El PIC16F84 es un microcontrolador con memoria de programa tipo *FLASH*, lo que representa gran facilidad en el desarrollo de prototipos y en su aprendizaje ya que no se requiere borrarlo con luz ultravioleta como las versiones EPROM sino, permite reprogramarlo nuevamente sin ser borrado con anterioridad. Por esta razón, lo usaremos en la mayoría de aplicaciones que se desarrollan a lo largo del curso.

Pines y funciones

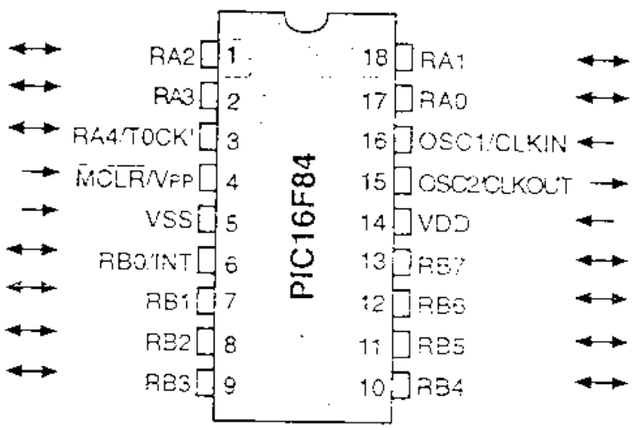

Figura 1.1. Diagrama de pines del PIC16F84

El PIC16F84 es un microcontrolador de *Microchip Technology* fabricado en tecnología CMOS, su consumo de potencia es muy bajo y además es completamente estático, esto quiere decir que el reloj puede detenerse y los datos de la memoria no se pierden.

El encapsulado más común para el microcontrolador es el DIP (*Dual In-line Pin*) de 18 pines, propio para usarlo en experimentación. La referencia completa es 16F84-04/P, para el dispositivo que utiliza reloj de 4 MHz. Sin embargo, hay otros tipos de encapsulado que se pueden utilizar según el diseño y la aplicación que se quiere realizar. Por ejemplo, el encapsulado tipo *surface mount* (montaje superficial) tiene un reducido tamaño y bajo costo, que lo hace propio para producciones en serie o para utilizarlo en lugares de espacio muy reducido, la figura 1.2 muestra los tipos de empaque que puede tener el integrado.

Figura 1.2. Tipos de encapsulado

Puertos del microcontrolador

Los puertos son el puente entre el microcontrolador y el mundo exterior. Son líneas digitales que trabajan entre cero y cinco voltios y se pueden configurar como entradas o como salidas.

El PIC16F84 tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8 líneas, figura 1.3. Cada pin se puede configurar como entrada o como salida independiente programando un par de registros diseñados para tal fin. En ese registro un "0" configura el pin del puerto correspondiente como salida y un "1" lo configura como entrada.

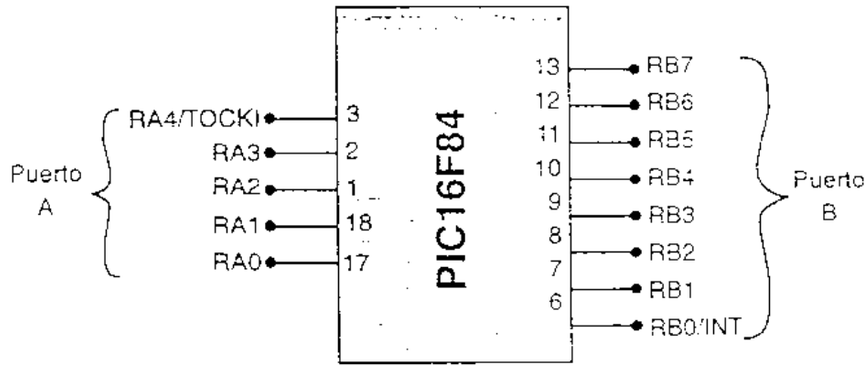

Figura 1.3. Puertos del PIC16F84

El puerto B tiene internamente unas resistencias de *pull-up* conectadas a sus pines (sirven para fijar el pin a un nivel de cinco voltios), su uso puede ser habilitado o deshabilitado bajo control del programa. Todas las resistencias de pull-up se conectan o se desconectan a la vez, usando el bit llamado RBPU que se encuentra en el registro (posición de memoria RAM) llamado OPTION. La resistencia de pull-up es desconectada automáticamente en un pin si este se programa como salida. El pin RB0/INT se puede configurar por software para que funcione como interrupión externa, para configurarlo se utilizan unos bits de los registros INTCON y OPTION.

El pin RA4/TOCKI del puerto A puede ser configurado como un pin de entrada/salida o como entrada del temporizador/contador. Cuando este pin se programa como entrada digital, funciona como un disparador de *Schmitt* (*Schmitt trigger*), puede reconocer señales un poco distorsionadas y llevarlas a niveles lógicos (cero y cinco voltios). Cuando se usa como salida digital se comporta como colector abierto, por lo tanto, se debe poner una resistencia de *pull-up* (resistencia externa conectada a un nivel de cinco voltios). Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega en el pin un "1" lógico. Además, como salida no puede manejar cargas como fuente, sólo en el modo sumidero.

Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectados a alguna parte, nunca dejarlos al aire porque se puede dañar el integrado. Los pines que no se estén usando se deben conectar a la fuente de alimentación de +5V, como se muestra en la figura 1.4.

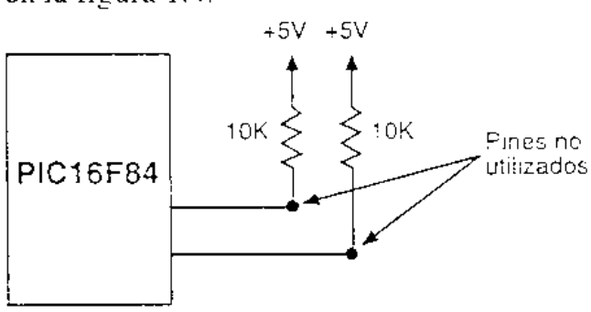

Figura 1.4. Los puertos no utilizados se deben conectar a la fuente

La máxima capacidad de corriente de cada uno de los pines de los puertos en modo sumidero (*sink*) es de 25 mA y en modo fuente (*source*) es de 20 mA, figura 1.5. La máxima capacidad de corriente total de los puertos es:

	PUERTO A	PUERTO B
Modo sumidero	80 mA	150 mA
Modo fuente	50 mA	100 mA

El consumo de corriente del microcontrolador para su funcionamiento depende del voltaje de operación, la frecuencia y de las cargas que tengan sus pines. Para un reloj de 4 MHz el consumo es de aproximadamente 2 mA; aunque este se puede reducir a 40 microamperios cuando se está en el modo *sleep* (en este modo el micro se detiene y disminuye el consumo de potencia). Se sale de ese estado cuando se produce alguna condición especial que veremos más adelante.

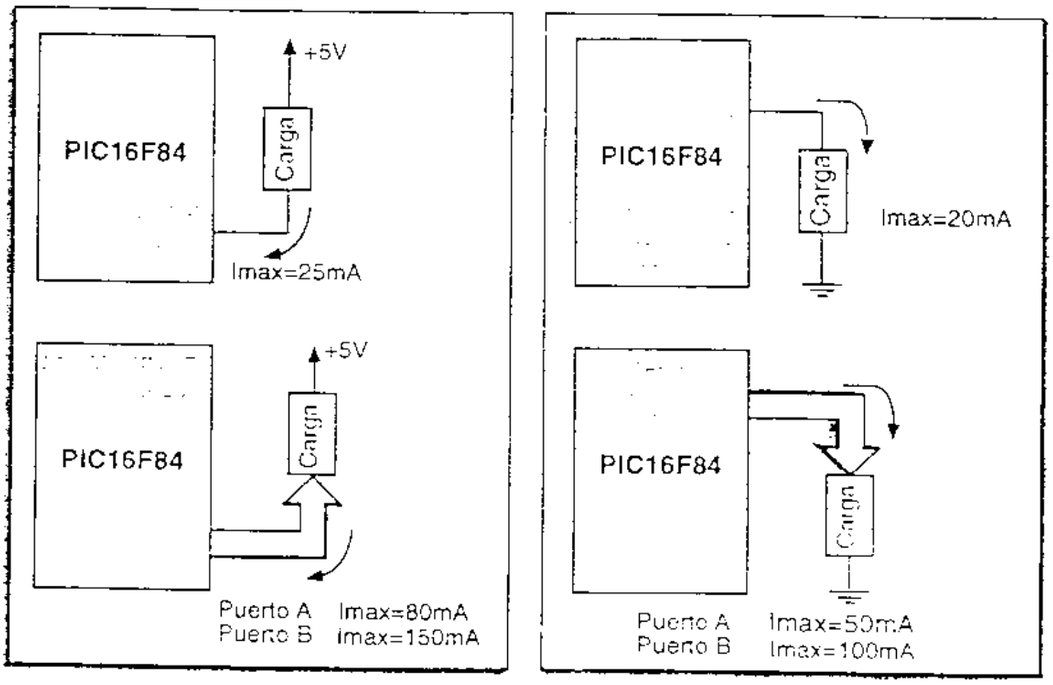

Figura 1.5. Capacidad de corriente del PIC16F84

El oscilador externo

Todo microcontrolador requiere un circuito externo que le indique la velocidad a la que debe trabajar. Este circuito, que se conoce como oscilador o reloj, es muy simple pero de vital importancia para el buen funcionamiento del sistema. El PIC16F84 puede utilizar cuatro tipos de oscilador diferentes. Estos tipos son:

- **RC**. Oscilador con resistencia y condensador.
- **XT**. Cristal.
- **HS**. Cristal de alta velocidad.
- **LP**. Cristal para baja frecuencia y bajo consumo de potencia.

En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados "*fusibles de configuración*".

El tipo de oscilador que se sugiere para las prácticas es el cristal de 4 MHz, porque garantiza mayor precisión y un buen arranque del microcontrolador. Internamente esta frecuencia es dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se ejecuta en un microsegundo. El cristal debe ir acompañado de dos condensadores y se conecta como se muestra en la figura 1.6.

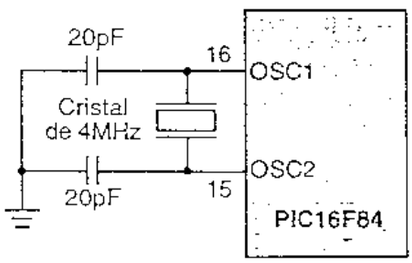

Figura 1.6. Conexión de un oscilador a cristal

Dependiendo de la aplicación, se pueden utilizar cristales de otras frecuencias; por ejemplo se usa el cristal de 3.579545 MHz porque es muy económico, el de 32.768 KHz cuando se necesita crear bases de tiempo de un segundo muy precisas. El límite de velocidad en estos microcontroladores es de 10 MHz.

Si no se requiere mucha precisión en el oscilador y se quiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra en la figura 1.7.

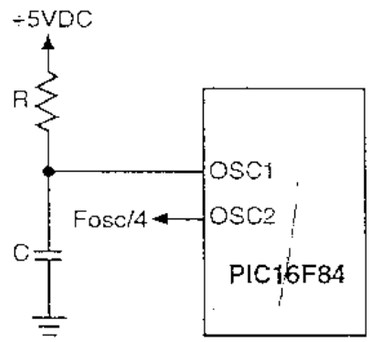

Figura 1.7. Conexión de un oscilador RC

Reset

En los microcontroladores se requiere un pin de *reset* para reiniciar el funcionamiento del sistema cuando sea necesario, ya sea por una falla que se presente o porque así se halla diseñado el sistema. El pin de reset en los PIC es llamado $\overline{\text{MCLR}}$ (*master clear*). El PIC16F84 admite diferentes tipos de *reset*:

- Al encendido (*Power On Reset*)
- Pulso en el pin $\overline{\text{MCLR}}$ durante operación normal

- Pulso en el pin $\overline{\text{MCLR}}$ durante el modo de bajo consumo (modo *sleep*)
- El rebase del conteo del circuito de vigilancia (*watchdog*) durante operación normal
- El rebase del conteo del circuito de vigilancia (*watchdog*) durante el modo de bajo consumo (*sleep*)

El *reset* al encendido se consigue gracias a dos temporizadores. El primero de ellos es el OST (*Oscillator Start-Up Timer:* Temporizador *de e*ncendido del oscilador), orientado a mantener el microcontrolador en *reset* hasta que el oscilador del cristal es estable. El segundo es el PWRT (*Power-Up Timer :* Temporizador de encendido), que provee un retardo fijo de 72 ms (nominal) en el encendido únicamente, diseñado para mantener el dispositivo en *reset* mientras la fuente se estabiliza. Para utilizar estos temporizadores, sólo basta con conectar el pin $\overline{\text{MCLR}}$ a la fuente de alimentación, evitándose utilizar las tradicionales redes RC externas en el pin de *reset*.

El *reset* por $\overline{\text{MCLR}}$ se consigue llevando momentáneamente este pin a un estado lógico bajo, mientras que el watchdog WDT produce el *reset* cuando su temporizador rebasa la cuenta, o sea que pasa de 0FFh a 00h. Cuando se quiere tener control sobre el *reset* del sistema se puede conectar un botón como se muestra en la figura 1.8.

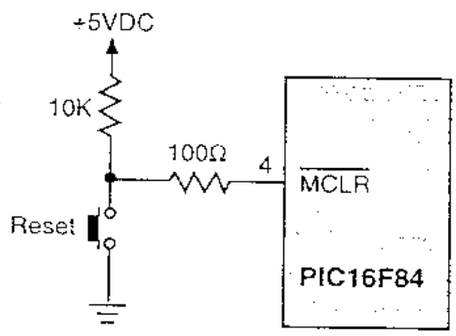

Figura 1.8. Conexión del botón de reset

Arquitectura

Este término se refiere a los bloques funcionales internos que conforman el microcontrolador y la forma en que están conectados, por ejemplo la memoria FLASH (de programa), la memoria RAM (de datos), los puertos, la lógica de control que permite que todo el conjunto funcione, etc.

La figura 1.9 muestra la arquitectura general del PIC16F84, en ella se pueden apreciar los diferentes bloques que lo componen y la forma en que se conectan. Se muestra la conexión de los puertos, las memorias de datos y de programa, los bloques especiales como el *watchdog*, los temporizadores de arranque, el oscilador, etc.

Todos los elementos se conectan entre sí por medio de buses. Un bus es un conjunto de líneas que transportan información entre dos o más módulos. Vale la pena destacar que el PIC16F84 tiene un bloque especial de memoria de datos de 64 bytes del tipo EEPROM, además de los dos bloques de memoria principales que son el de programa y el de datos o registros.

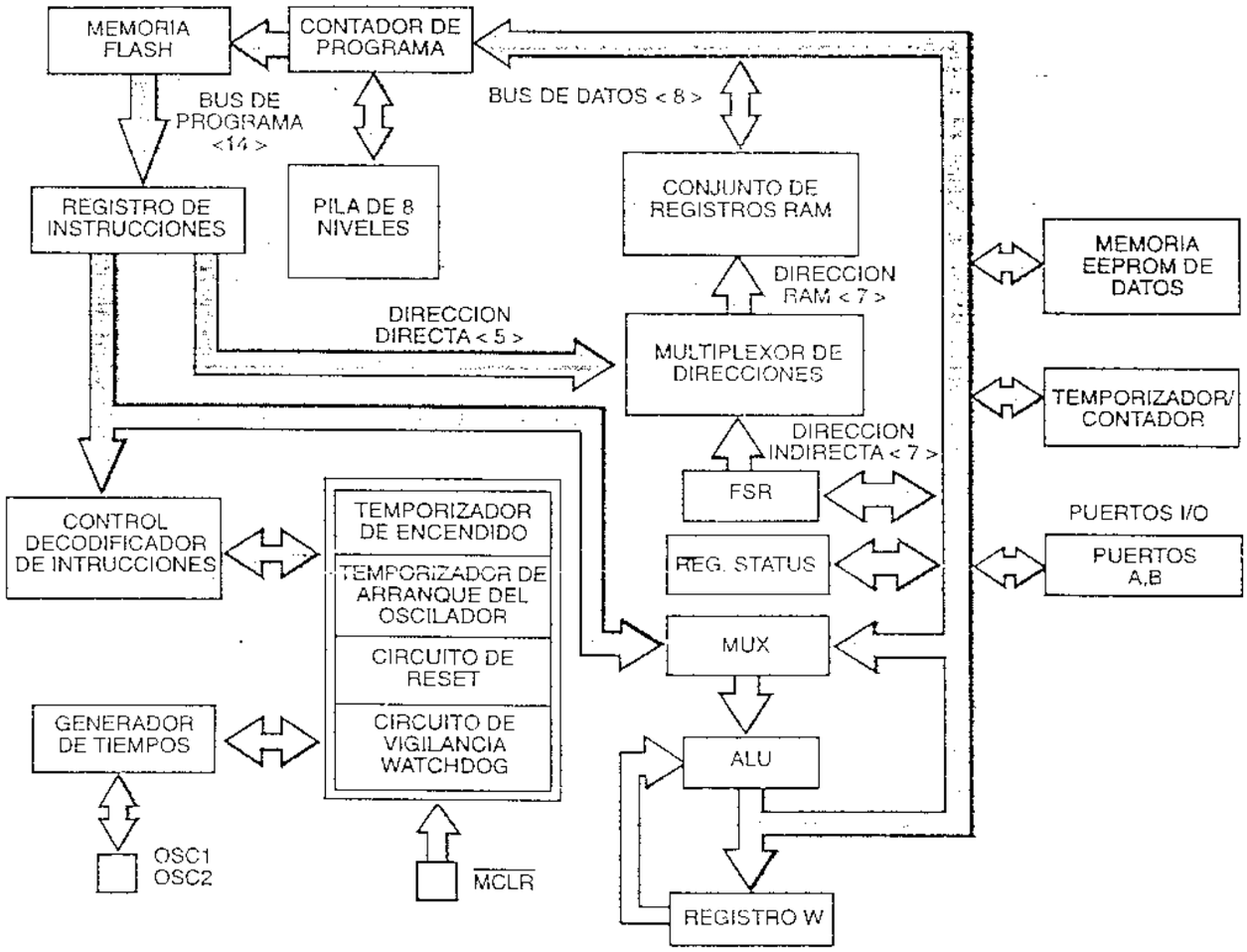

Figura 1.9. Arquitectura interna del PIC16F84

El PIC16F84 se basa en la arquitectura *Harvard*, en la cual el programa y los datos se pueden trabajar desde memorias separadas, lo que posibilita que las instrucciones y los datos posean longitudes diferentes. Esta misma estructura es la que permite la superposición de los ciclos de búsqueda y ejecución de las instrucciones, lo cual se ve reflejado en una mayor velocidad del microcontrolador.

Memoria de programa

Es una memoria de 1 Kbyte de longitud con palabras de 14 bits. Como es del tipo *FLASH* se puede programar y borrar eléctricamente, lo que facilita el desarrollo de los programas y la experimentación. En ella se graba, o almacena, el programa o códigos que el microcontrolador debe ejecutar. Dado que el PIC16F84 tiene un contador de programa de 13 *bits*, tiene una capacidad de direccionamiento de 8K x 14, pero solamente tiene implementado el primer 1K x 14 (0000h hasta 03FFh). Si se direccionan posiciones de memoria superiores a 3FFh se causará un solapamiento con el espacio del primer 1K. En la figura 1.10 se muestra el mapa de la memoria de programa.

Vector de reset. Cuando ocurre un reset al microcontrolador, el contador de programa se pone en ceros (000H). Por esta razón, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

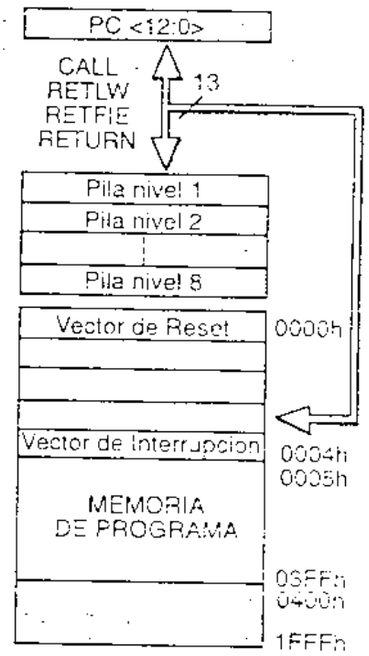

Figura 1.10. Mapa de la memoria de programa

Vector de interrupción. Cuando el microcontrolador recibe una señal de interrupción, el contador de programa apunta a la dirección 04H de la memoria de programa, por eso, allí se debe escribir toda la programación necesaria para atender dicha interrupción.

Registros (Memoria RAM)

El PIC16F84 puede direccionar 128 posiciones de memoria RAM, pero solo tiene implementados físicamente los primeros 80 (0-4F en hexadecimal). De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 68 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se está ejecutando, figura 1.11.

Los registros están organizados como dos arreglos (páginas) de 128 posiciones de 8 *bits* cada una (128 x 8); todas las posiciones se pueden accesar directa o indirectamente (esta última a través del registro selector FSR). Para seleccionar que pagina de registros se trabaja en un momento determinado se utiliza el *bit* RP0 del registro STATUS. A continuación haremos una descripción de los registros:

00h o INDO: Registro para direccionamiento indirecto de datos. Este no es un registro disponible físicamente: utiliza el contenido del FSR y el *bit* RP0 del registro STATUS para seleccionar indirectamente la memoria de datos o RAM del usuario: la instrucción determinará que se debe realizar con el registro señalado.

01h o TMR0. Temporizador/contador de 8 *bits*. Este se puede incrementar con una señal externa aplicada al pin RA4/TOCKI o de acuerdo a una señal interna proveniente del reloj de instrucciones del microcontrolador. La rata de incremento del registro se puede determinar por medio de un preescalador, localizado en el registro OPTION. Como una mejora con respecto a sus antecesores, se le ha agregado la generación de interrupción cuando se rebasa la cuenta (el paso de 0FFh a 00h).

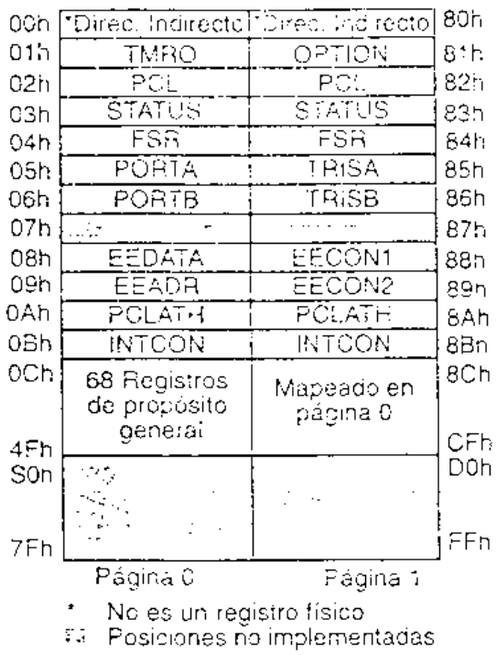

Figura 1.11. Registros del PIC16F84

02h o PCL: Contador de programa. Se utiliza para direccionar las palabras de 14 *bits* del programa del usuario que se encuentra almacenado en la memoria ROM; este contador de programas es de 13 *bits* de ancho, figura 1.12. Sobre el *byte* bajo, se puede escribir o leer directamente, mientras que sobre el byte alto, no. El byte alto se maneja mediante el registro PCLATH (0Ah). A diferencia de los PIC de primera generación, el 16F84 ante una condición de *reset* inicia el contador de programa con todos sus *bits* en "cero". Durante la ejecución normal del programa, y dado que todas las instrucciones ocupan sólo una posición de memoria, el contador se incrementa en uno con cada instrucción, a menos que se trate de alguna instrucción de salto.

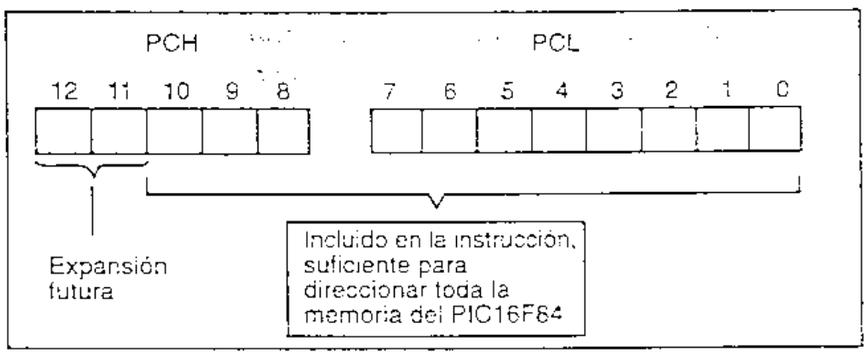

Figura 1.12. Contador de programa (13 bits)

En una instrucción CALL o GOTO, los bits PC<10:0> se cargan desde el código de operación de la instrucción, mientras que los bits PC<11:12> lo hacen desde el PCLATH<4:3>. Como solamente el primer 1K de memoria está implementado, el código de operación de la instrucción puede contener la dirección destino, eso quiere decir que se pueden hacer saltos y llamados a subrutinas sin necesidad de tener en cuenta la paginación de memoria de programa.

En otras instrucciones donde PCL es el destino, PC<12:8> se carga directamente desde el PCLATH<4:0>, por ejemplo en el caso de la instrucción ADDWF. Esto se debe tener en cuenta cuando se desea hacer lectura de tablas usando el comando: ADDWF PC,1 , en este caso se debe tener en cuenta que la tabla debe estar comprendida dentro de un solo bloque de 256 bytes (0-255, 256-511, etc.).

03h o STATUS: Registro de estados. Contiene el estado aritmético de la ALU, la causa del *reset* y los *bits* de preselección de página para la memoria de datos. La figura 1.13 muestra los *bits* correspondientes a este registro. Los *bits* 5 y 6 (RP0 y RP1) son los *bits* de selección de página para el direccionamiento directo de la memoria de datos; solamente RP0 se usa en los PIC16F84. RP1 se puede utilizar como un *bit* de propósito general de lectura/escritura. Los *bits* TO y PD no se pueden modificar por un proceso de escritura; ellos muestran la condición por la cual se ocasionó el último *reset*.

Registro: STATUS

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C

bit 7 bit 0

Dirección: 03h
condición de *reset*: 000??XXX

IRP: Selector de página para direccionamiento indirecto Este *bit* no se utiliza efectivamente en el PIC16F84, por lo que se puede utilizar como un *bit* de propósito general.

RP1,0: Selectores de página para direccionamiento directo. Solamente RP0 se utiliza en el PIC16F84. RP1 se puede utilizar como un *bit* de propósito general.

$\overline{TO}$: *Time Out* o *Bit* de finalización del temporizador. Se coloca en 0 cuando el circuito de vigilancia *Watchdog* finaliza la temporización.

$\overline{PD}$: *Power Down* o *Bit* de bajo consumo. Se coloca en 0 por la instrucción *SLEEP*.

Z: *Zero* o *Bit* de cero. Se coloca en 1 cuando el resultado de una operación lógica o aritmética es cero.

DC: *Digit Carry* o *Bit* de acarreo de dígito. En operaciones aritméticas se activa cuando hay acarreo entre el *bit* 3 y el 4.

C: *Carry* o *Bit* de acarrreo. En instrucciones aritméticas se activa cuando se presenta acarreo desde el *bit* más significativo del resultado.

Figura 1.13. Registro de estados

04h o FSR: Registro selector de registros. En asocio con el registro IND0, se utiliza para seleccionar indirectamente los otros registros disponibles. Mientras que los antecesores del PIC16F84 sólo poseían 5 *bits* activos, en este microcontrolador se poseen los 8 *bits*. Si en el programa no se utilizan llamadas indirectas, este registro se puede utilizar como un registro de propósito general.

Para entender mejor el funcionamiento de este registro veamos un programa simple que borra el contenido de la memoria RAM, empleando direccionamiento indirecto.

```
        MOVLW   20      ;inicializa el puntero en la memoria RAM
        MOVWF   FSR     ;que se va a borrar
NEXT    CLRF    INDO    ;borra el registro indexado (es decir el que está
                        ;siendo direccionado por el FSR)
        INCF    FSR,R   ;incrementa el puntero
        BTFSS   FSR,5   ;pregunta si ya acabó el banco de memoria
        GOTO    NEXT    ;sigue borrando los registros que faltan
continúa .....
```

05h o PORTA: Puerto de Entrada/Salida de 5 bits. Este puerto, al igual que todos sus similares en los PIC, puede leerse o escribirse como si se tratara de un registro cualquiera. El registro que controla el sentido (entrada o salida) de los pines de este puerto está localizado en la página 1, en la posición 85h y se llama TRISA.

06h o PORTB: Puerto de entrada/salida de 8 bits. Al igual que en todos los PIC, este puede leerse o escribirse como si se tratara de un registro cualquiera; algunos de sus pines tienen funciones alternas en la generación de interrupciones. El registro de control para la configuración de la función de sus pines se localiza en la página 1, en la dirección 86h y se llama TRISB.

08h o EEDATA: Registro de datos de la EEPROM. Este registro contiene el dato que se va a escribir en la memoria EEPROM de datos o el que se leyó de ésta.

09h o EEADR: Registro de dirección de la EEPROM. Aquí se mantiene la dirección de la EEPROM de datos que se va a trabajar, bien sea para una operación de lectura o para una de escritura.

0Ah o PCLATH: Registro para la parte alta de la dirección. Este contiene la parte alta del contador de programa y no se puede acceder directamente.

0Bh o INTCON: Registro para el control de interrupciones. Es el encargado del manejo de las interrupciones y contiene los *bits* que se muestran en la figura 1.14.

81h u OPTION: Registro de configuración múltiple. Posee varios *bits* para configurar el preescalador, la interrupción externa, el timer y las características del puerto B. Los *bits* que contiene y las funciones que realiza este registro se muestran en la figura 1.15. El preescalador es compartido entre el MTRO y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.

85h o TRISA: Registro de configuración del puerto A. Como ya se mencionó, es el registro de control para el puerto A. Un "cero" en el *bit* correspondiente al pin lo configura como salida, mientras que un "uno" lo hace como entrada.

86h o TRISB: Registro de configuración del puerto B. Orientado hacia el control del puerto B. Son válidas las mismas consideraciones del registro anterior.

Registro: INTCON

GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

bit 7 bit 0

Dirección: 0Bh
condición de *reset*: 0000000Xb

GIE: *Global Interrupt Enable* o Habilitador general de interrupciones.
0: deshabilita todas las interrupciones
1: habilita las interrupciones

EEIE: *EEPROM Write Interrupt Enable* o Habilitación de interrupción por escritura de la EEPROM
0: la deshabilita
1: la habilita

TOIE: *TMR0 Interrupt Enable* o Habilitación de interrupción del temporizador TMR0.
0: la deshabilita
1: la habilita

INTE: *INT Interrupt Enable* o Habilitación de la interrupción INT.
0: la deshabilita
1: la habilita

RBIE: *RBIF Interrupt Enable* o Habilitación de la interrupción RBIF.
0: la deshabilita
1: la habilita

TOIF: *TMR0 Overflow Interrupt Flag* o Bandera de la interrupción por sobreflujo del TMR0.
Se coloca en 1 cuando el TMR0 pasa de 0FFh a 00h; ésta debe ser puesta a 0 por programa.

INTF: *INT Interrupt Flag* o Bandera de interrupción INT.
Se coloca en 1 cuando la interrupción INT (RB<0>) ocurre; ésta debe ser puesta a 0 por programa.

RBIF: *RB Port Change Interrupt Flag* o Bandera de interrupción por cambio en el puerto B.
Se coloca en 1 cuando una de las entradas RB<7:4> cambia; ésta debe ser puesta a 0 por programa.

Figura 1.14. Registro INTCON

88h o EECON1: Registro para el control de la memoria EEPROM de datos. Este es el registro de control de la memoria de datos y sólo destina cinco *bits* para ello, los más bajos; los tres *bits* superiores permanecen sin implementar. En la figura 1.16 se muestran las funciones de estos *bits*.

89h o EECON2: Registro auxiliar para control de la memoria EEPROM de datos. Registro que no está implementado físicamente en el microcontrolador, pero que es necesario en las operaciones de escritura en la EEPROM de datos; ante cualquier intento de lectura se obtendrán "ceros".

0Ch a 4Fh: Registros de propósito general. Estas 68 posiciones están implementadas en la memoria RAM estática, la cual conforma el área de trabajo del usuario; a ellas también se accede cuando en la página 1 se direccionan las posiciones 8Ch a

Registro: OPTION

$\overline{\text{RBPU}}$	INTEDG	GRTS	RTE	PSA	PS2	PS1	PSO

bit 7 bit 0

Dirección: 81h
condición de *reset*: 1111 1111b

$\overline{\text{RBPU}}$: *PortB Pull-up Enable* o Habilitación de *pull-up* del puerto B.
0: habilita las *pull-ups* internas
1: las deshabilita

INTEDG: *INT Interrupt Edge Select* o Selector de flanco de la interrupción INT
0: flanco de bajada
1: flanco de subida

RTS: *TMR0 Signal Source* o Fuente de señal de TMR0.
0: ciclo de instrucciones interno (Temporizador)
1: transición en el pin RA4/TOCK (Contador)

RTE: *TMR0 Signal Edge* o Flanco de la señal TMR0
0: incremento en transición de bajo a alto
1: incremento en transición de alto a bajo

PSA: *Prescaler Assignment* o Asignación del preescalador
0: TRM0 (Contador/Temporizador)
1: WDT (Circuito de vigilancia)

PS2,1,0: *Prescaler Value* o Valores del preescalador.

Valor	TMR0	WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Figura 1.15. Registro OPTION

CFh. Esto se ha diseñado así para evitar un excesivo cambio de páginas en el manejo de la RAM del usuario, agilizando los procesos que se estén llevando a cabo y descomplicando la labor del programador.

Registro de trabajo W. Este es el registro de trabajo principal, se comporta de manera similar al acumulador en los microprocesadores. Este registro participa en la mayoría de las instrucciones.

Pila (Stack)

Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina o cuando se atiende una interrupción; luego, cuando el micro regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila. El PIC16F84 tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problemas.

Registro: EECON1

U	U	U	EEIF	WRERR	WREN	WR	RD
bit 7							bit 0

Dirección: 88h
condición de *reset*: 0000X000b

U: *Unimplemented* No implementadas. Estos *bits* se leen como ceros.
EEIF: *EEPROM Write Completion Interrupt Flag* o Bandera de finalización de la escritura. Se coloca en "1" cuando finaliza con éxito la escritura en la EEPROM de datos; se debe colocar en "0" por programa. El bit de habilitación correspondiente es el EEIE, localizado en el registro INTCON (0B<6>).
WRERR: *Write Error Flag* o Bandera de error de escritura. Se coloca en 1 cuando la operación de escritura termina prematuramente, debido a cualquier condición de *reset*.
WREN: *Write Enable* o habilitación de escritura. Si se coloca en "0" no permite las operaciones de escritura; en "1" las habilita.
WR: *Write Control* o Control de escritura. Al colocarse en "1" inicia un ciclo de escritura. Este bit sólo es puesto a "0" por *hardware*, una vez la escritura termina.
RD: *Read Control* o Control de lectura. Al colocarse en "1" se inicia una lectura de la EEPROM de datos, la cual toma un ciclo del reloj de instrucciones. Este bit sólo se limpia (se coloca en '0") por *hardware*, al finalizar la lectura de la posición de la EEPROM.

Figura 1.16. Registro EECON1

Características especiales

Algunos elementos que forman parte de los PIC no se encuentran en microcontroladores de otros fabricantes, o simplemente representan alguna ventaja o facilidad a la hora de hacer un diseño. Veamos una breve descripción de las más significativas:

Circuito de vigilancia (*Watchdog Timer*)

Su función es restablecer el programa cuando éste se ha perdido por fallas en la programación o por alguna razón externa. Es muy útil cuando se trabaja en ambientes con mucha interferencia o ruido electromagnético. Está conformado por un oscilador RC que se encuentra dentro del microcontrolador.

Este oscilador corre de manera independiente al oscilador principal. Cuando se habilita su funcionamiento, dicho circuito hace que el microcontrolador sufra un reset cada determinado tiempo (que se puede programar entre 18 ms y 2 segundos). Este reset lo puede evitar el usuario mediante una instrucción especial del microcontrolador (**CLRWDT**: borrar el conteo del watchdog), la cual se debe ejecutar antes de que termine el período nominal de dicho temporizador. De esta manera, si el programa se

ha salido de su flujo normal, por algún ruido o interferencia externa, el sistema se reiniciará (cuando se acabe el tiempo programado y no se haya borrado el contador) y el programa puede restablecerse para continuar con su funcionamiento normal.

En las primeras prácticas no se utiliza el circuito de vigilancia para facilitar el trabajo; por eso, en el momento de programar el microcontrolador se debe seleccionar en los fusibles de configuración "watchdog timer OFF". Más adelante veremos algunos ejemplos que ilustran su funcionamiento y la manera de utilizarlo.

Temporizador de encendido (*Power-up Timer*)

Este proporciona un reset al microcontrolador en el momento de conectar la fuente de alimentación, lo que garantiza un arranque correcto del sistema. En el momento de grabar el micro se debe habilitar el fusible de configuración "*Power-up Timer*", para ello se debe seleccionar la opción "ON". Su tiempo de retardo es de 72 milisegundos.

Modo de bajo consumo (*sleep*)

Esta característica permite que el microcontrolador entre en un estado pasivo donde consume muy poca potencia. Cuando se entra en este modo el oscilador principal se detiene, pero el temporizador del circuito de vigilancia (watchdog) se reinicia y empieza su conteo nuevamente. Se entra en ese estado por la ejecución de una instrucción especial (llamada **SLEEP**) y se sale de él por alguna de las siguientes causas: cuando el microcontrolador sufre un reset por un pulso en el pin MCLR, porque el watchdog hace que se reinicie el sistema o porque ocurre una interrupción al sistema.

Interrupciones

Este microcontrolador incluye el manejo de interrupciones, lo cual representa grandes ventajas. El PIC16F84 posee cuatro fuentes de interrupción a saber:

- Interrupción externa en el pin RB0/INT
- Finalización del temporizador/contador TMRO
- Finalización de escritura en la EEPROM de datos
- Cambio de nivel en los pines RB4 a RB7

El registro 0Bh o INTCON contiene las banderas de las interrupciones INT, cambio en el puerto B y finalización del conteo del TMRO, al igual que el control para habilitar o deshabilitar cada una de las fuentes de interrupción, incluida la de escritura en memoria EEPROM. Sólo la bandera de finalización de la escritura reside en el registro 88h (EECON1<4>).

Si el *bit* GIE (*Global Interrupt Enable*) se coloca en 0, deshabilita todas las interrupciones. Cuando una interrupción es atendida, el *bit* GIE se coloca en 0 automáticamente para evitar interferencias con otras interrupciones que se pudieran presentar, la dirección de retorno se coloca en la pila y el PC se carga con la dirección 04h. Una vez en la rutina de servicio, la fuente de la interrupción se puede determinar examinando las banderas de interrupción. La bandera respectiva se debe colocar, por *software*, en cero antes de regresar de la interrupción, para evitar que se vuelva a detectar nuevamente la misma interrupción.

La instrucción RETFIE permite al usuario retornar de la interrupción, a la vez que habilita de nuevo las interrupciones, al colocar el *bit* GIE en uno. Debe tenerse presente que solamente el contador de programa es puesto en la pila al atenderse la interrupción; por lo tanto, es conveniente que el programador tenga cuidado con el registro de estados y el de trabajo, ya que se pueden producir resultados inesperados si dentro de ella se modifican.

Interrupción externa. Actúa sobre el pin RB0/INT y se puede configurar para activarse con el flanco de subida o el de bajada, de acuerdo al *bit* INTEDG (OPTION<6>). Cuando se presenta un flanco válido en el pin INT, la bandera INTF (INTCON<1>) se coloca en uno. La interrupción se puede deshabilitar colocando el *bit* de control INTE (INTCON<4>) en cero. Cuando se atiende la interrupción, a través de la rutina de servicio, INTF se debe colocar en cero antes de regresar al programa principal. La interrupción puede reactivar al microcontrolador después de la instrucción SLEEP, si previamente el *bit* INTE fue habilitado.

Interrupción por finalización de la temporización. La superación del conteo máximo (0FFh) en el TMR0 colocará el *bit* TOIF en uno (INTCON<2>). El *bit* de control respectivo es TOIE (INTCON<5>).

Interrupción por cambio en el puerto RB. Un cambio en los pines del puerto B <7:4> colocará en uno el *bit* RBIF (INTCON<0>). El *bit* de control respectivo es RBIE (INTCON<3>).

Interrupción por finalización de la escritura. Cuando la escritura de un dato en la EEPROM finaliza, se coloca en 1 el *bit* EEIF (EECON1<4>). El *bit* de control respectivo es EEIE (INTCON<6>)

Memoria de datos EEPROM

El PIC16F84 tiene una memoria EEPROM de datos de 64 posiciones (0h a 3Fh), de 8 *bits* cada una. Este bloque de memoria no se encuentra mapeado en ningún banco, el acceso a esas posiciones se consigue a través de dos registros de la RAM:

- el registro EEADR (posición 09), que debe contener la dirección de la posición de la EEPROM a ser accesada
- el registro EEDATA (posición 08), que contiene el dato de 8 *bits* que se va a escribir o el que se obtuvo de la última lectura.

Adicionalmente, existen dos registros de control: el EECON1 (88h), que posee cinco *bits* que manejan las operaciones de lectura/escritura y el EECON2 (89h), que aunque no es un registro físico, es necesario para realizar las operaciones de escritura.

La lectura toma un ciclo del reloj de instrucciones, mientras que la escritura, por ser controlada por un temporizador incorporado, tiene un tiempo nominal de 10 milisegundos, este tiempo puede variar con la temperatura y el voltaje. Cuando se va a realizar una operación de escritura, automáticamente se hace primero la operación de borrado. El número típico de ciclos de borrado/escritura de la EEPROM de datos es de 1.000.000.

Fusibles de configuración

El PIC16F84 posee cinco fusibles, cada uno de los cuales es un *bit*. Estos fusibles se pueden programar para seleccionar varias configuraciones del dispositivo: tipo de oscilador, protección de código, habilitación del circuito de vigilancia y el temporizador al encendido. Los *bits* se localizan en la posición de memoria 2007h, posición a la cual el usuario sólo tiene acceso durante la programación del microcontrolador. Cuando se programa la protección de código, el contenido de cada posición de la memoria no se puede leer completamente, de tal manera que el código del programa no se puede reconstruir. Adicionalmente, todas las posiciones de memoria del programa se protegen contra la reprogramación.

Una vez protegido el código, el fusible de protección sólo puede ser borrado (puesto a 1) si se borra toda la memoria del programa y la de datos.

Las pull-ups internas

Cada uno de los pines del puerto B tiene un débil elemento *pull-up* interno (250 µA típico); este elemento es automáticamente desconectado cuando el pin se configura como salida. Adicionalmente, el *bit* RBPU (OPTION<7>) controla todos estos elementos, los cuales están deshabilitados ante una condición de *reset*. Estos elementos *pull-up* son especialmente útiles cuando el microcontrolador va a colocarse en el modo de bajo consumo, ya que ayudan a no tener las entradas flotantes, significando una reducción en el consumo de corriente.

El conjunto de instrucciones

Estas se clasifican en orientadas a registros, orientadas al *bit* y operaciones literales y de control. Cada instrucción es una palabra de 14 *bits*, dividida en un código de operación (el cual especifica la orden a ejecutar) y uno o más operandos sobre los que se actúa. En el apéndice A se encuentra la lista completa de instrucciones, la cual incluye ejemplos y explicaciones. Como se puede observar allí, en total son 35, las cuales tardan un ciclo de máquina, a excepción de los saltos, que toman dos ciclos.

El PIC16C84

El PIC16C84 es un microcontrolador de la familia Microchip, totalmente compatible con el PIC16F84. Su principal característica es que posee memoria "EEPROM" en lugar de memoria *Flash*, pero su manejo es igual. Con respecto al PIC16F84, este microcontrolador presenta dos diferencias:

- La memoria de datos tiene menor tamaño, aquí se tienen 32 registros de propósito general (el mapa de memoria de datos llega hasta 2FH).
- En el momento de programar el microcontrolador, el fusible de selección del temporizador de arranque (*Power Up Timer*) trabaja de forma inversa, es decir, si en el PIC16F84 se selecciona la opción "Low" para activarlo, en el PIC16C84 se debe seleccionar "High".

Este microcontrolador ha sido reemplazado de forma gradual por el PIC16F84, por lo tanto, los diseños que lo utilicen como elemento de control deben ser actualizados. Aunque, como se ve, es un proceso casi transparente.