13 bits
Periféricos	- Puertos de entrada/salida programable de hasta 13 bits, pueden ser usados por otros periféricos. - Un temporizadores/Contadores de 8 bits con prescaler.

Tabla 3. Características y prestaciones del microcontrolador PIC 16F84A

Entrada/Salida

Módulo de Entrada/Salida

Características

Normalmente, la mayor parte de los pines de los microcontroladores pueden usarse como entradas o salidas digitales, con la salvedad de aquellos que tienen una función fundamental para su funcionamiento como son los pines de alimentación y reinicio. Cada puerto disponible de los microcontroladores se encuentra en una dirección de memoria y se lo denomina con el nombre PORT{X} donde X es una letra, por lo que el primer puerto recibe el nombre PORTA, el segundo PORTB y así sucesivamente. Los bits de cada uno de estos registros se pueden acceder de manera independiente y se los denomina con el nombre R{X}{Y} donde X es el nombre del puerto que pertenecen e Y es el número de bit, por lo que el primer bit del puerto C (PORTC) tiene el nombre RC0 y el último RC7 (notar que los nombres de los pines del microcontrolador reciben también esta denominación).

Los bits de puertos o registros destinados para entrada/salida están conectados directamente con los pines del exterior del Microcontrolador. La lectura de cada bit (cuando está configurado como entrada) toma el

valor del pin asociado y la escritura de cada bit (cuando está configurado como salida) establece la tensión de salida del pin asociado.

Para cada puerto de entrada/salida existe un registro de control que permite configurar, de forma independiente, cada bit como entrada (valor "1") o salida (valor "0"). Estos registros son denominados con TRIS{X} donde X es la misma letra del puerto asociado, por lo que PORTA se configura a través de TRISA, PORTB a través de TRISB y así sucesivamente. La configuración de un puerto siempre depende de las conexiones del pines externos del Microcontrolador. Por ejemplo, si un bit de TRISA se establece en "1" se configura el hardware del bit correspondiente de PORTA para esperar una conexión de entrada (ej: interruptor, pulsador, sensor de contacto, óptico o magnético, estado o señal proveniente de otro dispositivo, etc.). Si un bit de TRISA se establece en "0" se configura el hardware del bit correspondiente de PORTA para esperar una conexión de salida (ej: led, activador de potencia , estado o señal para otro dispositivo, etc.).

El integrado PIC16F84Aposee 2 puertos (A y B) de entrada /salida, mientras que el PIC16F877 posee 5 puertos (A,B, C ,D y E). A continuación se muestra una tabla detallada de los mismos.

	PIC 16F877				
Puerto	**PORTA**	**PORTB**	**PORTC**	**PORTD**	**PORTE**
Dir. Memoria	05H	06H	07H	08H	09H
Control	TRISA	TRISB	TRISC	TRISD	TRISE
Bits de E/S	6	8	8	8	3

PIC 16F84A	
PORTA	**PORTB**
05H	06H
TRISA	TRISB
5	8

Algo importante que se debe tener en cuenta a la hora de configurar los puertos de entrada/salida es si el pin del microcontrolador asociado puede ser compartido con otros módulos de hardware. En estos casos es conveniente leer la especificación técnica del microcontrolador para determinar el comportamiento y la configuración requerida. Para explicar esto de forma más clara puede tomarse como ejemplo el pin RA0/AN0 del microcontrolador 16F877 que puede seleccionarse para realizar conversiones A/D o E/S digital. El registro TRISA controla la dirección de la entrada digital mientras que el registro ADCON1 establece que entradas seránanalógicas. En ambos casos debe asegurarse consistencia:

- para utilizar un pin como E/S digital: TRISA determina si es de entrada o salida, pero ADCON1 debe configurarse para no utilizar dicho pin como entrada analógica.
- para utilizar un pin como entrada analógica: ADCON1 determina que pin se utilizara como entrada analógica, pero TRISA debe configurar dicho pin como entrada.

Programación

Los pasos de programación de un puerto de entrada/salida es sencilla, salvo la consideración del punto anterior. Consiste simplemente en establecer un "1" en cada bit del puerto de configuración que se requiere como entrada y establecer un "0" en cada bit que se requiere como salida. Por ejemplo el siguiente código configura los 7 bits más altos el puerto B como entradas y el último como salida:

```
3   // configuración del puerto B
4   // como 7 entradas y 1 salida
5   TRISB = 0xFE;       // hexadecimal
6   TRISB = 0b11111110 // binario
7   TRISB = 254         // decimal
```

```
3   // Puerto B, 7 entradas y 1 salida
4   // lee 8, ult. bit no importa
5   mi_var = PORTB;
6   // activa salida de puerto B
7   RB0    = 1;
```

Temporizadores

Módulo de temporización "Timer0"

Características

Tanto el microcontrolador 16F84A y como el microcontrolador 16F877 poseen un modulo con un temporizador programable de 8 bits con las mismas características y el mismo funcionamiento. En particular el PIC 16F877 tiene dos módulos de temporización adicionales de 16 bits denominados "Timer1" y "Timer2". Lascaracterísticas del módulo de temporización "Timer0" son:

- funciona como contador de eventos externos al detectar cambios en un pin de entrada o temporizador de 8 bits.
- Permite lectura/escritura.
- Permite el uso de un Prescaler programable de 8 bits.
- Fuente de reloj interno (ciclos de instrucción) o externo a través del pin **T0CLKI**.
- Posibilidad generar interrupción al desbordar la cuenta (cuando pasa de 255 a 0).
- Se puede seleccionar el flanco de transición cuando se utiliza un reloj externo.

Modos de operación

El módulo del Timer0 tiene dos modos de operación. El modo temporizador se selecciona poniendo a cero el bit **T0CS** (registro **OPTION_REG<5>**). En el modo temporizador, el módulo Timer0 se incrementa en cada ciclo de instrucción (sin el preescaler). Si el registro **TMR0** se escribe, el incremento se inhibe durante los siguientes dos ciclos de instrucción. EL usuario puede trabajar teniendo en cuenta esto y ajustando el valor a cargar en el **TMR0**.

El modo contador se selecciona poniendo a uno el bit **T0CS** (registro **OPTION_REG<5>**). En este modo, Timer0 se incrementa en cada flaco de subida o de bajada de la señal que le llega por **RA4/TOCK1**. El flanco de incremento se determina por el bit **T0SE** (registro **OPTION_REG<4>**). Poniéndose a cero **T0SE** se selecciona el flanco ascendente.

Interrupción

La interrupción de Timer0 se produce cuando el registro **TMR0** se desborda al pasar de FFh a 00h. Este desbordamiento pone a uno el bit **T0IF** (**INTCON<2>**). La interrupción puede enmascararse (deshabilitarse) poniendo a cero el bit **T0IE** (**INTCON <5>**). EL bit **T0IF** debe ponerse a cero por software al finalizar la rutina de atención a la interrupción del desbordamiento de **TMR0**. El temporizador permanece deshabilitado en el modo SLEEP por lo que no será posible utilizar la interrupción para despertarlo.

Registros asociados

El Módulo Timer0 tiene varios registros asociados que permiten programarlo y conocer su estado. Estos registros son los siguientes:

- **TMR0**:Se ubica en la dirección 01h de memoria de datos. Almacena el valor del contador y se incrementa con cada ciclo de instrucción o con cada ciclo del reloj externo (según se programe).En particular es importante notar 2 cosas sobre cuando se realiza una escritura sobre este registro:
    - Provoca un retardo de 2 ciclos de instrucción antes de comenzar la nueva cuenta.
    - Provoca la inicialización a 0 del prescaler cuando se encuentra asociado al timer0.

- **OPTION_REG** :Se ubica en la dirección 81h de memoria de datos. Controla la programación del prescaler y la selección de la fuente de reloj con el que el temporizador incrementará su valor. El significado de los bits de este registro es:

Bit	Nombre	Descripción
7	RBPU	No usado por el timer
6	INTEDG	No usado por el timer
5	T0CS	Bit de selección de reloj para incremento(**T**MR**0C**lock**S**ourceSelect): • 1 = reloj externo conectado a pin T0CKI • 0 = reloj interno asociado al ciclo de instrucción.
4	T0SE	Bit de selección de flanco de transición (**T**MR**0S**ource**E**dgeSelect) , solo tiene sentido cuando la fuente es reloj externo (T0CS=1): • 1 = Incrementa en transición alto-bajo del pin T0CKI. • 0 = Incrementa en transición bajo-alto del pin T0CKI.
3	PSA	Bit de asignación de prescaler (**Pre**s**c**aler **A**ssignment): • 1 = Prescaler asignado a Temporizador Watchdog. • 0 = Prescaler asignado a módulo del Timer0.
2,1,0	PS2:PS0	Combinación de 3 bits (valor de 0 a 7) que indican la cantidad de ciclos que debe contar el prescaler antes de generar un ciclo para el temporizador: • 0:0:0=0 ➔ 1:2 (1 ciclo temporizador cada 2 ciclos prescaler). • 0:0:1=1 ➔ 1:4 (1 ciclo temporizador cada 4 ciclos prescaler). • 0:1:0=2 ➔ 1:8 (1 ciclo temporizador cada 8 ciclos prescaler). • 0:1:1=3 ➔ 1:16 (1 ciclo temporizador cada 16 ciclos prescaler). • 1:0:0=4 ➔ 1:32 (1 ciclo temporizador cada 32 ciclos prescaler). • 1:0:1=5 ➔ 1:64 (1 ciclo temporizador cada 64 ciclos prescaler). • 1:1:0=6 ➔ 1:128 (1 ciclo temporizador cada 128 ciclos prescaler). • 1:1:1=7 ➔ 1:256 (1 ciclo temporizador cada 256 ciclos prescaler).

Tabla 4. Descripción del registro OPTION_REG

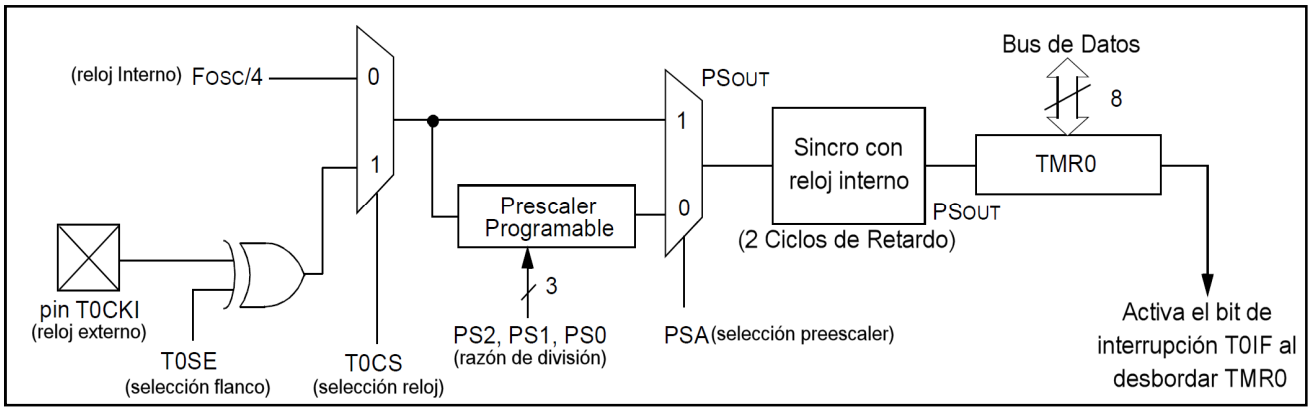


Figura 4. Diagrama en bloques del módulo “Timer0”.

- **INTCON** :Se ubica en la dirección 0Bh de memoria de datos . Controla las interrupciones en general. En cuanto al temporizador posee un bit de control que establece si debe generar interrupción cuando desborde y un bit de estado para indicar que se ha producido el desborde.

Registro de control/estado INTCON		
Bit	**Nombre**	**Descripción**
7	GIE	Habilitación General de Interrupciones (**G**lobal **I**nterrupt**E**nable): • 1 = Habilitación general de interrupciones. • 0 = Deshabilita todas las interrupciones.
6	PEIE	No usado por el timer
5	T0IE	Habilitación de interrupción por desborde del Timer0 (**T**MR**0**Overflow**I**nterrupt**E**nable): • 1 = Habilita interrupción del Timer0. • 0 = Deshabilita interrupción del Timer0.
4	INTE	No usado por el timer
3	RBIE	No usado por el timer
2	T0IF	Flag de estado de interrupción por desborde del Timer0 (**T**MR**0**Overflow**I**nterrupt**F**lag): • 1 = El registro TMR0 ha desbordado (requiere limpiado por soft). • 0 = El registro TMR0 no ha desbordado.
1	INTF	No usado por el timer
0	RBIF	No usado por el timer

Tabla 5. Descripción de registro INTCON.

Cálculos de tiempo

El cálculo del tiempo de interrupción de un temporizador, en este caso del Timer0, tiene cierta complejidad. Para analizar algunas situaciones supongamos las siguientes condiciones:

- El reloj de sistema es de 1Mhz, por lo tanto como 1 instrucción requiere 4 ciclos de reloj el ciclo de instrucción lo que el ciclo de instrucción es de 4 μs o microsegundos (1seg/(1.000.000hz/4hz)).
- La entrada de reloj del Timer0 se programo para estar conectada al reloj de ciclo de instrucción.

Si solo disponemos del Timer0 el tiempo más lento para interrumpir esta dado por los 256 incrementos que hará el temporizador hasta desbordar, por lo que el tiempo que tardará será de 1024μs (4μs *256) o 1,024 milisegundos o ms.

Para tiempos más lentos podemos usar el prescaler como entrada del Timer0. El retardo más grandecon el que podemos programar el prescaler es de 256 ciclos, lo que significa que por cada 256 ciclos de este el temporizador incrementará 1. La cantidad máxima de ciclos antes de desbordar el timer será de 65536 (256*256) con lo que el tiempo de retardo será de262144μs (4μs *256*256) o 262ms, aproximadamente ¼ de segundo.

Supongamos ahora que queremos una espera lo más cercana posible a 100 ms. Para conseguir esto debemos analizar qué valores necesitamos asignar al prescaler y al reloj teniendo en cuenta que el ciclo de instrucción es de 4μs. La fórmula que determina el tiempo que transcurre es:

Tiempo = {tiempo de instrucción} * {ciclos de prescaler} *{ciclos de timer}

Como el prescaler solo puede tomar 8 valores (se utiliza como potencia de 2) podemos despejar la ecuación:

{ciclos de timer}= Tiempo / ({tiempo de instrucción} * {ciclos de prescaler})

Con los valores planteados en esta sección el cálculo de tiempo sería:

{ciclos de timer}= 100000 µs / (4µs * {prescaler con 2, 4, 8, 16, 32, 64, 128 o 256})

La siguiente tabla muestra las combinaciones posibles, teniendo en cuenta los valores de 128 y 256 para el prescaler, ya que aquellos que son menores o iguales a 64 hacen que el tiempo quede por debajo de lo requerido.

Prescaler		**Timer**		**Tiempo en *µs***	
Ciclos	**Valor de Programación (3 bits PS)**	**Ciclos**	**Valor de Programación (256-ciclos)**	**Valor más cercano a 100000**	**Diferencia**
256	8	97	159	99328	662
256	8	98	158	100352	352
128	**7**	**195**	**61**	**99840**	**160**
128	7	196	60	100352	352

Programación

La inicialización del módulo Timer0 debe realizarse de la siguiente manera:

- Establecer el bit de la fuente de reloj del registro OPTION_REG:
  - Reloj interno (incrementa con ciclo de instrucción): T0CS=0.
  - Reloj externo (pin T0CLKI): T0CS=1 y T0SE indicando flanco de transición.
- Establecer los bits asociados al prescaler del registro OPTION_REG:
  - Usa prescaler: PSA=1 y PS2:PS0 con valor para prescaler.
  - No usa prescaler: PSA=0.
- Establecer los bit asociados a la interrupción del registro INTCON:
  - Usa interrupción: T0IE=1 y GIE=1
  - No usa interrupción: T0IE=0 y GIE depende si hay mas dispositivos programados para interrumpir.
- Inicialización del registro TMR0. Esto depende de la función que se realiza con el módulo del Timer0. En general si se lo utiliza sin interrupciones normalmente se lo inicializa en 0. Cuando se lo usa con interrupción puede inicializarse con un valor que se ajuste a la frecuencia de interrupción. Por ejemplo se desea que el timer interrumpa cada 64 incrementos el valor que debería escribirse es 192 (256-64).

En cuanto al uso del Timer0, depende si se programo para interrupciones:

- Sin interrupciones: se consulta (“polling”) el valor del registro TMR0 hasta llegar al valor deseado y luego se reinicia con un valor adecuado.
- Con interrupciones: cuando se ejecuta la interrupción ya ha transcurrido el tiempo programado. Cuando se utiliza múltiples fuentes de interrupción (otros dispositivos programados para interrumpir) debe consultarse el bit T0IF del registro de interrupción INTCON para confirmar que el dispositivo que generó la interrupción es el TMR0. En la subrutina de atención debe limpiarse el bit T0IF para habilitar una nueva interrupción cuando desborde el registro TMR0. Normalmente se vuelve a inicializar el registro TMR0 si se necesita un valor distinto de 0 (toma este valor al desbordar).

Como ejemplo de programación, a continuación se muestra un código que utiliza el prescaler y el temporizador para generar una interrupción con el tiempo máximo de retardo:

```
11  // selecciona prescaler para TIMER0
12  PSA=0;
13  // prescaler divide por 256 => PS2:PS1:PS0=0b111
14  OPTION_REG = (OPTION_REG  & 0b11111110) | 0b111;
15  // Timer0 funciona como temporizador
16  T0CS=0;
17  // habilita interrupción general y de timer
18  INTCON = 0b10100000;
```

```
25 void interrupt manejador(void){
26      // hacer algo
27      // ....
28
29      // si no se limpia el flag
30      // no genera mas int de TMR0
31      T0IF=0;
32 }
```

Conversión Analógica/Digital

Módulo de conversión Analógica/Digital

Características

El módulo de conversión Analógico/Digital (A/D) del microcontrolador 16F877 dispone de hasta 8 canales multiplexados. Tiene una resolución de 10 bits con un tiempo mínimo de conversión de aproximadamente 20μs.

Funcionamiento

A través de la entrada analógica se toma la señal a convertir y se captura por un dispositivo de muestreo y retención ("sample and hold") que después se introduce en el dispositivo de conversión. Este conversor transforma la señal analógica de entrada en un valor digital de 10 bits utilizando aproximaciones sucesivas.

El módulo de conversión A/D permite seleccionar la tensión de referencia para trabajar. Puede usarse la tensión interna VDD (ej: 5V) y la de tierra (0V) o bien una tensión externa que se introduzca entre RA3/AN3/VREF+ y RA2/AN2/-VREF.

El convertidor A/D tiene la capacidad de seguir trabajando mientras el dispositivo esté en el modo SLEEP. Para ello el convertidor A/D debe estar conectado al oscilador interno RC.

Registros asociados

El Módulo de conversión A/D tiene cuatro registros asociados que permiten programarlo y conocer su estado. Estos registros son los siguientes:

- **ADRESH**(**A/DRes**ult**H**igh): parte alta del resultado de la conversión (8 o 2 bits más significativos).
- **ADRESL**(**A/DRes**ult**L**ow): parte baja del resultado de la conversión (2 u 8 bits menos significativos).
- **ADCON0** (**A/D Con**trol **0):** registro de Control 0, configura la operación del módulo A/D.
- **ADCON1** (**A/D Con**trol **1):** registro de Control 1, configura la función de los pines del puerto.
- **TRISA:** registro de configuración de puerto A. Cuando se configura un pin de entrada A/D en el registro **ADCON1** debe configurarse el bit correspondiente del registro **TRISA** también como entrada.
- **PIE1 (P**eripherical**I**nterrupt**E**nable **1):** controla la habilitación de interrupciones asociadas a los periféricos.
- **PIR1 (P**eripherical**I**nterrupt**R**egister **1):** contiene los bits de estado de interrupción asociados con los periféricos.
- **INTCON (Int**errupt **Con**trol**):** controla la habilitación general de interrupciones.

Registro de control ADCON0		
Bit	**Nombre**	**Descripción**
7,6	ADCS1:ADCS0	Bits de selección del reloj para realizar la conversión (**A/DC**lock**S**election): • 0:0 = 0 ➔ FOSC/2 (1 ciclo del reloj de conversión cada 2 del reloj del sistema). • 0:1 = 1 ➔ FOSC/8 (1 ciclo del reloj de conversión cada 8 del reloj del sistema). • 1:0 = 2 ➔ FOSC/32 (1 ciclo del reloj de conversión cada 32 del sistema). • 1:1 = 3 ➔ FRC : reloj derivado del oscilador RC interno del modulo A/D
5,4,3	CHS2:CHS0	Selector del canal A/D para convertir (**Ch**annel**S**election): • 0:0:0 = 0 ➔ canal 0, pin RA0/AN0 • 0:0:1 = 1 ➔ canal 1, pin RA1/AN1 • 0:1:0 = 2 ➔ canal 2, pin RA2/AN2 • 0:1:1 = 3 ➔ canal 3, pin RA3/AN3 • 1:0:0 = 4 ➔ canal 4, pin RA5/AN4 • 1:0:1 = 5 ➔ canal 5, pin RE0/AN5 • 1:1:0 = 6 ➔ canal 6, pin RE1/AN6 • 1:1:1 = 7 ➔ canal 7, pin RE2/AN7
2	GO/$\overline{\text{DONE}}$	Bit de estado y control de la conversión: • 1 ➔ conversión A/D en progreso. Si se escribe un “1” en este bit comienza la conversión. • 0 ➔no hay conversión A/D en progreso. Este bit es puesto en “0” por hardware cuando se completa la conversión.
1	-	No implementado. Al leerlo siempre hay un “0”.
0	ADON	Bit de control para activar el módulo de conversión A/D (**A/D ON**): • 1 ➔El módulo A/D esta activado. • 0 ➔El módulo A/D esta desactivado. No consume corriente.

Tabla 6. Descripción del registro ADCON0.

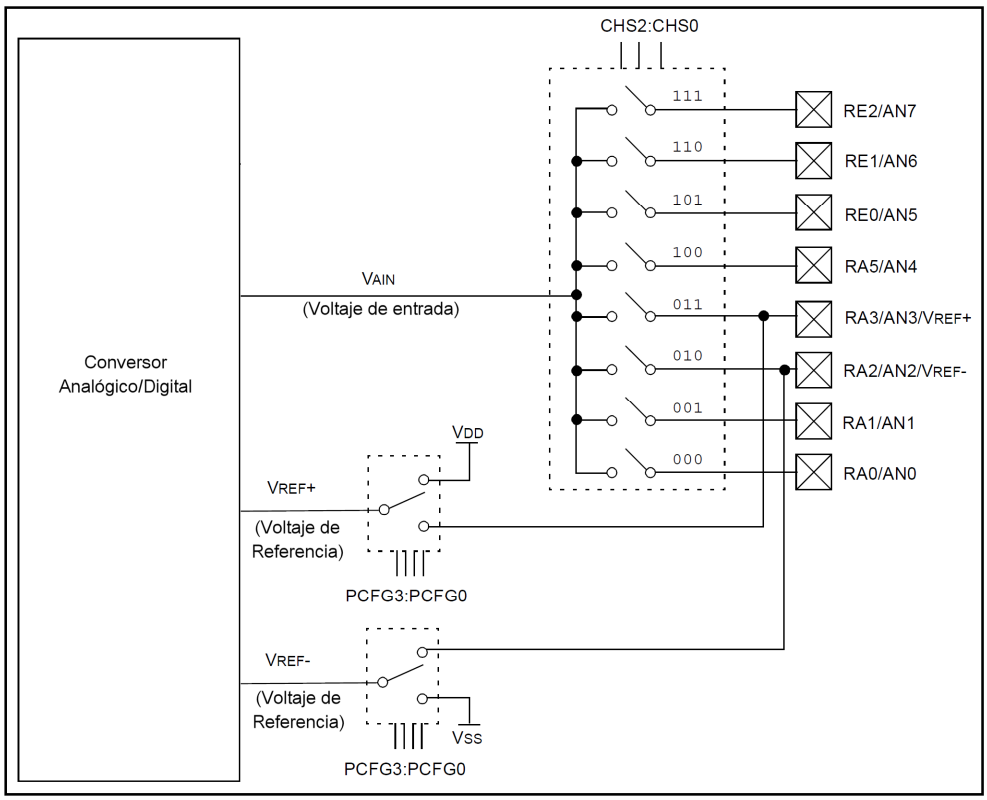


Figura 5. Esquema de modulo de conversión analógica/digital.

<table>
<tr><th colspan="3">Registro de control ADCON1</th></tr>
<tr><th>Bit</th><th>Nombre</th><th>Descripción</th></tr>
<tr><td>7</td><td>ADFM</td><td>Bit de selección para establecer la forma en que se almacena el valor de la conversión A/D en los registros ADRESH:ADRESL :
1 ➔ Justificar a derecha: los 10 bits del valor A/D se almacenan en los 10 menos significativos (2 en ADRESH y 8 en ADRESL). Los 6 bits más significativos quedan en “0”.
0 ➔ Justificar a izquierda: los 10 bits del valor A/D se almacenan en los 10 más significativos (8 en ADRESH y 2 en ADRESL). Los 6 bits menos significativos quedan en “0”. Si no interesa la precisión esta configuración es práctica para leer solo ADRESH donde están los 8 bits más significativos.</td></tr>
<tr><td>6,5,4</td><td>-</td><td>No implementados. Siempre se lee “0”</td></tr>
<tr><td>3,2,1,0</td><td>PCFG3:PCFG0</td><td>Bits de configuración de la combinación de bits para el puerto de control A/D. La combinación de estos bits establece la cantidad de entradas analógicas, entradas digitales y entradas que se utilizarán como tensión de referencia:
<table>
<tr><th>Valor PCFG</th><th>AN7 RE2</th><th>AN6 RE1</th><th>AN5 RE0</th><th>AN4 RA5</th><th>AN3 RA3</th><th>AN2 RA2</th><th>AN1 RA1</th><th>AN0 RA0V</th><th>V_{REF+}</th><th>V_{REF-}</th><th>Canales/Ref</th></tr>
<tr><td>0000</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>V_{DD}</td><td>V_{SS}</td><td>8/0</td></tr>
<tr><td>0001</td><td>A</td><td>A</td><td>A</td><td>A</td><td>V_{REF+}</td><td>A</td><td>A</td><td>A</td><td>RA3</td><td>V_{SS}</td><td>7/1</td></tr>
<tr><td>0010</td><td>D</td><td>D</td><td>D</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>V_{DD}</td><td>V_{SS}</td><td>5/0</td></tr>
<tr><td>0011</td><td>D</td><td>D</td><td>D</td><td>A</td><td>V_{REF+}</td><td>A</td><td>A</td><td>A</td><td>RA3</td><td>V_{SS}</td><td>4/1</td></tr>
<tr><td>0100</td><td>D</td><td>D</td><td>D</td><td>D</td><td>A</td><td>D</td><td>A</td><td>A</td><td>V_{DD}</td><td>V_{SS}</td><td>3/0</td></tr>
<tr><td>0101</td><td>D</td><td>D</td><td>D</td><td>D</td><td>V_{REF+}</td><td>D</td><td>A</td><td>A</td><td>RA3</td><td>V_{SS}</td><td>2/1</td></tr>
<tr><td>011X</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>V_{DD}</td><td>V_{SS}</td><td>0/0</td></tr>
<tr><td>1000</td><td>A</td><td>A</td><td>A</td><td>A</td><td>V_{REF+}</td><td>V_{REF-}</td><td>A</td><td>A</td><td>RA3</td><td>RA2</td><td>6/2</td></tr>
<tr><td>1001</td><td>D</td><td>D</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>RA3</td><td>V_{SS}</td><td>6/0</td></tr>
<tr><td>1010</td><td>D</td><td>D</td><td>A</td><td>A</td><td>V_{REF+}</td><td>A</td><td>A</td><td>A</td><td>RA3</td><td>V_{SS}</td><td>5/1</td></tr>
<tr><td>1011</td><td>D</td><td>D</td><td>A</td><td>A</td><td>V_{REF+}</td><td>V_{REF-}</td><td>A</td><td>A</td><td>RA3</td><td>RA2</td><td>4/2</td></tr>
<tr><td>1100</td><td>D</td><td>D</td><td>D</td><td>A</td><td>V_{REF+}</td><td>V_{REF-}</td><td>A</td><td>A</td><td>RA3</td><td>RA2</td><td>3/2</td></tr>
<tr><td>1101</td><td>D</td><td>D</td><td>D</td><td>D</td><td>V_{REF+}</td><td>V_{REF-}</td><td>A</td><td>A</td><td>RA3</td><td>RA2</td><td>2/2</td></tr>
<tr><td>1110</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>A</td><td>V_{DD}</td><td>V_{SS}</td><td>1/0</td></tr>
<tr><td>1111</td><td>D</td><td>D</td><td>D</td><td>D</td><td>V_{REF+}</td><td>V_{REF-}</td><td>D</td><td>A</td><td>RA3</td><td>RA2</td><td>½</td></tr>
</table>
A= Entrada Analógica; D = E/S Digital</td></tr>
</table>

Tabla 10. Descripción del registro ADCON1.

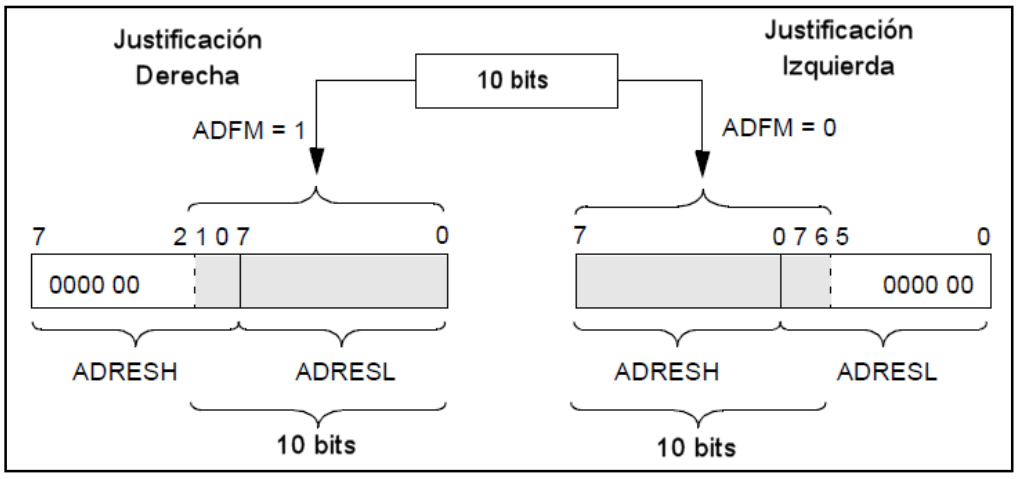


Figura 7. Esquema con el formato de los 10 bits de conversión.

Registro de control PIE1 (dirección 8Ch)		
Bit	**Nombre**	**Descripción**
7	PSPIE	No relacionado con el módulo A/D.
6	ADIE	Bit de habilitación de interrupción al finalizar una conversión A/D (**A/DI**nterrupt**E**nabled): • 1 ➔ habilita interrupción A/D. • 0 ➔ deshabilita interrupción A/D.
5,4,3, 2,1,0	RCIE, TXIE, SSPIE, CCP1IE, TMR2IE,	No relacionado con el módulo A/D.

Tabla 8. Descripción del registro PIE1.

Registro de control PIR1 (dirección 0Ch)		
Bit	**Nombre**	**Descripción**
7	PSPIF	No relacionado con el módulo A/D.
6	ADIF	Bit de interrupción de finalización de conversión A/D (**A/DI**nterrupt**F**lag): • 1 ➔ conversión A/D completa. • 0 ➔ conversión A/D en curso.
5,4,3, 2,1,0	RCIF, TXIF, SSPIF, CCP1IF, TMR2IF, TMR1IF	No relacionado con el módulo A/D.

Tabla 9. Descripción del registro PIR1.

Programación

La inicialización del módulo A/D debe realizarse de la siguiente manera:

- Configurar el módulo A/D:
  - Inicializar el registro **ADCON1**, seleccionando la combinación de los pines Analógicos y la tensión de referencia a utilizar (bits **PCFG**) y el formato del resultado (bit **ADFM**).
  - Inicializar el registro **TRISA**, seleccionando los bits del registro **TRISA** como entradas para que coincidan con los pines de entrada del registro **ADCON1**.
  - Inicializar el registro **ADCON0**, seleccionando la frecuencia del reloj para la conversión (bits **ADCS**), el canal de entrada para la conversión (bits **CHS**) y la activación del módulo A/D (bit **ADON**).
- Configurar interrupción (opcional):
  - Configurar el registro PIR1, poniendo en "0" el bit ADIF.
  - Configurar el registro PIE1, poniendo en "1" el bit ADIE para habilitar las interrupciones A/D.
  - Configurar el registro INTCON, poniendo en "1" los bits PEIE y GIE para habilitar las interrupciones de periféricos y habilitar las interrupciones generales, respectivamente.

Por ejemplo para configurar el módulo A/D sin interrupciones utilizando solo el canal AN0, sin disparar una conversión:

```
ADCON1 = 0b10001110; // ADFM=1 (ADRESH:ADRESL 2:8 bit), xxx, PCFG3:PCFG0 =14 (AN0 con VDD)
ADCON0 = 0b01000001; // ADCS1:ADCS0=0 (OSC/8), CHS2:CHS0=0 (canal 0), GO=0, ADON=1 (activa modulo)
ADIE   = 0;          // ADIE (bit 6 de PIE1) = 0 (sin interrupciones)
TRISA  = TRISA | 1;  // bit 0 en "1" porque AN0 se utiliza como entrada
```

Para la adquisición de un valor de entrada se deben realizar los siguientes pasos:

- Configurar **ADCON0** con el canal deseado en los bits **CHS** y disparando la conversión poniendo en "1" el bit **GO**. En este paso la selección del canal solo se realiza si se está trabajando con más de un canal A/D ya que de lo contrario quedo configurado en la inicialización.
- Esperar el tiempo necesario de conversión:
    - Sin interrupción: realizando "polling" sobre el bit **GO/DONE** del puerto **ADCON0** . Al finalizar la conversión este bit se pone en "0".
    - Con interrupción: el microcontrolador invoca la rutina de interrupción cuando el bit **GO/DONE** se pone en "0". Si está configurado más de un dispositivo para interrumpir, se puede consultar el bit **ADIF** del registro **PIR1** con el fin de identificar que la fuente de interrupción es una conversión A/D.
- Leer los 10 bits de los registros **ADRESH:ADRESL** según el formato configurado con el bit **ADFM** del registro **ADCON1**.
- Solo si se habilito interrupción para el módulo A/D, poner en "0" el bit ADIF del registro PIR1 para permitir una nueva interrupción.

A continuación se muestran 3 ejemplos diferentes para adquirir un valor por medio de una conversión A/D. En los 3 casos se realiza la conversión sin interrupciones y utilizando solo el canal 0 (entrada AN0):

```
50  // lecturas continuas
51  // inicio de conversión sin selección de canal
52  GO=1;
53  while (1){
54     while (GO){} // espera de conversión
55     // lee A/D asumiendo formato ADRES 2:8
56     unsigned short valor = ADRESH << 8 | ADRESL;
57     // dispara conversión para prox. iteración
58     GO=1;
59     // sigue codigo con procesamiento de valor
60     // mientras termina la nueva conversion
```

```
51  // inicio de conversión sin selecciar canal
52  GO=1;
53  while (GO){} // espera de conversión
54  // lee A/D asumiendo formato ADRES 2:8
55  unsigned short valor = ADRESH << 8 | ADRESL;
```

```
58  // inicio de conversión sin selección de canal
59  ADCON = ADCON | 0b00000100;
60  //  espera de conversión
61  while (ADCON & 0b00000100) {}
62  // lee A/D asumiendo formato de ADRES 2:8
63  unsigned short valor = ADRESH << 8 | ADRESL;
```