Escuela Politécnica Superior
Universidad de Huelva

TERCER CURSO. INFORMÁTICA INDUSTRIAL II

Introducción a la Programación de PIC de Microchip

Manuel Sánchez Raya
Versión 0.9
24 de Febrero de 2002

ÍNDICE

BIBLIOGRAFÍA:

Microcontroladores de Bajo Costo. Familia PIC 15C5X
Antonio Bueno Juan. Editorial Ediatec

Curso de programación de PIC. Revista Delek

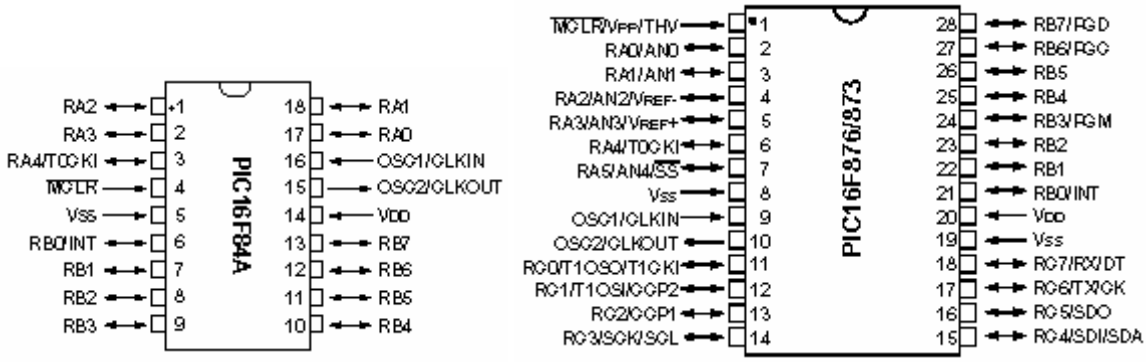
1.- Introducción.

La familia de microcontroladores PIC 16C5x es un producto de la sociedad Microchip Technology Inc. Radicada en Chandler, Arizona (USA). Esta familia de dispositivos consta de una serie de microcontroladores CMOS con memorias de datos y programa internas. La memoria de programa tiene un tamaño de palabra mínimo de 12 bits, lo cual obviamente hace que estos microcontroladores sean mucho más potentes que sus competidores de 8 bits.

El tamaño de la memoria de datos y de la memoria de programa depende del tipo de controlador. El diseño totalmente estático de los microcontroladores permite reducir la frecuencia del reloj hasta el valor de la continua. La ventaja de tener un tamaño de palabra de 12 bits radica en que la mayoría de las instrucciones sólo requieren una palabra, incluyendo sus operandos.

El controlador posee 33 instrucciones todas ellas muy fáciles de recordar. Con excepción de los saltos (directos o indirectos), todas las instrucciones invierten un ciclo máquina. En consecuencia, Microchip anuncia los PIC16C5x como un dispositivo que emplea una arquitectura tipo RISC (arquitectura de procesador con juego reducido de instrucciones), la cual garantiza una compacta pero rápida secuencia de instrucciones cada una de las cuales se ejecuta en un único ciclo máquina.

Además de las diferencias en el tamaño de la memoria, los controladores PIC tienen disponibles diferentes números de líneas de E/S y diferentes tipos de osciladores de reloj (frecuencia). Además, los dispositivos están disponibles en distintos encapsulados, dos de los cuales se muestran en la figura.



Comparando los distintos encapsulados, la característica más curiosa es la presencia o ausencia de ventana de cristal. Los modelos sin ventana, y por tanto más baratos, son idóneos para grandes volúmenes de producción. Las versiones con ventana contienen una EPROM y son ideales para desarrollar aplicaciones, dado que su memoria de programa se puede borrar utilizando luz ultravioleta.

Todos los dispositivos de la familia disponen de bit de protección anticopia. Cuando se programa, este bit hace imposible la lectura del código de un controlador programado. Existen también versiones más modernas que emplean memoria de tipo FLASH que tiene la propiedad de borrarse de forma eléctrica cuantas veces se desee y que hace más sencillo el desarrollo de programas empleando estos dispositivos.

2.- Miembros de la familia.

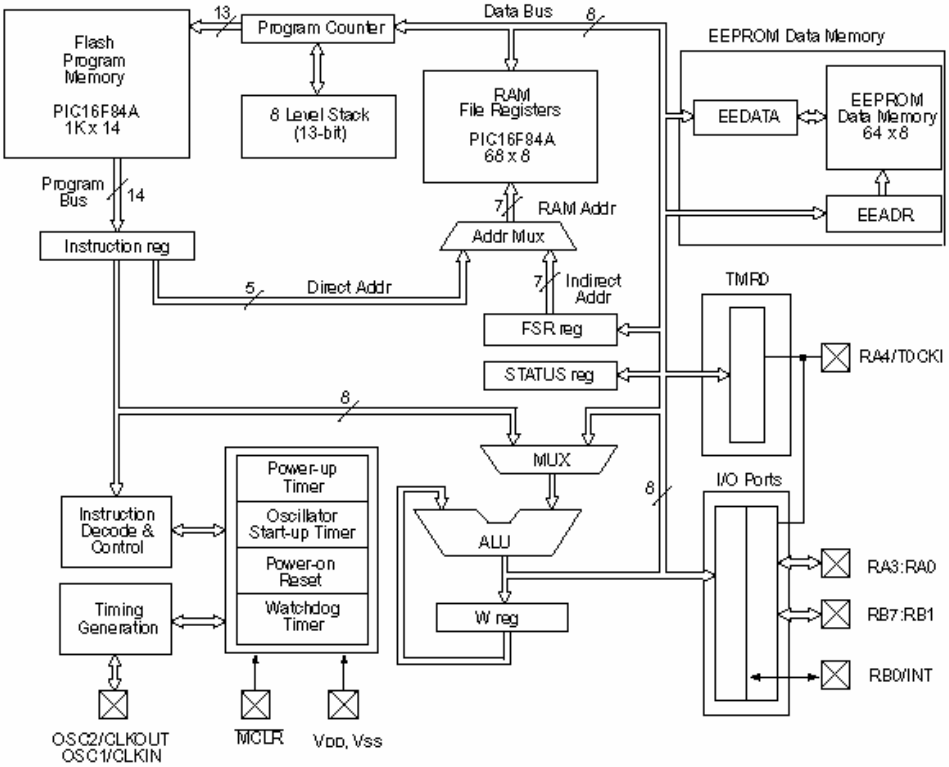
En la tabla siguiente se pueden ver varios miembros, siendo los más usados los que integran memoria FLASH que tienen la letra F en su código.

Tipo	Fmax	Programa	Datos	EEPROM	Timers	E/S	Vcc	Inst.
12C508	4 MHz	512x12	25	-	1+WDT	6	2-5.5	33
12C509	4 MHz	1024x12	41	-	1+WDT	6	2-5.5	33
16C54	20 MHz	512x12	25	-	1+WDT	12	2-6.5	
16C55	20 MHz	1024x12	25	-	1+WDT	20	2-6.5	
16C56	20 MHz	1024x12	25	-	1+WDT	12	2-6.5	
16F84	20 MHz	1024x14	68	64	1+WDT	13	2-5.5	35
16F873	20 MHz	4096x14	192	128	3+2	22	2-5.5	35
16F874	20 MHz	4096x14	192	128	3+2	33	2-5.5	35
16F876	20 MHz	8192x14	368	256	3+2	22	2-5.5	35
16F877	20 MHz	8192x14	368	256	3+2	33	2-5.5	35
17C42	25 MHz	2048x16	232	-	3+2	33	4-5.5	55
17C43	25 MHz	4096x16	454	-	3+2	33	4-5.5	55

3.- Hardware: estructura interna del PIC

La arquitectura del PIC se basa en el concepto de fichero de registros con bus y memorias independientes para datos e instrucciones. Este tipo de arquitectura o estructura interna, también denominada arquitectura Harvard, permite al procesador poder ejecutar una instrucción y al mismo tiempo "capturar" de la memoria de programa la siguiente instrucción a ejecutar. Desafortunadamente, esta característica no se puede utilizar con una serie de instrucciones de salto y, en consecuencia, estas instrucciones invierten dos ciclos máquina en lugar de uno solo.

Nos centraremos en el PIC16F84, que es el más usado. La arquitectura del controlador se muestra en la figura siguiente.

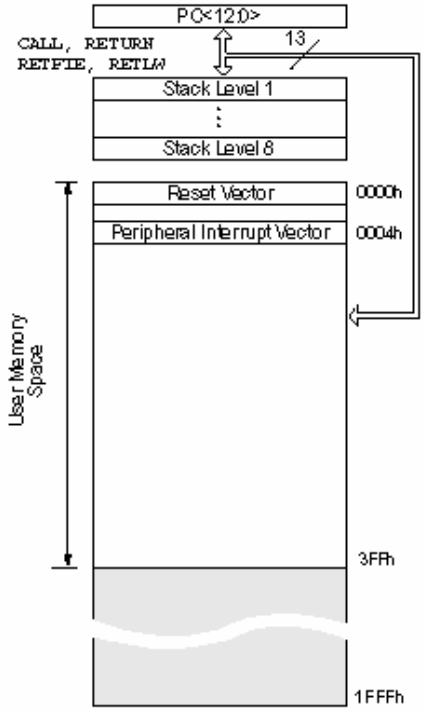


La memoria de programa junto con su contador asociado, el registro de instrucción y el decodificador de instrucciones se pueden observar en la esquina superior izquierda del diagrama. Debajo de estos bloques podemos observar la unidad aritmética lógica o ALU junto con su registro de trabajo asociado (identificado como 'W' en el diagrama). Desde la parte central y hacia la derecha encontramos la memoria de datos (fichero de registros de propósito general) y encima de ésta se representa el generador de reloj con el circuito de supervisión tipo perro guardián asociado y el registro de entrada y salida (E/S).

3.1.- Memoria de programa

La memoria de programa consta de páginas de 512 palabras cada una. El PIC16C54 y el 16C55 sólo tienen una de estas páginas mientras que el PICl6C56 y el PIC16F84 tiene dos. El PIC16F876 y demás disponen de varias páginas de memoria. El bit de dirección A9 se utiliza para seleccionar las dos páginas. El PIC 16C57 tiene cuatro páginas, que se seleccionan con la ayuda de los bits A9 y A10.

La estructura de la memoria de programa se ilustra en la figura siguiente.



Tipo	Bits del PC
16C54	9
16C55	9
16C56	10
16F84	13
16F873	13
16F874	13
16F876	13
16F877	13

La secuencia de ejecución de las microinstrucciones se controla a través del denominado contador de programa (PC) utilizando direcciones de 9, 11 o 13 bits dependiendo del micro. Cuando se ejecutan instrucciones de salto, o cuando se invoca a una subrutina se deben tomar algunas precauciones. Los problemas pueden surgir cuando se asuma (error típico en la programación de estos dispositivos) que los datos se codifican en 8 bits, lo cual es imposible con operandos que tienen una anchura de 9 bits o más.

Con los procesadores 16C5X, el programador debía tener en cuenta la restricción de anchura de palabra, con lo que las instrucciones que alteran el flujo del programa SOLO se podían utilizar en la mitad inferior del bloque de memoria (A8=0) y se debían programar los bits 5 y 6 del registro STATUS para cambiar de página. Esto ya no es necesario con el PIC16F84 o con el PIC16F876, puesto que en este caso el contador de programa es de 13 bits y con él podemos saltar a cualquier posición de memoria de código.

3.2.- Memoria de datos y fichero de registros

Microchip Technology denomina a la memoria interna de los PIC fichero de registros (RF) dado que, para ellos, un registro es una posición de memoria que se puede acceder directamente por la ALU. Con este concepto, es fácil concebir que un conjunto de registros se denomine entonces como un fichero.

La estructura del fichero de registros se muestra en la figura siguiente.

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.

3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device reset, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

Todos los registros se pueden incluir en una de las siguientes categorías:

- registros de operación.
- registros de entrada/salida.
- registros de propósito general.

Los registros de operación no sirven sólo para controlar el núcleo del PIC sino que también proporcionan resultados de ciertas acciones disponibles en el programa. Los registros de entrada y salida se utilizan para ganar accesibilidad en los puertos de entrada y salida utilizados para comunicarse con el mundo exterior al PIC. Por último, los registros de propósito general se utilizan para almacenar datos. A nivel del usuario, todos los registros se comportan de la misma forma. Veamos con todo detalle cada uno de estos registros.

3.3.- Registros de operación

Los PIC disponen de 5 registros de operación:

- Registro INDF, Direccionamiento indirecto de datos.
- Registro TMR0, registro del contador/reloj en tiempo real
- Registro PCL, registro contador de programa (8 bits bajos solo)
- Registro STATUS, Registro de la palabra de estado
- Registro FSR, registro de selección de fichero

3.3.1.- Registro INDF.

Este registro permite al programador un acceso indirecto a los datos. El acceso se obtiene en conjunción con el registro FSR descrito mas adelante. Un comando de lectura o escritura en el registro INDF provoca que la ALU seleccione la posición cuya dirección es igual al contenido del registro FSR. La combinación del registro INDF y del registro FSR crea múltiples formas de manejar áreas de memoria de manera muy eficiente.

En el trivial caso de una lectura del registro INDF a través de un direccionamiento indirecto (FSR contiene entonces el valor 00h), se lee el valor 00h. Si se escribe el registro INDF a través de un direccionamiento indirecto, el resultado es un NOP (no operación).

3.3.2.- Registro TMR0.

Este registro se puede comparar a una posición de la memoria de datos. El contenido de esta posición se incrementa continuamente utilizando una señal de reloj. La señal de reloj puede ser externa (aplicada a través de la entrada RTCC) o interna, derivada entonces del reloj del ciclo de instrucción del controlador. Un predivisor interno permite dividir la señal de reloj según las necesidades de cada aplicación. Más detalles sobre esta función se incluyen en la descripción del temporizador tipo perro guardián.

3.3.3.- Registro PCL.

El contador de programa genera las direcciones de las posiciones de la memoria de programa. Dependiendo del tipo de dispositivo, el contador de programa y su pila hardware de dos niveles en el caso del 16C54 o de ocho en el caso del 16F84 asociada tienen una anchura de 9, 10, 11 o 13 bits, utilizando los bits A8, A9, A10, A11 y A12 como extensión de los A0 a A7.

Normalmente, el contador de programa se incrementa en una unidad después de la ejecución de cada instrucción. Sin embargo, en el 16C56/57 las siguientes instrucciones hacen que el contador de programa tenga un comportamiento distinto:

- GOTO Los bits A0-A8 se cargan directamente, mientras que el A9 y el Al0 se toman de los bits 5 y 6 del registro de estado (PA0, PA1). Cuando se utilicen el PIC16C56 y el 16C57, el registro de estado se debe cargar apropiadamente.
- CALL La instrucción CALL difiere de la GOTO en que el bit 8 siempre se pone a 0. En consecuencia, el rango de direccionamiento de esta instrucción es limitado.

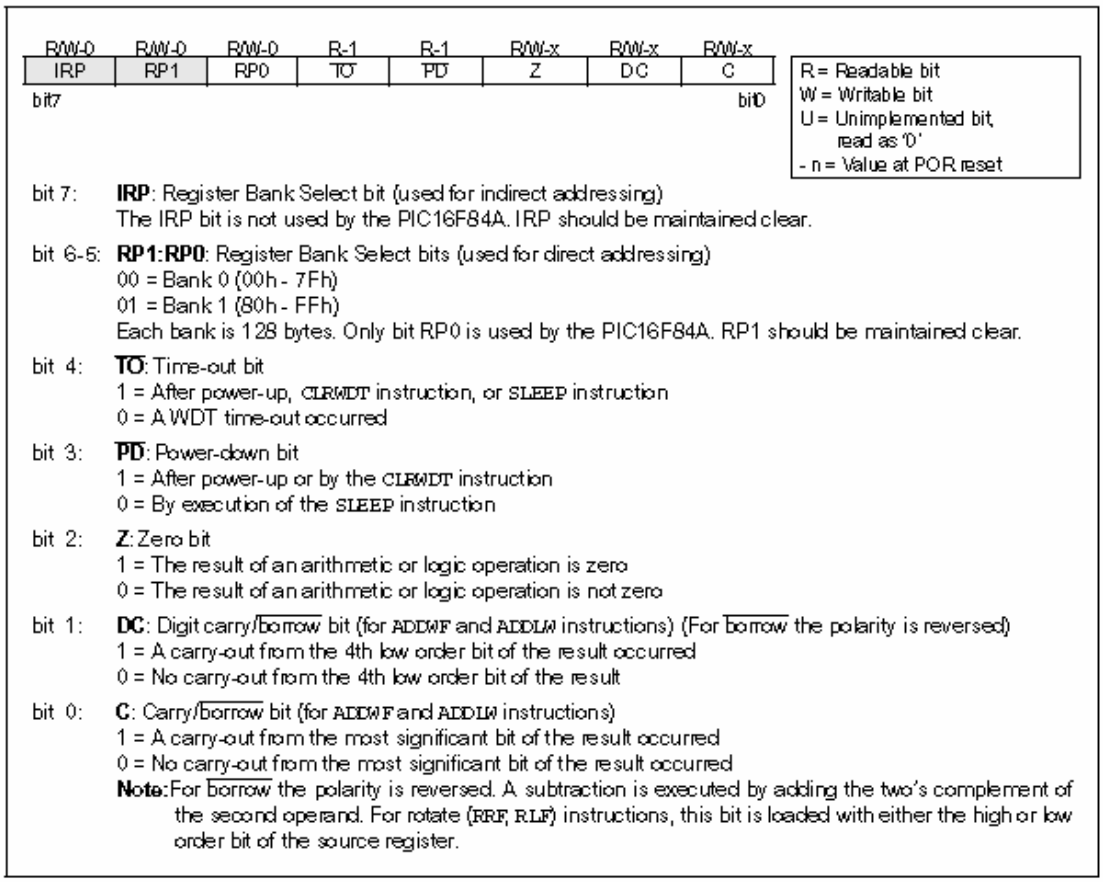
El resto de instrucciones de escritura en el PC funcionan de la misma forma que la instrucción CALL, esto es, borran el bit A8 y leen el A9 y el Al0 del registro de estado (sólo en el PIC16C56/57).

En el caso del 16F84 y del 16F876 no hay ningún problema porque el PC es de 13 bits y los bits 5 y 6 del registro de estado se emplean para cambiar de banco de memoria de datos al realizar direccionamiento directo.

3.3.4.- Registro STATUS.

Los indicadores C, D y Z (bits 0, 1 y 2) de este registro proporcionan el estado de una operación aritmética realizada por la ALU. El indicador PD (bit 3) y el T0 (bit 4) indican el estado de inicialización (reset), mientras que el PA0 (bit 5) y el PA1 (bit 6) se utilizan como bits auxiliares (A9 y A10) para ciertas operaciones del contador de programa (ver instrucciones CALL y GOTO descritas anteriormente) en el 16C54 o bien como selección de banco de memoria RAM para el 16F84 y F876.

Los bits 3 y 4 (T0 y PD) del registro de la palabra de estado no están afectados por una operación de escritura en 8 bits.



R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	TO	PD	Z	DC	C
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **IRP**: Register Bank Select bit (used for indirect addressing)
The IRP bit is not used by the PIC16F84A. IRP should be maintained clear.

bit 6-5: **RP1:RP0**: Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
Each bank is 128 bytes. Only bit RP0 is used by the PIC16F84A. RP1 should be maintained clear.

bit 4: **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3: **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2: **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC**: Digit carry/$\overline{\text{borrow}}$ bit (for ADDWF and ADDLW instructions) (For $\overline{\text{borrow}}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0: **C**: Carry/$\overline{\text{borrow}}$ bit (for ADDWF and ADDLW instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
Note: For $\overline{\text{borrow}}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

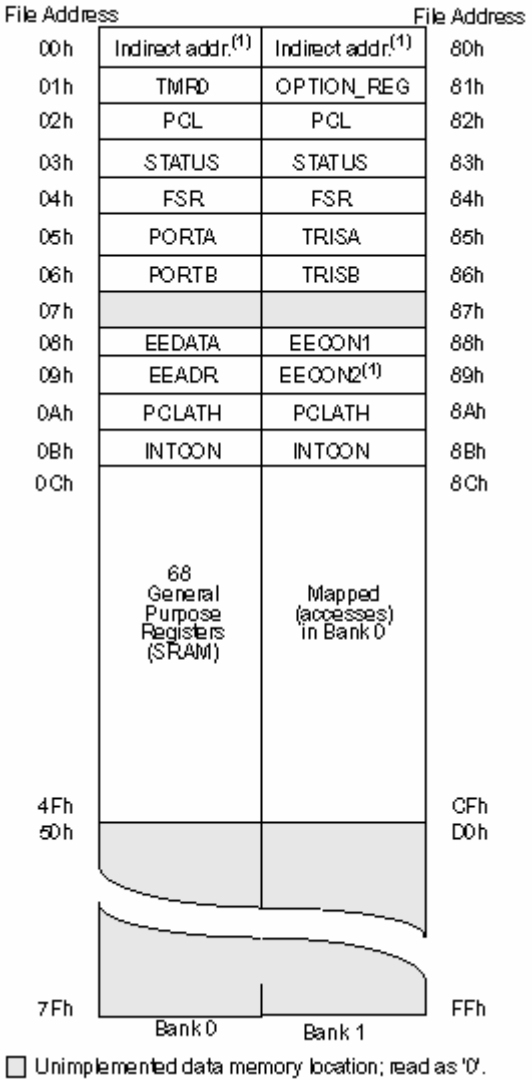
Si este registro se utiliza para indicar el resultado de una operación aritmética debe tenerse en cuenta que los bits de estado se ponen a uno después de la siguiente escritura. Se recomienda utilizar sólo las instrucciones BCF, BSF y MOVWF para alterar el registro de estado, dado que éstas no afectan a ningún bit de estado.

Función	TO	PD
Encendido	1	1
Fin de tiempo WDT	0	X
Instrucción SLEEP	1	0
Instrucción CLRWDT	1	1

Inicialización por	TO	PD
Fin modo SLEEP para WDT	0	0
Fin tiempo WDT (no durante SLEEP)	0	1
Fin modo SLEEP inicialización externa	1	0
Encendido	1	1
Nivel 0 en entrada MCLR	X	X

3.3.5.- Registro FSR.

En el PIC16C54/55/56 los bits 0 a 4 seleccionan uno de los 32 registros disponibles en el modo de direccionamiento indirecto, esto es, utilizando el registro INDF. Los bits 5,6 y 7 son sólo de lectura y siempre están a 1. Si no se utiliza el direccionamiento indirecto, el FSR se puede utilizar como un registro de propósito general de 5 bits de anchura.



Para PIC16C57 los bits 5 y 6 seleccionan el banco de memoria en curso, tanto en los modos de direccionamiento directo como indirecto. Téngase presente que esto sólo es válido para las direcciones de registro 10h a 1Fh, dado que las direcciones 00h a 0Fh siempre apuntan al mismo registro. Por último, el bit 7 siempre está a 1.

En el caso de 16F84 puede seleccionar cualquier dirección comprendida para el banco 0 de 00h a 4Fh, y para el banco 1 de 80h a CFh, si bien como indicaba la tabla anterior se produce una duplicación de ciertos registros entre cada banco de memoria.

3.4.- Registros de gestión de la pila.

Los dispositivos de la familia PIC16C5x (16F8x) disponen de dos (ocho) registros de pila que se caracterizan por su utilización desde el punto de vista hardware.

Una instrucción CALL sitúa el valor del contador de programa en curso, incrementado en una unidad, en el nivel 1 de la pila. A continuación, el nivel 1 de la pila se envía al nivel 2. Esta característica permite que el programa “salte” dos veces de su flujo normal. Si se anidan más llamadas a subrutinas, las direcciones de regreso se pierden.

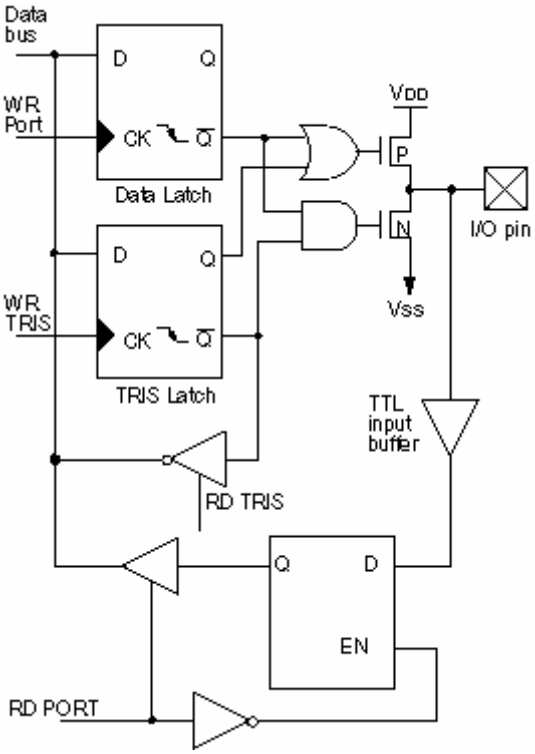
Una instrucción RETLW carga de nuevo el contenido del nivel 1 de la pila en el contador de programa, mientras que el contenido del nivel 2 de la pila se copia en el nivel 1 de la pila. El nivel 2 de la pila retiene su valor.

3.5.- Registros de E/S

Los registros de entrada/salida o puertos constituyen la interfaz entre el software y los dispositivos hardware externos. Estos registros pueden leerse o escribirse. Una instrucción de lectura siempre lee el estado de un terminal, independientemente de si éste está programado como entrada o salida. Después de una inicialización, todos los terminales se definen como entradas (modo de alta impedancia).

A continuación, se puede utilizar la instrucción TRIS para definir los terminales como salida. Un ‘0’ en el registro TRIS es necesario para asignar la función salida a un determinado terminal

de E/S. Una vez que un terminal se ha definido como salida, éste se conmuta al nivel indicado por el bit asociado del registro. En consecuencia, el programador debe asegurar que estos bits quedan programados en un nivel de seguridad antes de que se conmuten a la función de salida. El circuito equivalente de un terminal de E/S se muestra en la figura siguiente.



El puerto A se programa a través del registro PORTA. Sólo se utilizan los cuatro bits de orden inferior, RA0-RA3. Los bits RA4-RA7 no están implementados y se leen cero. Para asegurar que una aplicación software será compatible con versiones posteriores y mas avanzadas del controlador, estos bits se deben tratar como no definidos. Los puertos B y C se programan a través de los registros PORTB y PORTC respectivamente, que por cierto no se encuentra implementado en el 16F84.

3.6.- Registros de propósito general.

Para el 16C5X el banco de registros deseado se selecciona utilizando los bits 5 y 6 del registro FSR. Dado que las instrucciones disponen de hasta 5 bits para direccionamiento, la dirección del registro se tiene que convertir. Para el 16F84 esto no es necesario, pues las instrucciones se codifican con dos bits más.

3.7.- Registros de propósito especial.

3.7.1.- Registro W.

El registro de trabajo, W (del inglés “working”) se puede comparar funcionalmente hablando con el acumulador utilizado en la mayoría de los microprocesadores. Una función especial del registro W en los procesadores es que éste no siempre tiene que ser el destino de una operación de la ALU.

3.7.2.- Registros TRIS.

Los registros TRIS (de tri-estado) se utilizan para configurar los terminales de E/S. Cada terminal esta representado por una configuración de bits en el respectivo registro TRIS. Un ‘1’ en la correspondiente posición del bit provoca que el terminal asociado funcione como una salida. La instrucción TRIS sólo tiene función de escritura. Después de una inicialización, todos los bits están a ‘l’, por lo que todos los terminales se comportan como entradas.

3.7.3.- Registro OPTION.

Este registro se utiliza para programar el predivisor del temporizador perro guardián o el del Reloj/temporizador de tiempo real. El registro de opción tiene una anchura de 6 bits y es del tipo de sólo escritura utilizando la instrucción OPTION. Las funciones de cada uno de los bits en particular se puede encontrar en la tabla siguiente. En este caso, también todos los bits se sitúan a ‘ 1’ después de una inicialización.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **RBPU**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled (by individual port latch values)

bit 6: **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5: **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4: **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3: **PSA**: Prescaler Assignment bit
1 = Prescaler assigned to the WDT
0 = Prescaler assigned to TMR0

bit 2-0: **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
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

3.11.- Temporizador perro guardián

El temporizador perro guardián (referido como WDT en la documentación de Microchip) es un contador especial que se programa para inicializar un programa cuando se produce una condición de desbordamiento en el contador (esto es, cuando finaliza el tiempo programado en el temporizador). Mientras que un programa funciona normalmente, el software inicializa el contador antes de que se produzca un desbordamiento.

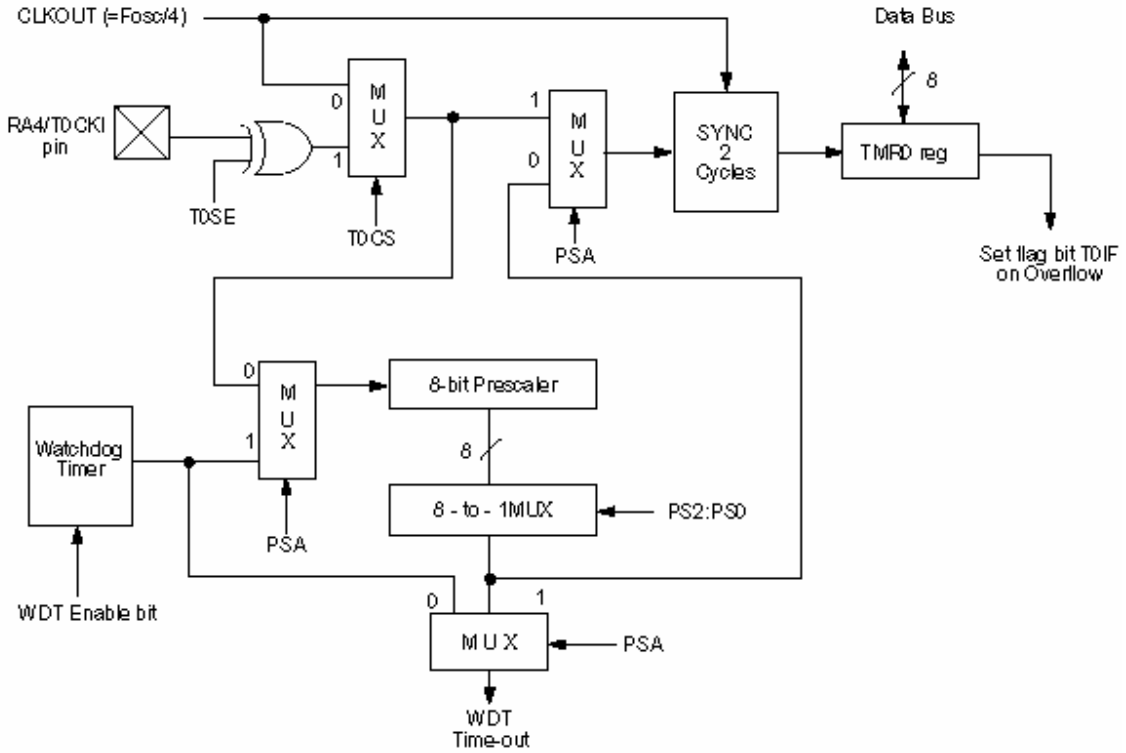
Si el programa deja de funcionar, el contador no se vuelve a inicializar y, en consecuencia, el WDT genera un desbordamiento que provoca la inicialización del programa para que éste vuelva a comenzar desde el principio. Este temporizador es muy útil en sistemas de control automáticos donde los fallos pueden desencadenar situaciones críticas. Obviamente, el programador debe asegurar que el temporizador se inicializa en una posición fija del programa. El WDT no se puede inicializar en una rutina de atención a una interrupción.

El tiempo que transcurre antes de que el temporizador genera un desbordamiento debe estar comprendido entre 9 y 30 ms. Los pulsos de reloj necesarios para que el contador funcione se aplican desde un oscilador RC interno. Si se requiere un tiempo mayor, se puede utilizar el predivisor interno. El valor de predivisión se puede programar a través del registro OPTION.

El temporizador perro guardián se habilita a través de un bit del registro OPTION. Una vez habilitado, el temporizador no se puede deshabilitar por ninguna instrucción. La única forma de parar el temporizador es eliminando el reloj del sistema. El WDT y su predivisor asociado sólo pueden inicializarse por medio de las instrucciones CLRWDT y SLEEP.

3.12.- Predivisor de doble función

El procesador PIC dispone de un contador de 8 bits que funciona como un predivisor y se puede asignar al temporizador perro guardián (WDT) o al reloj/temporizador de tiempo real (RTCC) programando el bit PSA del registro OPTION. De hecho, el predivisor es un autentico postdivisor para el temporizador perro guardián. Sin embargo, por razones de simplicidad, nos referiremos a él también como predivisor en la siguiente explicación.



El valor de predivisión se programa a través de los bits PS0, PS1 y PS2 que también se encuentran en el registro OPTION. Si el predivisor se asigna al reloj de tiempo real, éste se pondrá a cero en cada acción de escritura. Si el predivisor se asigna al temporizador perro guardián, se pondrá a cero por medio de la instrucción CLRWDT.

Puesto que la función del contador viene determinada por el software, la configuración se puede cambiar en cualquier momento durante la ejecución del programa. La asignación WDT/RTCC es mutuamente exclusiva. Para evitar una inicialización del dispositivo no deseada, Microchip sugiere la secuencia de instrucciones listada en el programa siguiente, cuando se cambie la asignación del predivisor del RTCC al WDT.

```
MOVLW xx0x0xxxB   ;selección reloj interno y nuevo
                  ;valor de predivisión, si nuevo valor
OPTION            ;es 000 ó 001, entonces selección de otro
                  ;valor temporalmente
CLRF  RTCC        ;Poner a cero RTCC y predivisor
MOVLW xxxx1xxxB   ;Selección WDT, no cambio predivisión
OPTION
CLRWDT            ;Poner a cero predivisor y WDT
MOVLW xxxx1xxxB   ;Selección nuevo factor predivisión
OPTION
CLRWDT            ;Poner a cero WDT y predivisor
MOVLW xxxx1xxxB   ;Selección nuevo valor predivisión
```

Los dos primeros pasos sólo son necesarios si se utiliza una fuente de reloj externa para el RTCC. Los últimos dos pasos sólo son necesarios si el valor de predivisión deseado es 000 ó

001. Para cambiar el predivisor del WDT al RTCC se recomienda la secuencia listada en el programa siguiente.

```
CLRWDT            ;Poner a cero predivisor y WDT
MOVLW xxxx0xxxB   ;selección RTCC, nuevo valor predivisión
OPTION            ;fuente de reloj
```

3.13.- RTCC: reloj/temporizador de tiempo real

El reloj/temporizador de tiempo real en el controlador PIC toma forma de un contador de 8 bits cuyo contenido (estado) se puede leer modificar en cualquier instante. El RTCC básicamente es una posición de memoria normal. Una vez que se alcanza el estado máximo (FFh), el contador pasa automáticamente al estado 00h. El contador tiene dos posibles fuentes de reloj: (1) una señal interna, que se corresponde con un cuarto de la frecuencia del cristal (fosc/4), o (2) una señal de reloj externa que se aplica a la entrada RTCC del controlador.

Cuando se utiliza reloj externo, el flanco de la señal que incrementa el reloj se puede definir por software. El predivisor se puede conectar "delante" del contador para evitar que el contador se ponga a cero transcurridos 256 pulsos de reloj. Recuérdese que el predivisor se puede asignar bien al temporizador perro guardián 0 al reloj/temporizador de tiempo real, pero NO ambos contadores al mismo tiempo.

El valor de predivisión se programa a través del registro OPTION. Cuando se utiliza el RTCC, los programadores deben tener en cuenta que los pulsos de reloj se retrasan dos ciclos maquina por la unidad de sincronización. En consecuencia, una instrucción de escritura en el registro del RTCC no se reconoce por el programa realmente hasta después de dos'ciclos de instrucción más un pulso de reloj. Si se utiliza el predivisor es indispensable saber que la unidad de sincronización está conectada detrás del predivisor. Si el RTCC utiliza reloj interno también es indispensable mantener el terminal de entrada RTCC a un nivel fijo.

Una señal de reloj aplicada al terminal RTCC debe cumplir las siguientes condiciones:

Predivisor no utilizado:
RTCC alto: **≥2tosc + 20 ns**
RTCC bajo: **≥ 2tosc + 20 ns**
Predivisor utilizado:
Periodo
RTCC: **≥(4tosc + 40 ns)/N**
RTCC alto: **≥ 10 ns**
RTCC bajo: 210 **ns**

donde, t_{osc} es el período de la señal del oscilador y N es el valor de predivisión.

El diagrama de bloques combinado del temporizador perro guardián, predivisor y reloj/temporizador de tiempo real se muestra en la figura anterior. Las distintas opciones que se pueden programar para estas unidades (utilizando el registro OPTION) son fácilmente reconocibles.

4.- Funcionamiento básico.

4.1.- SLEEP: modo de bajo consumo

El modo de funcionamiento de bajo consumo se activa ejecutando una instrucción SLEEP. Esta instrucción inicializa el temporizador perro guardián (si había sido habilitado), borra el bit PD del registro de estado STATUS, activa el bit T0 y para el oscilador del reloj. Los puertos de E/S mantienen el estado que tenían antes de ejecutar la instrucción SLEEP. Para obtener el mínimo consumo posible en este modo, todos los terminales de E/S deben estar a un nivel fijo, esto es, conectados a Vdd o Vss.

Cuando se para el oscilador del reloj, el RTCC también se para. El dispositivo puede "despertar" (de ahí el nombre de la instrucción) de este modo al finalizar el tiempo del temporizador perro guardián o por la detección de un flanco positivo al final de un nivel bajo aplicado a la entrada /MCLR. En ambos casos, el procesador PIC invertirá un periodo de arranque del oscilador antes de que el programa comience desde la dirección de inicialización.

El bit PD del registro de estado se puede utilizar para determinar si el procesador esta en secuencia de encendido, o en secuencia de salida del modo de bajo consumo. De forma análoga, el bit T0 nos indica si la salida del modo de bajo consumo se debe a una señal /MCLR externa o a una finalización del período del temporizador perro guardián.

4.2.- Inicialización

Existen tres formas diferentes de inicializar un procesador PIC:

- Aplicar un nivel lógico bajo la entrada /MCLR
- Desconectar y volver a conectar la tensión de alimentación (inicialización en el encendido);
- Inicialización automática al finalizar el tiempo programado en el temporizador perro guardián.

El controlador permanece en el estado de inicialización durante el tiempo de arranque del oscilador (OST) y/o mientras que la entrada MCLR se mantenga a nivel lógico bajo. El tiempo de arranque del oscilador comienza con el flanco positivo de la señal /MCLR. Si la entrada /MCLR se conecta a la línea de alimentación positiva, el OST comienza en el instante en que se aplica la tensión de alimentación.

El OST tiene una duración de 9ms a 30ms. Durante una condición de inicialización, el estado del controlador PIC viene definido de la siguiente forma:

- El oscilador está funcionando o comenzando a funcionar (encendido o arranque desde el modo SLEEP).
- Todos los terminales de E/S de los puertos se sitúan en el modo de alta impedancia al programar los registros TRIS todos a '1' (esto es, en modo entrada).
- El contador de programa se sitúa en la siguiente dirección:
    - 01FFh en el PIC16C54/55,
    - 03FFh en el PIC 6C56,
    - 07FFh en el PIC 16C57.
    - 0000h en el PIC 16F84.
    - 0000h en el PIC 16F87X.

- El registro OPTION se sitúa todo a '1'.
- El temporizador perro guardián y su predivisor se ponen a cero.
- Los tres bits superiores del registro de estado (STATUS) se ponen a cero.
- La señal CLKOUT del terminal OSC se sitúa a nivel bajo (SOLO en los dispositivos tipo RC) .

5.- Juego de instrucciones del PIC16C5X

Las instrucciones en la mayoría de los juegos de instrucciones de los procesadores normalmente constan de dos partes: un código de operación (la instrucción propiamente dicha) y un operando (una posición de memoria o un registro). El operando es el objeto del código de operación. La mayoría de microprocesadores y microcontroladores utilizan uno o más bytes para códigos de operación y operandos. No ocurre lo mismo con los procesadores PIC, que trabajan con palabras de 12,13 o 14 bits de anchura que contienen el código de la instrucción y los operandos. Esta configuración tiene sus ventajas y sus desventajas. Por ejemplo, la estructura de 12 bits limita a 33 o 35 el número de posibilidades para el código de instrucción. Por otra parte, tiene la ventaja de ofrecer un procesado muy rápido de la instrucción dado que toda la información se capta de un solo golpe.

Las instrucciones utilizadas por el PIC16C5x se pueden dividir en tres grupos diferentes:

- Instrucciones de fichero de registros orientadas a nivel de byte.
- Instrucciones de fichero de registros orientadas a nivel de bit.
- Instrucciones de constantes y de control.

Debemos resaltar una marcada diferencia con otros procesadores en lo que a instrucciones orientadas a nivel de byte se refiere. Por ejemplo, usualmente, se utiliza un registro de trabajo (registro W en los dispositivos PIC, acumulador en los otros procesadores) cuando se suman dos números.

Con los dispositivos PIC, el programador tiene la opción de almacenar el resultado en el registro de trabajo, W, o en un fichero de registros especificado en el operando. En el código de instrucción, la elección se realiza a través del denominado bit de destino, d.

Desafortunadamente, Microchip Technology confía al programador la misión de declarar el bit de destino. En consecuencia, el bit 'd' tiene que declararse de la siguiente forma al comienzo de cada programa:

```
; definición de destino
W     EQU   0H    ; destino = W
F     EQU   1H    ; destino = F
```

El juego de instrucciones del PIC se resume en las tabla siguientes.

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

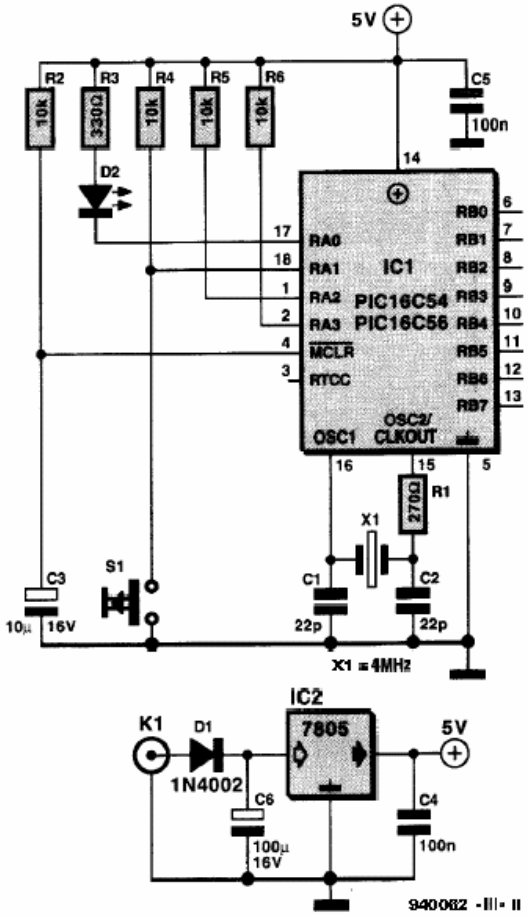
Los comandos y directivas utilizados por el ensamblador de PIC (entorno MPLAB) se listan en la tabla siguiente.

Directivas de datos		
data	<expresión>	Crear valor dato o cadena de 14 bits
zero	<mem>	Inicializar con cero zona del espacio de programa
set	<etiqueta> | <expresión>	Definir valor al ensamblador
res	<mem>	Reservar palabras del espacio de programa
equ	<etiqueta> | <expresión>	Definir constante al ensamblador
Incluye	“<fichero>”	Incluir un fichero en el fuente de ensamblador

Ejemplo de programación: un LED intermitente.

Las salidas de un procesador PIC pueden excitar cargas de hasta 20 mA. Sin embargo, la máxima corriente consumida por el dispositivo PIC debe ser inferior a 50 mA, mientras que la corriente que circula por la conexión de masa (GND) no debe exceder los 150 mA.

Un ejemplo de programación, trabaja en conjunción con la configuración hardware mostrada en la figura. Un LED y una resistencia serie de 330 Ω se conectan entre la línea de tensión positiva (+5 V) y la línea de E/S RA0. El resto de terminales de E/S se conectan a +5 V a través de resistencias de pull-up de 10 KΩ. Un pulsador se conecta entre la línea RA1 y masa.



La función del programa es muy sencilla: hacer que el LED comience a parpadear en cuanto se pulse S1. Cuando se programe el PIC, asegurarse de que se utiliza la opción de oscilador XT (cristal de cuarzo).

```
      TITLE "Ejemplo de parpadeo de LED"
; Opciones del procesador empleado
      list  P=16f84a
; Definiciones de registros basicos (en el fichero p16f84.inc)
#include    p16f84a.inc
;  Fija  los  bits  de  configuracion  a  partir  de  definiciones  en
P16F84.INC
      __CONFIG  _LP_OSC & _PWRTE_OFF & _WDT_OFF & _CP_OFF

; Dirección de los puertos de E/S
Entrada           EQU   01H
Salida            EQU   00H
; Registros de proposito general definidos por usuario
us_registro       EQU   09H
ms_registro       EQU   0AH
```

```
; Declaraciones de recursos hardware
LED_B             EQU   00H   ; Led conectado a RA0
LED_P             EQU   PORTA
LED_off_time      EQU   00H   ; LED off 256 ms
LED_on_time       EQU   00H   ; LED on 256 ms
TECLA_B           EQU   01H   ; Tecla conectada a RA1
TECLA_P           EQU   PORTA

;*****************************************************************
;                 Programa principal
;*****************************************************************
      ORG   00H
Inicio                      ;Comienzo tras reset
            MOVLW 0EH       ;Pone RA0, conectada al LED como salida
            TRIS  PORTA
            GOTO  LED_OFF           ;Inicialmente LED apagado
LOOP        BTFSC TECLA_P,TECLA_B   ;Tecla pulsada? (RA1=0?)
            GOTO  LOOP              ;No, esperar a que se pulse
            BCF   LED_P,LED_B       ;Encender LED
            MOVLW LED_on_time       ;Esperar
            CALL  WAIT_MS
LED_OFF     BSF   LED_P,LED_B       ;LED off
            MOVLW LED_off_time      ;Esperar
            CALL  WAIT_MS
            GOTO  LOOP

;*****************************************************************
; Subrutina WAIT_MS
; Parametro:            ms en el registro W
; Valor devuelto:       00H
; El tiempo de retraso es 1ms * W para Fosc=4MHz
; Si W=0 -> 256 ms
;*****************************************************************
WAIT_MS     MOVWF ms_registro ;Almacena ms
            MOVLW 0FEH        ;Corrige el tiempo de llamada
            MOVWF us_registro ;para el primer bucle
W_LOOP_MS   MOVLW 0F9H        ;un bucle es 249*W_LOOP_US+4ciclos
            ADDWF us_registro,F
W_LOOP_US   NOP               ;un bucle es 4 ciclos
            DECFSZ us_registro,F    ;ms_registro-1, salta si cero
            GOTO  W_LOOP_MS
            NOP               ;corregir tiempo de ultimo bucle ms
            RETLW 00H
      END
```

Después del titulo se define el tipo de procesador empleado con la directiva LIST P=16F84A. A continuación se emplea la directiva #include para incluir un fichero donde aparecen las etiquetas que definen los registros especiales, las palabras W y F que se utilizan como destino de la operación de la ALU y otras etiquetas más como los puertos de E/S.

A continuación con la directiva __CONFIG se definen el estado de los bits de configuración que necesita el dispositivo programador para configurar cosas como el tipo de oscilador que va a emplear el chip, el uso de Watchdog o los bits de protección anticopia.

Se siguen declarando más constantes del programa y de recursos hardware. Unas definiciones bien comentadas son esenciales para disponer de un buen conocimiento de la estructura del programa, y son esenciales para un eficiente desarrollo y posterior depuración del programa.

5.1.- Instrucciones de transferencia de datos

La primera instrucción de la subrutina 'Wait_ms' es la MOVWF. Esta instrucción "ordena" al controlador que guarde el contenido del registro W directamente en la posición 'ms'. Esta posición fue declarada anteriormente en la línea 65 (EQU 00AH).

Las siguientes instrucciones provocan que se cargue una posición denominada 'us_registro' (de microsegundo) con el valor 0FEH. Dado que no existe una instrucción que permita realizar esta función directamente, se recurre a una alternativa en la cual el registro W juega un papel importante. La instrucción MOVLW sitúa la constante en el registro W. Existen dos formas posibles para situar el valor '0' en el registro W: utilizando la instrucción MOVLW 00H o de forma más sencilla utilizando la instrucción CLRW. Sin embargo, obsérvese que la instrucción CLRW también sitúa a '1' el indicador de cero, Z.

Otra instrucción para borrar el contenido de una posición de memoria en el fichero de registros es la 'CLRF'. Al igual que la CLRW, la instrucción CLRF no utiliza el registro W y activa el indicador Z.

Una última opción es la instrucción MOVF, que permite copiar el contenido de una posición de memoria en el registro W. Alternativamente, esta instrucción permite utilizar como destino la posición de memoria incluida en el operando, en lugar del registro W. El resultado de esta instrucción es que los contenidos de una posición simplemente se devuelven a la misma posición. Esta opción de la instrucción es muy útil, por ejemplo, en las operaciones con los puertos de E/S. Una instrucción es suficiente para leer el estado lógico de un puerto, y devolver la información al registro temporal. La misma instrucción también se puede utilizar para comprobar si el contenido de un registro es igual a '0'.

Como ya hemos mencionado, la subrutina "espera" el número de milisegundos definidos a través del registro W. Esto se consigue utilizando dos bucles anidados. El bucle exterior tarda exactamente 1 ms si se utiliza una frecuencia de reloj de 4 MHz. Este bucle se invoca tantas veces como se defina en el registro W. El número de iteraciones del bucle interior no se almacena como constante en el registro 'us_registro', sino que se añade a la constante del registro. Esto se hace así con el propósito de permitir una compensación del tiempo necesario para que arranque la subrutina. La subrutina finaliza su función si la posición 'us_registro' alcanza el valor 00H.

5.2.- Instrucciones aritméticas

Las instrucciones aritméticas disponibles con los procesadores PIC se limitan a la pareja ADDWF y SUBWF, y sus derivadas INCF y DECF. Una aplicación de la instrucción ADDWF se puede encontrar en la línea donde un valor igual a 0F9H (copiado en el registro W) se suma al contenido de la posición 'us_registro':

```
ADDWF us_registro
```

Todas las instrucciones aritméticas permiten que bien el registro W o el registro F en el operando funcionen como registro destino. Esto permite sumar un cierto valor a varios registros 'F' de una sola vez. Desafortunadamente, los procesadores PIC carecen de instrucciones del tipo sumar con acarreo y restar con acarreo. En consecuencia, la suma de números con anchura mayor de 8 bits requiere procesar independientemente el indicador de acarreo. A continuación se muestra un programa ejemplo en el que se suman dos números de 32 bits. Estos números son 'Z' y 'N'. 'Z' consta de los bytes Z_HH, Z_HL, Z_LH y Z-LL. Lo mismo se aplica para 'N'.

```
; Suma dos numeros de 32 bits N y Z. El resultado lo guarda en N
; Si N+Z > 2**32 se pone a uno el flag de CARRY
;
;  N_LL, Z_LL: Bit 0..7
;  N_LH, Z_LH: Bit 8..15
;  N_HL, Z_HL: Bit 16..23
;  N_HH, Z_HH: Bit 24..31

ADD_LL      MOVF        Z_LL,W      ;Carga W con Z_LL
            ADDWF       N_LL,F      ;Suma N_LL y W, almacena en N_LL
            BTFSC       STATUS,C    ;Salta si C=0
            GOTO        CARRY_LH
            MOVF        Z_LH,W      ;Carga W con Z_LH
ADD_LH      ADDWF       N_LH,F      ;Suma N_LH y W, almacena en N_LH
            BTFSC       STATUS,C    ;Salta si C=0
            GOTO        CARRY_HH
            MOVF        Z_HH,W      ;Carga W con Z_HH
ADD_HH      ADDWF       N_HH,F      ;Suma N_HH y W, almacena en N_HH
            RETLW       00H         ;Retorna, carga W con 00H

; Rutina de propagacion del acarreo

CARRY_LH    INCFCZ      Z_LH,W      ;Carga W con Z_LH+1, salta si W=0
            GOTO        ADD_LH
CARRY_HL    INCFCZ      Z_HL,W      ;Carga W con Z_HL+1, salta si W=0
            GOTO        ADD_HL
CARRY_HH    INCFCZ      Z_HH,W      ;Carga W con Z_HH+1, salta si W=0
            GOTO        ADD_HH
            SETC                    ; overflow -> CARRY=1
            RETLW 00H               ;Retorna, carga W con 00H
```

Cuando se utiliza la instrucción SUBWF, debe tenerse en cuenta que ésta se realiza conforme al método de complemento a dos. En consecuencia, el indicador de acarreo se activa inversamente.

La utilización de las instrucciones DECF e INCF es evidente. Estas instrucciones respectivamente decrementan (disminuyen) 0 incrementan (aumentan) en una unidad el contenido de una determinada posición de memoria. En este caso, también el resultado se guarda en el registro F declarado en el operando, o en el registro de trabajo W. Si el resultado es igual a cero, se activa el indicador de cero, Z.

5.3.- Control del flujo del programa: instrucciones de salto.

Usualmente, un programa de ordenador se ejecuta desde la posición más baja de memoria a la posición más alta de memoria. Para controlar el flujo o sentido de ejecución del programa se disponen de instrucciones especiales, por ejemplo para saltar una sección concreta del programa. La más sencilla de todas estas instrucciones es la instrucción NOP, utilizada en la línea 106 del programa ejemplo. Esta instrucción simplemente incrementa en una unidad el contador de programa e invierte exactamente un ciclo de instrucción. Desde cualquier otro punto de vista, la instrucción NOP no hace nada. La descripción formal de la instrucción NOP, al igual que del resto de instrucciones, se muestra en las tablas adjuntas.

El resto de instrucciones que controlan el flujo de un programa son todas ellas instrucciones de salto. Por ejemplo, instrucciones no condicionales son GOTO y CALL, que siempre ejecutan un salto. En contraste, las instrucciones de salto condicional no realizan el salto hasta que se cumpla una determinada condición. Veamos en detalle este tipo de instrucciones.

5.3.1.- Instrucciones de salto incondicional

En una instrucción GOTO, los bits inferiores se utilizan para indicar la dirección de destino. En el caso del PIC16C56, el nivel del bit 9 se determina a través del registro de estado, de forma análoga, el bit 9 y el bit 10 se determinan así en el PIC16C57. Estos bits se deben configurar en función de la dirección deseada. En la familia PIC16F84 no existe este problema puesto que la instrucción GOTO proporciona 11 bits de la dirección, por lo que se puede direccionar hasta 2K palabras.

En el PIC16F876 la instrucción GOTO solo proporciona 11 bits de la dirección, pudiendo seleccionar una de las páginas mediante los bits 3 y 4 del registro PCLATH si se produce un cambio de página. En este último caso se opta a guardar las rutinas de cada módulo que forme el programa en paginas separadas para evitar tener que estar cambiando de pagina continuamente.

En las técnicas de programación “clásicas”, las subrutinas tenían una función importante: permitían incluir en una sola posición del programa una secuencia de instrucciones que se utilice muchas veces. En cualquier instante en que se necesite la subrutina, simplemente se invoca.

Las modernas técnicas de programación utilizan las subrutinas para construir un programa en base a bloques funcionales. Las subrutinas siempre tienen una dirección de inicio y se terminan con una instrucción RET, la cual marca el final del subprograma que a fin y al cabo constituye la subrutina. Cuando se llama a una subrutina utilizando la instrucción CALL, es importante recordar que el bit 8 de la dirección de salto siempre está a 0. La instrucción CALL sólo permite definir los bits 0 a 7. Al igual que con la instrucción GOTO, el bit 9 en el PIC16C56 y los bits 9 y 10 en el PIC16C57 vienen determinados por los bits PA0 y PA1 del registro de estado. En consecuencia, las direcciones de comienzo de subrutinas siempre deben estar comprendidas en la parte inferior de la memoria de programa. Esta pequeña restricción se puede obviar situando una instrucción GOTO en el bloque de memoria inferior. Este GOTO deberá apuntar a subrutinas situadas en el bloque de memoria superior. Esto no es necesario tenerlo en cuenta con el PIC16FXXX.

Cuando se ejecuta una instrucción CALL, el contenido en curso del contador de programa se envía a la pila. Puesto que sólo hay dos posiciones de memoria disponibles en la pila, sólo se pueden anidar dos subrutinas. Dado que el contenido completo del contador de programa se guarda en una posición de memoria de la pila, las instrucciones CALL se pueden incluir en cualquier posición del programa.

Cada subrutina debe terminar con una instrucción RETLW. Esta instrucción recupera el contenido de la pila y lo “copia” en el contador de programa. El programa continua entonces desde la primera instrucción después del CALL. La instrucción RETLW también sitúa una constante en el registro W. Esta constante también se puede utilizar para enviar información desde una subrutina al programa principal.

5.3.2.- Instrucciones de salto condicional

Los saltos condicionales en el código ensamblado del PIC 16C5x tienen una dirección de salto fija. Si se cumple la condición previamente definida, la siguiente instrucción después del salto condicional se “salta”. Según se ilustra en las líneas 130 y 131 la posición de memoria “saltada” puede contener una instrucción CALL si se desea saltar a una cierta (sub)rutina o GOTO si se desea saltar a una cierta posición del programa.

El juego de instrucciones de los PIC sólo tiene cuatro instrucciones de salto condicional: BTFSC F,b y BTFSS F,b, que comprueban el bit b del registro F. Las instrucciones DECFSZ

F,d e INCFSZ F,d decrementan e incrementan en una unidad, respectivamente, los contenidos del registro f. La instrucción que sigue a cualquiera de estas instrucciones de salto condicional siempre se salta si el resultado de la correspondiente operación es 0.

Si d=l, el nuevo valor se guarda de nuevo en el registro F. Si d=0, se guarda en el registro W. En el último caso, el valor original se desplaza a la izquierda en el registro F. Véase las tablas adjuntas para las descripciones formales de estas instrucciones.

5.3.3.- Comparación con una constante

A menudo se requiere comprobar una variable contra una constante. El programa ejemplo listado en la figura muestra cómo se puede conseguir esto de una forma muy sencilla. El método es casi idéntico si deseamos comparar los contenidos de dos registros; todo lo que se requiere es sustituir la instrucción MOVLW Comp_-Value por una instrucción MOVF.

```
; Comparar registro 'us_registro' con 046H

Comp_value  EQU   046H

COMPARE     MOVLW Comp_value        ;Carga W con 046H
            XORWF us_registro,W     ;XOR W con us_registro
            BTFSC STATUS,Z          ;W=0?
            GOTO  IGUAL
NO_IGUAL
            MOVLW Comp_value
            SUBWF us_registro,W     ;us_registro - W
            BTFSC STATUS,C          ;(W>0) (Carry=1?)
            GOTO  MAYOR             ;Si->us_registro>46H
            GOTO  MENOR             ;Si->us_registro<46H

MENOR                               ;Rutina para us_registro<46H

MAYOR                               ;Rutina para us_registro>46H

IGUAL                               ;Rutina para us_registro=46H
```

5.4.- Instrucciones lógicas

El procesador PIC ofrece siete instrucciones lógicas. Una de ellas es la XORWF utilizada en el programa ejemplo. No hay mucho que decir respecto estas instrucciones, todas las operaciones se realizan bit a bit. Con las instrucciones que hacen uso de un registro del fichero de registros, el resultado de la operación (por ejemplo, una suma) se puede almacenar bien en el registro W o de nuevo en el fichero de registros. Con todas las instrucciones lógicas, se activa el indicador de cero (Z) si el resultado de la operación es cero.

Las instrucciones de rotación usualmente se clasifican dentro del grupo de instrucciones lógicas. El procesador PIC dispone de dos instrucciones de rotación: en sentido de las agujas del reloj, RRF (rotación a la derecha), y en sentido contrario al de las agujas del reloj, RLF (rotación a la izquierda). El bit de acarreo también está involucrado en la operación de rotación. Rotar a la derecha significa que los bits se desplazan hacia abajo una posición, el MSB (bit más significativo) toma el valor del bit de acarreo, y el LSB (bit menos significativo) se desplaza al bit de acarreo. La otra operación, RLF, hace lo mismo pero en el sentido contrario. Al igual que con las otras instrucciones orientadas a nivel de byte, el resultado de RRF y RLF puede guardarse bien en el registro W o en una posición de memoria F.

5.5.- Instrucciones de manejo de bit

Los procesadores PIC tienen un total de 4 instrucciones orientadas a nivel de bit o de manejo de bit. Dos de ellas, BTFSC y BTFSS, son instrucciones de salto orientadas a nivel de bit. Estas instrucciones realizan un salto dependiendo de que un bit cumpla o no una cierta condición. Ambas instrucciones han sido explicadas anteriormente dentro de la sección de instrucciones de salto condicional. Las otras dos instrucciones se pueden utilizar para fijar a 0 o a 1 un bit de una de las posiciones de memoria de un fichero de registro. Estas instrucciones tienen dos operandos: el primero indica la dirección del fichero de registro, y el segundo, el número de bit. El bit 0 siempre se refiere al bit menos significativo (LSB).

La línea 133 del programa ejemplo contiene la instrucción BCF. Esta instrucción se utiliza para encender el LED conectado a la línea RA0 al situar a nivel lógico bajo (0) esta línea de puerto. El LED se apaga de nuevo en la línea 137 al situar a nivel lógico alto (1) la línea RA0 utilizando la instrucción BSF.

Las instrucciones BCF y BSF tienen una característica especial: se procesan por medio de un acceso interno a la posición indicada con una anchura de direccionamiento de 8 bits. Esto significa que el estado en curso del puerto (con la excepción del bit a cambiar) se lee y guarda directamente en una memoria buffer. Si un puerto de E/S se configura como entrada para la instrucción de bit que se aplica a uno de los otros terminales, y éste se configura como salida después de la ejecución de la instrucción, el nivel en la salida correspondiente es igual al que había en la entrada mientras que la instrucción se está ejecutando. Por tanto, se recomienda asegurarse de que el buffer esté a un nivel fijo antes de la conmutación de entrada a salida del correspondiente puerto.

6.- Definición de componentes hardware

Volvamos con el programa ejemplo. El programa principal siempre comenzaría con una inicialización de los componentes hardware. Esta información está contenida en las líneas 126 y 127, que configura los puertos de E/S el registro OPTION. Para realizar esto existen dos posibles instrucciones. La instrucción TRIS siempre nos permite definir una línea de puerto como entrada o salida, mientras que la instrucción OPTION sirve para llenar o definir el registro OPTION. La configuración hardware se puede cambiar según evoluciona el programa.

Para asegurar que el programa no falla en un entorno eléctricamente ruidoso, Microchip recuerda a los programadores la obligatoriedad de repetir la definición de los componentes hardware en distintas posiciones del programa. De hecho, un mecanismo interno del procesador incluso repite la definición de los terminales de E/S como cada acción sobre un puerto de E/S.

Estas medidas tan drásticas son necesarias debido a la irrevocable acción de redefinición del temporizador perro guardián, el cual fuerza una inicialización cuando el programa falla. Esta inicialización también puede provocar que otros bits de los especificados se inicialicen de forma incorrecta.

La instrucción OPTION sitúa los contenidos del registro W en el registro OPTION. Las funciones de los distintos bits de este registro ya las hemos tratado al hablar de la descripción hardware del procesador.

7.- Otras instrucciones

Además de las instrucciones descritas con referencia al programa ejemplo, existes tres instrucciones más. La instrucción SWAPF es difícil de encuadrar en un tipo de categoría de instrucciones. Como indica la traducción del término, la instrucción SWAPF intercambia alterna (SWAP) entre sí los nibbles alto y bajo. Recordar que un nibble es la mitad de un byte, esto es, un conjunto de 4 bits. Por tanto, en sentido práctico la instrucción SWAPF intercambia los bits 0 a y 4 a 7 respectivamente. Esta instrucción es muy utilizada en operaciones aritméticas en BCD.

El temporizador perro guardián del procesador PIC se activa por software, esto es, por programación. El temporizador se puede inicializar utilizando la instrucción CLRWDT. Si el predivisor se asigna a este temporizador, también éste se inicializará. El temporizador perro guardián también se puede inicializar utilizando la instrucción SLEEP. Esta instrucción provoca la conmutación del procesador al modo de bajo consumo o reposo. El procesador "despierta" de nuevo cuando ha transcurrido el tiempo del temporizador, o cuando ocurre una inicialización hardware.

8.- Fin de programa

El final de un programa para PIC invariablemente contiene la definición de un vector de inicialización (RESET). Dado que, normalmente, un programa no ocupará la capacidad total del procesador, se debe utilizar un truco para situar el puntero del vector al final del espacio de memoria. Esto se puede lograr utilizando la instrucción ORG, según se ilustra en la línea 144. Dado que los diferentes tipos de controladores PIC tienen diferentes tamaños de memoria, la dirección del fin se define con la ayuda de una estructura IF-ENDIF al comienzo del programa.