PIC's 1ª PARTE RESUMEN Y EJERCICIOS_06/07

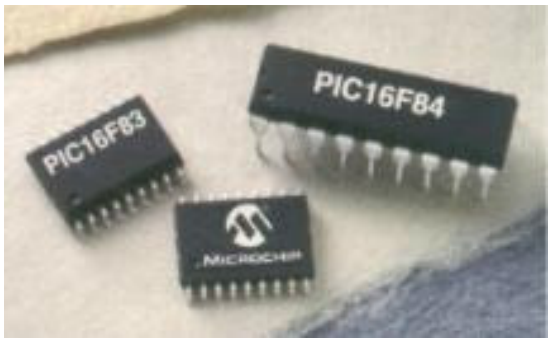
INDICE

CAPITULO 1.
ARQUITECTURA PIC

Fundamento de los sistemas controlados por microprocesador, diagrama de bloques

Concepto de microcontrolador.

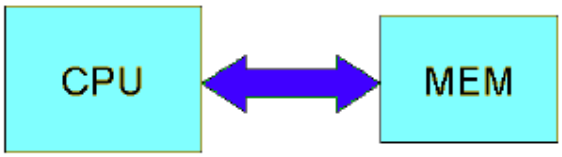
Hasta llegar al microcontrolador podemos establecer las distintas fases.
1- Lógica cableada, función definida por el cableado de componentes entre si.
2-Sistemas lógicos programables integrados en distintos bloques: microprocesador, memoria, bloques I/O
3- Microcontrolador: Sistema que incorpora todos sus componentes en un sólo chip.
Son sistemas cerrados, mientras que los sistemas con microprocesador son sistemas abiertos, al poder conexionar los distintos bloques a los buses del sistema basado en el microprocesador..
También los PIC's difieren de los microcontroladores en que tienen aplicaciones de propósito específico, su arquitectura se adapta a aplicaciones concretas y su programa realiza una única tarea principal.



La arquitectura Von Neuman, y Havard.

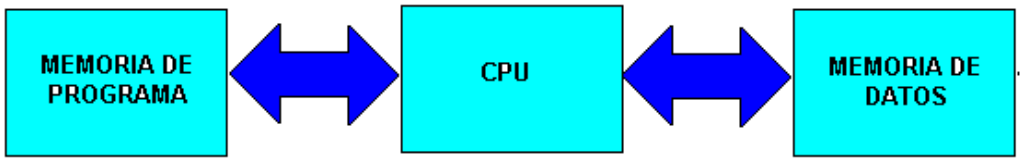
La arquitectura Von Neuman.

Una de sus principales características, es que en su conjunto las instrucciones tienen distinto número de Bytes, y que cada dirección de memoria o port, hace referencia a un Byte.
La misma memoria guarda las instrucciones del SO, programa y también los datos del proceso, comunicándose por un mismo BUS



Utiliza instrucciones **CISC**: los microcontroladores suelen estar basados en la utilización de tipo de instrucciones CISC (Computadores de Juego de Instrucciones Complejo). Pueden llegar a disponer de un considerable número de instrucciones, muchas de ellas altamente complejas y potentes, cada una de estas instrucciones requiere un elevado número de microinstrucciones para su ejecución en código máquina. Presentan la ventaja de una mas fácil y automática programación, al sustituir cada de ellas a un conjunto de instrucciones a modo de macro o subrutina

La arquitectura Harvard
Es la utilizada por los microcontroladores PIC (Programable Interface Controller)



La arquitectura Harvard, consiste en que existen dos memorias separadas, conectadas con la CPU mediante buses diferentes, de tal forma que en las posiciones de la memoria de datos puede haber un numero de bits diferente al de la memoria de programa.
La memoria que contiene el programa es no volátil (EEPROM, FLASH), y todas las. Instrucciones son de formato de longitud constante, es decir tienen el mismo número de Bits con el fin de optimizar el funcionamiento del sistema.
La otra memoria almacena los datos de proceso y es volátil (RAM), y puede existir otra memoria de tipo EEPROM, para copiar y guardar parte de la RAM con el fin de que no se pierdan esta información al desconectar la alimentación.
Al tener la CPU dos memorias independientes con sus respectivos Buses, es posible el acceso solapado de instrucciones de programa y de los datos, permitiendo una mayor velocidad de trabajo del sistema, cada instrucción se ejecuta en un ciclo de instrucción, 4 ciclos de reloj , excepto en las instrucciones de salto
Las Instrucciones se denominan ortogonales, al poder controlar los elementos del sistema como fuente o destino.
Utilizan un conjunto de instrucciones reducido, **RISC** (Reduced Instrucción Set Computer, las instrucciones son simples y, generalmente, se ejecutan en un solo ciclo, de tal forma que por su simplicidad cada instrucción se puede ejecutar con mucha rapidez y optimizar el funcionamiento .
Las instrucciones SISC, (Computadores de Juego de Instrucciones Específico), consisten en un conjunto de instrucciones específico para microcontroladores en aplicaciones concretas.

Principales componentes del microcontrolador PIC.
La CPU (Unidad Central de Proceso)
Realiza el control del sistema en base a direccionar para buscar las instrucciones del programa, recibir y decodificar los códigos de operación, y generar las microinstrucciones para la ejecución de las tareas a realizar, como la búsqueda de los operandos, almacenamiento de resultados, también realizar las operaciones aritméticas y lógicas, etc.
Memoria
La memoria de instrucciones y datos está integrada en el propio chip. Como ya se ha indicado anteriormente la memoria que contiene el programa es no volátil (EEPROM, FLASH),la otra

memoria almacena los datos y es volátil (RAM), y puede existir otra memoria de tipo EEPROM, para copiar y guardar parte de la RAM.
Los microcontroladores se diferencian de los ordenadores y PC's, en que no disponen de elementos de almacenamiento masivo como disco duro, cintas, Cd's, etc, y solo ejecutan el programa que previamente se ha grabado en la memoria de datos.

Memorias OTP (One Time Programmable) son memorias no volátiles de sólo lectura que solo pueden programarse una sola vez.

Memorias EPROM (Erasable Programmable Read Only Memory) Son memorias de lectura, que permiten ser borradas y grabadas un considerable número de veces. El borrado se realiza al disponer de una ventana de cristal de cuarzo en someter la EPROM a radiaciones ultravioleta un determinado periodo de tiempo.

Memorias EEPROM, E2PROM o E²PROM. Electrical Erasable Programmable Read Only Memory),
También son otro tipo de memorias que pueden ser grabadas y borradas un considerable número de veces. El borrado se realiza eléctricamente, siendo este procedimiento mas rápido y sencillo que el anterior de la EPROM.

Memorias FLASH
Similar a la EEPROM, pero más rápida y de mayor capacidad.

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes

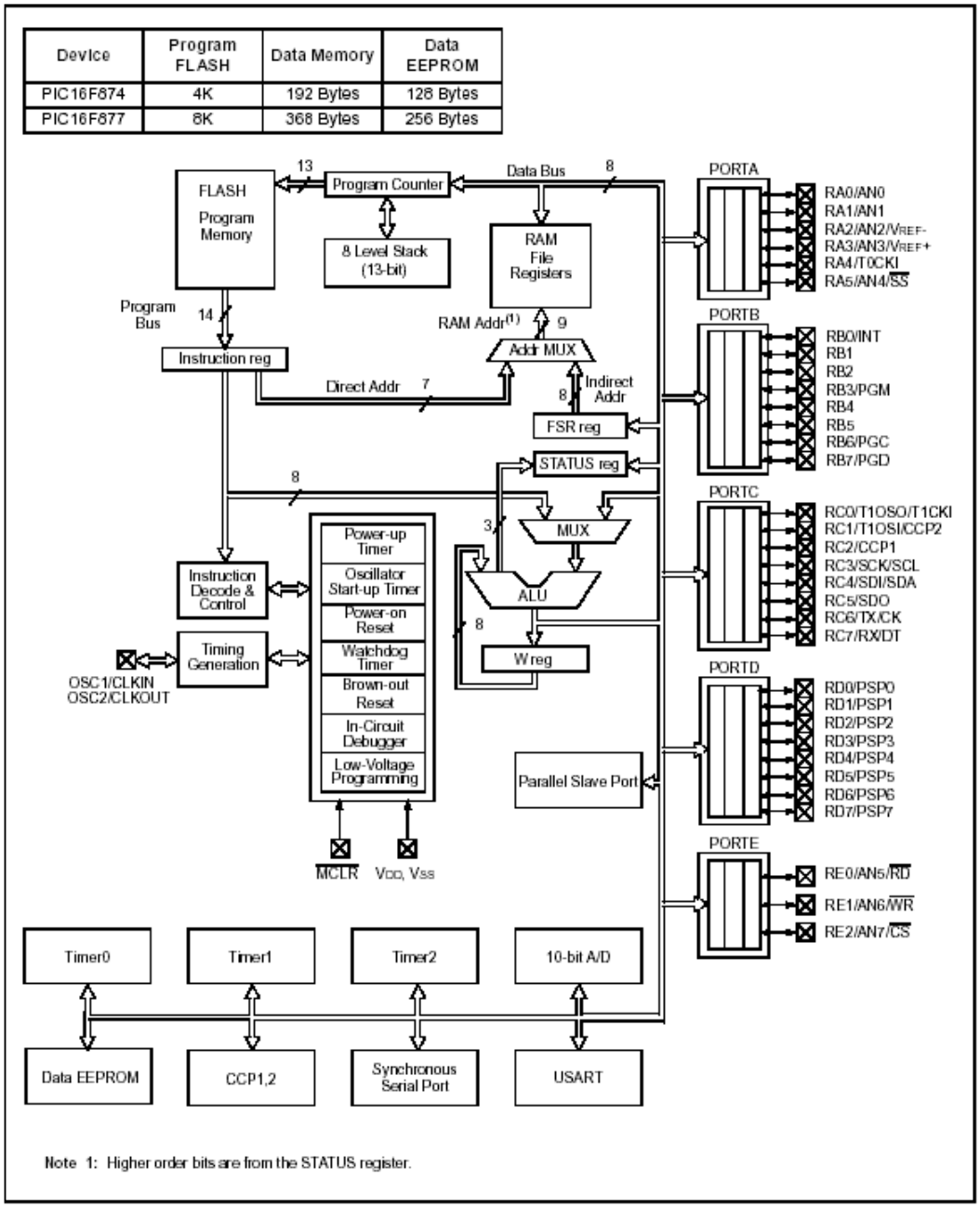


Diagrama de bloques de los principales componentes del PIC.

Ports de Entrada y Salida

Las lineas de Entrada y Salida (E/S) forman parte de los dispositivos del microcontrolador para comunicar con el exterior.

Son programables como entrada o salida y también realizar funciones específicas..

Reloj principal.

Consta de circuito oscilador de cristal de cuarzo o RC, genera una señal rectangular del orden de MHz, que se utiliza para desencadenar cada una de las secuencias de control y de sincronismo del sistema. A frecuencias de reloj mas altas se disminuye el tiempo de ejecución de cada ciclo de programa, estando limitado por la tecnología y características del sistema.

Control de interrupciones

Permite controlare el salto desde el programa principal a determinadas subrutinas, cuando se dan unas determinadas condiciones hardware o software.

Recursos especificos.

Cada tipo de microprocesador puede incorporar toda una serie de módulos, con circuitos que permitan la realización de funciones especiales como:

Temporizadores o “Timers”

Se emplean para controlar periodos de tiempo (temporizadores) y para la cuenta de eventos (contadores) como impulsos, pasos de programa, etc.

Como temporizador se carga un registro con el valor de para la temporización, y se van aplicando impulsos definidos por un tiempo base y a través de un divisor. El valor del registro se va incrementa (o decrementando), hasta que alcanza el valor de desbordamiento y produce un aviso al microprocesados.

Para contar acontecimientos, el valor del registro que realiza el conteo se incrementando (o decrementando) al ritmo de los eventos a contar, hasta que se produce un rebosamiento o se alcanza un determinado valor, y se comunica para el control del microprocesador.

Watchdog **(Perro guardián)**

Consta de un contador que al desbordar hace el reset del microcontrolador, en el caso de fallo de funcionamiento del programa.

Cuando la ejecución del programa es correcta, el propio programa se encarga de realizar el clear del contador con el fin de evitar su desbordamiento. En el caso de no realizar este clear cuando el contador se desborda provoca un reset automático del sistema.

Browout (Fallo de alimentación)

Es la protección ante fallo de tensión de alimentación. Al iniciar el funcionamiento cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo (“brownout”) realiza un reset del microcontrolador. En caso que la tensión de alimentación no alcance la de brownout el microprocesador se mantiene en reset

SLEEP **Estado de reposo ó de bajo consumo.**

En caso que el microprocesador este en espera que se produzca algún acontecimiento externo para realizar el programa, con la finalidad deahorrar energía existe la instrucción SLEEP que pasa el sistema al estado de reposo o de bajo consumo.

Para lo cual se detiene el reloj principal permaneciendo inactivos casi todos los circuitos del sistema , mediante la activación de una interrupción generada por un determinado evento externo, se reanuda la actividad del sistema.

CAD **Convertidor Analógico/Digital**

Mediante el CAD se pueden aplicar señales analógicas al microprocesador, para que una vez realizada su conversión se puedan procesar en forma digital.

En caso de disponer de varias de estar entradas, su selección se realiza dentro del propio microcontrolador mediante un multiplexor analógico.

CDA **Conversor Digital/Analógico**

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de las patillas del chip. Existen muchos circuitos que trabajan con señales analógicas.

Comparador analógico

Consta de un Amplificador Operacional que realiza la comparación entre una señal de referencia y otra variable de entrada, la salida del comparador proporciona una señal digital con respecto al resultado de comparación.

PWM **Modulador de anchura de impulsos**

Proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patitas del encapsulado.

Puertas de comunicación

Permiten al microprocesador la comunicación con otros dispositivos externos, mediante otros distintos buses y protocolos de comunicación.

UART, adaptador de comunicación serie asíncrona.(Ej: Puerto Serie)

USART, adaptador de comunicación serie síncrona y asíncrona

Puertas paralelas de conexión con los buses de otros microprocesadores.

USB (Universal Serial Bus).

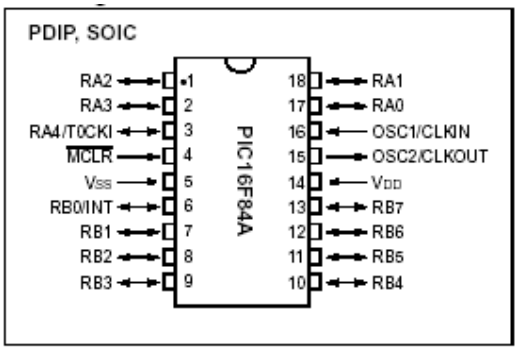
Bus I^2C, interfaz serie desarrollado por Philips, (constituido por dos cables conductores).

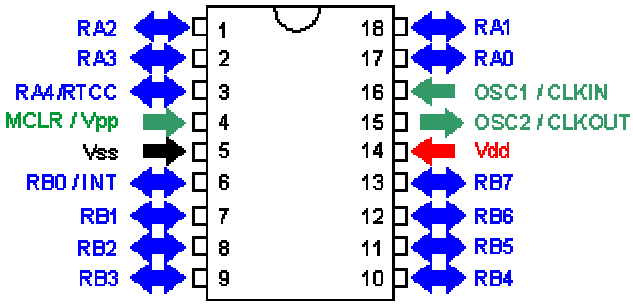
CAN (Controller Area Network), redes de conexionado multiplexado para el cableado electrico en automóviles, desarrollado por Bosch e Intel.

Bus I2C es un bus paralelo de 8 líneas para simplificar la circuiteria de dispositivos dentro

del televisor.

CAPITULO 2.
BLOQUES Y REGISTROS DEL PIC 16F84.





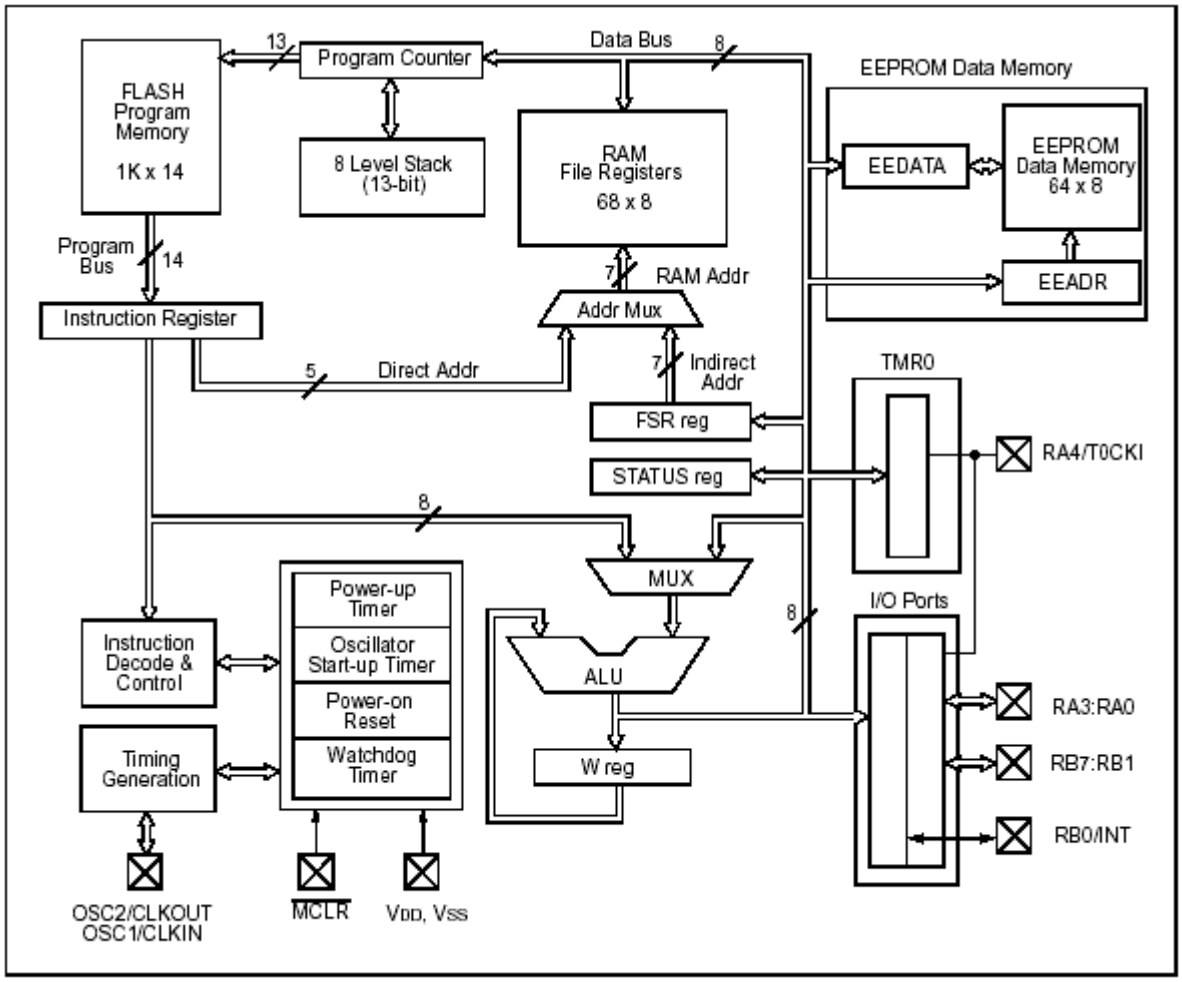
Distribución de los Pins del 16F84, y función de cada unos de ellos.

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

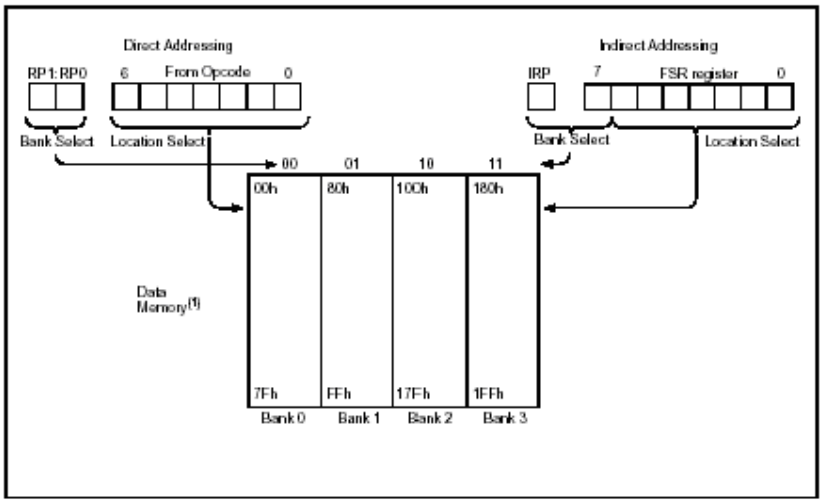
Legend: I= input O = Output I/O = Input/Output P = Power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

Diagrama de bloques del PIC 16F84.



Bancos y direccionamiento de la MEMORIA DE DATOS

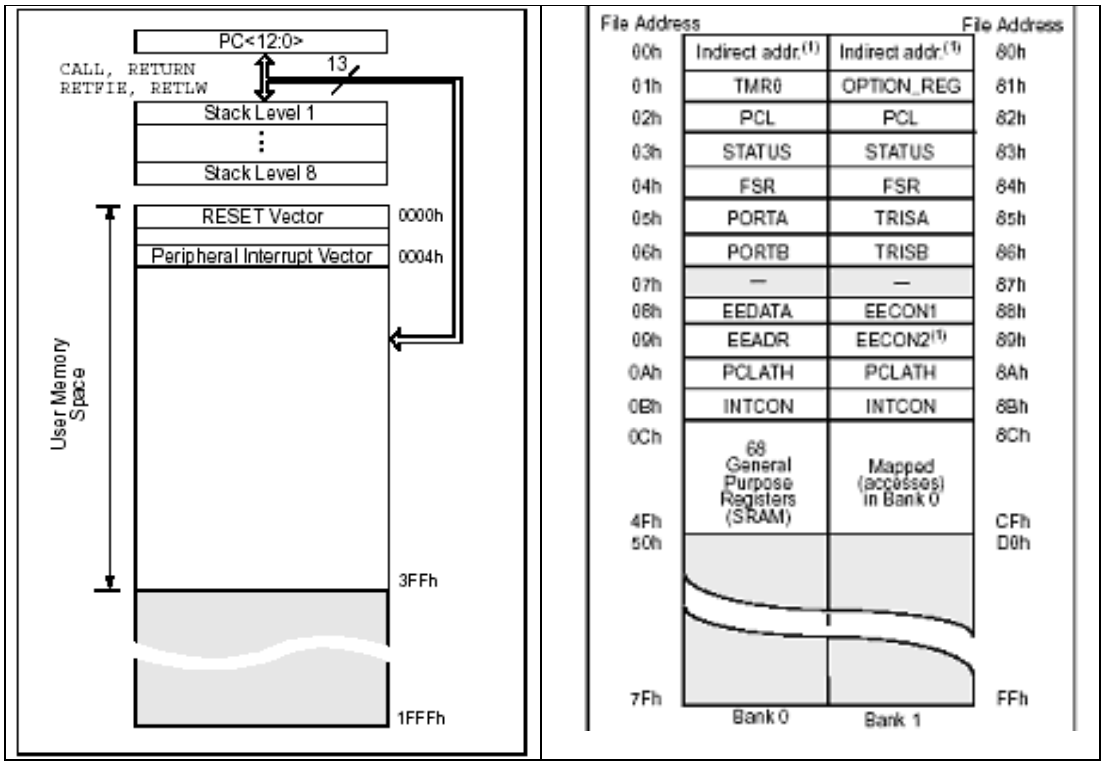


La memoria para su direccionamiento esta dividida en Bancos, pero de tal forma que algunas direcciones están solapadas.
El direccionamiento puede ser de dos formas, Directo o Indirecto

Direccionamiento DIRECTO
La dirección viene incluida en el propio código OP de la instrucción Bits6-0, y el acceso al banco viene determinada por los Bits RP del Registro de ESTADO

Direccionamiento INDIRECTO
Se utiliza en caso que el Registro INDF contenga el código Op de la Intrucción. En este caso la dirección viene definida por el contenido de los Bits6-0 del Reg FSR, y la dirección del Banco por el Bit RP del Registro de ESTADO y el Bit 7 del Reg FSR.

Organización de la memoria FLASH **Organización de la memoria RAM**



Están los Registros de propósito especifico **SFR** los cuales se encuentran a partir de la posición 00H en el Banco 0 de 80H en el Banco1, estros registros corresponden a funciones especificas a realizar dentro del propio PIC, y los Registros de propósito general **GPR** que se encuentran a partir de la posición 0CH en el Banco0 y de la 8CH en el Banco1 y pueden ser utilizados a voluntad del programador.

Función de cada uno de los bits de los registros SFR de la RAM

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

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> are 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.

Función de cada unos de los bits de los principales registros de propósito específico

Registro OPTION en Banco1. Asignación de divisor de frecuencia y tipo de temporizador.

TER 2-2: OPTION REGISTER (ADDRESS 81h)

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
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: 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 is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

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

Registro STATUS. Bits de selección de banco y señalización.

TER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

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	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0**: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

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/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$, 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/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$, the polarity is reversed)
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

Registro INTCON en Banco1 y Banco2. Tratamiento de las interrupciones

TER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE**: Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Registros TRISA y TRISB. Corresponden a los Ports A y B respectivamente. Programando el BIT correspondiente a cada línea con **0 la configuración es salida**, y con **1 queda configurada como entrada.**

CAPITULO 3

INTRUCCIONES DEL PIC 16F84,

El lenguaje ensamblador del 16F84

Identificación de elementos.
Cada mnemónico o instrucción consta de 14 bits, divididos en el código OPCODE que especifica el tipo de instrucción, y en los operandos (uno o mas).

OPCODES o instrucciones son abreviaturas usados en el lenguaje ensamblador, los 6 primeros Bits componen estos OPCODES.

OPERANDOS son los datos con los que la instrucción debe operar dentro del microcontrolador. Estos operandos son:
f Define el registro a utilizar (**f**ile register address) (0x00-0x7F)
w Registro de trabajo (**W**orking Register)

b Dirección de un **b**it dentro de un registro de 8 bits (0-7)
l ó k **L**iteral
d Bit de **d**estino 1->f 0->w
x Los bits representados por este tipo de dato no tienen ninguna función y su valor esta definido en el compilador.

f (file register)
Define el registro a utilizar, y contiene la dirección del registro, no su contenido. Un registro puede variar entre las direcciones 00h y 7Fh. Se puede poner la dirección o el nombre del registro.
Ej: BSF 03,5 equivale a BSF STATUS,5

w (working register)
W (w) acumulador.

b (bit addres dentro de un registro)
Define la dirección de un bit dentro de un byte.
Ej: BSF STATUS,5 equivale a BSF STATUS,RP

l (L) o k (literal) Es el valor almacenado que corresponde a la propia instrucción, está comprendido entre 0 y 255.

d (destiny bit)
Especifica donde se alamacenará el resultado de una instrucción, en w o en un registro. Si d=1: El resultado se almacenará en f
Si d=0: El resultado se almacenará en W

Formas de representación de las constantes.

Valor binario	ej. B'10110010'
Decimal	ej. D'65', .65
Hexadecimal	ej. H'E0' , 0xE0, E0H, E0
Octales	ej. O'37'
Codigo ASCII	ej. ' L'

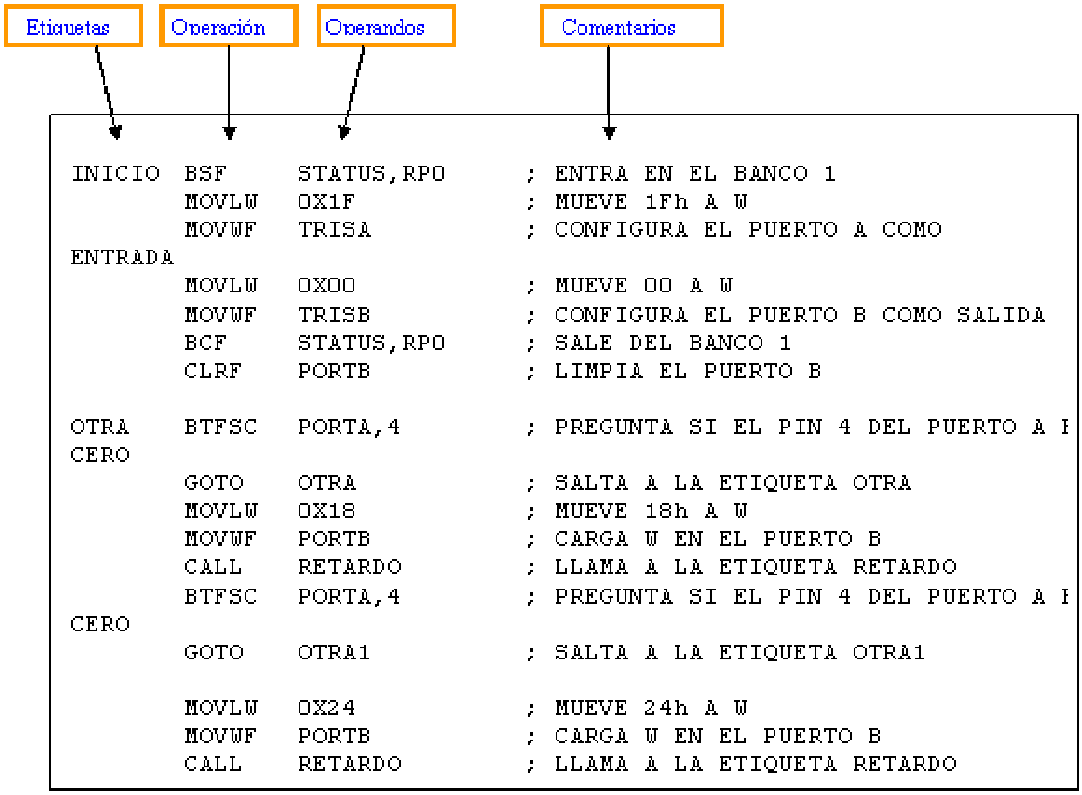
Programación del TRIS.
0 en Bit del TRIS -> Línea como Salida
1 en Bit del TRIS -> Línea como Entrada

Destino del resultado de una operación.
d = 0 ->El resultado se guarda en el acumulador W
d = 1 -> El resultado se guarda en el propio registro f

Formato de sentencia de instrucción .
Consta de cuatro campos:
[etiqueta] nombre_instrucción [operandos] [comentario].
Los corchetes, de acuerdo con la normativa informática, indican que el campo especificado es opcional, de acuerdo con la instrucción seleccionada.

Campo de etiqueta. Es el nombre simbólico de acceso a la posición de una instrucción.
Campo de instrucción. Contiene el mnemónico de cada instrucción, o de una directiva.
Campo de operándos. Contiene los datos asociados a la instrucción, y en referencia a los registros (f, l o k , b y w).
Campo de comentarios. Precedido punto y coma (;) es el comentario que se adjunta para poder conocer con mayor detalle la tarea a realizar por la instrucción.

Ejemplo de los campos de una instrucción




```
INICIO  BSF     STATUS,RP0      ; ENTRA EN EL BANCO 1
        MOVLW   0X1F            ; MUEVE 1Fh A W
        MOVWF   TRISA           ; CONFIGURA EL PUERTO A COMO
ENTRADA
        MOVLW   0X00            ; MUEVE 00 A W
        MOVWF   TRISB           ; CONFIGURA EL PUERTO B COMO SALIDA
        BCF     STATUS,RP0      ; SALE DEL BANCO 1
        CLRF    PORTB           ; LIMPIA EL PUERTO B

OTRA    BTFSC   PORTA,4         ; PREGUNTA SI EL PIN 4 DEL PUERTO A I
CERO
        GOTO    OTRA            ; SALTA A LA ETIQUETA OTRA
        MOVLW   0X18            ; MUEVE 18h A W
        MOVWF   PORTB           ; CARGA W EN EL PUERTO B
        CALL    RETARDO         ; LLAMA A LA ETIQUETA RETARDO
        BTFSC   PORTA,4         ; PREGUNTA SI EL PIN 4 DEL PUERTO A I
CERO
        GOTO    OTRA1           ; SALTA A LA ETIQUETA OTRA1

        MOVLW   0X24            ; MUEVE 24h A W
        MOVWF   PORTB           ; CARGA W EN EL PUERTO B
        CALL    RETARDO         ; LLAMA A LA ETIQUETA RETARDO
```

Listado de instrucciones del PIC 16F84

TABLE 7-2: PIC16CXXX INSTRUCTION SET

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 1fff 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 1fff 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`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

Descripción de las instrucciones para el PIC 16F84

ADDWF

Acción	Suma el contenido del acumulador y el registro dado, y el resultado lo guarda en d
Sintaxis	**ADDWF f,d**
Funcionamiento	Add W to file register (Añade W al registro)
Hexadecimal	07 ff
Bits (OPCODE)	00 0111 dfff ffff
Operación	d = W + f (d puede ser W ó f).
Descripción	Esta instrucción suma el contenido de un registro específico al contenido de W donde f puede ser un registro cualquiera con un determinado valor.
Comentarios	Aunque ya conocemos el funcionamiento del **bit d**, se repetirá de nuevo para el resto de instrucciones: - **Si vale 1,** el resultado se guarda en el registro **f** - **Si vale 0,** el resultado se guarda en el acumulador **W**
Registro STATUS	Modifica los bits Z, DC y C. Z vale 1 si el resultado de la operación es 0. DC vale 1 si el resultado de la operación es un número superior a 15. C vale 1 si el resultado de la operación es positivo o el bit 7 del registro que contiene el resultado vale 0. En caso contrario C vale 0 (resultado negativo).

Ejemplo	Tomamos como valores iniciales W = 5 y DATO = 10, donde dato es un registro cualquiera.
	ADDWF DATO ; DATO = 15 y W = 5.
	ADDWF DATO, 1 ; DATO = 15 y W = 5.
	ADDWF DATO, 0 ; W = 15 y DATO = 10.
	ADDWF DATO, W ; W = 15 y DATO = 10.

Ciclos de máquina	1

ANDWF

Acción	Realiza la operación AND entre un registro y W
Sintaxis	**ANDWF f,d**
Funcionamiento	AND W with f
Hexadecimal	05 ff
Bits (OPCODE)	00 0101 dfff ffff
Operación	d = W AND f (d puede ser W o f).
Descripción	Esta instrucción realiza la operación lógica AND entre el acumulador y el registro f. el resultado se guarda dependiendo del valor de d. Si este se omite, el valor por defecto es 1 y se guarda en f
Comentarios	La operación AND es una de las operaciones básicas del álgebra de Boole. Esta viene descrita en el apartado de electrónica digital. Para comprender de nuevo cual es su comportamiento, recojo en la siguiente tabla los valores de la tabla de verdad de esta operación. **ENTRADA** | | **SALIDA** **f** | **W** | **S** 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1 La operación lógica es: **S = f · W** Esta instrucción realiza esta operación para cada uno de los 8 bits de los dos registros, dos a dos, guardando el resultado en el registro correspondiente.
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo 1	Supongamos que W= 00001111 y f = 11110000 antes de ejecutar la instrucción ANDWF f,d **ENTRADA RESULTADO**

	W f W · f Bit 7 0 1 0 Bit 6 0 1 0 Bit 5 0 1 0 Bit 4 0 1 0 Bit 3 1 0 0 Bit 2 1 0 0 Bit 1 1 0 0 Bit 0 1 0 0
Ejemplo 2	Ahora en nuestro segundo ejemplo tenemos que W = 01110011 y f = 00101001 antes de ejecutar la instrucción ANDWF f,d tenemos: **ENTRADA RESULTADO** **W f W · f** Bit 7 0 0 0 Bit 6 1 0 0 Bit 5 1 1 1 Bit 4 1 0 0 Bit 3 0 1 0 Bit 2 0 0 0 Bit 1 1 0 0 Bit 0 1 1 1
Ciclos de máquina	1

CLRF

Acción	Borra un registro
Sintaxis	**CLRF f**
Funcionamiento	Clear file register
Hexadecimal	01 8f
Bits (OPCODE)	00 0001 1fff ffff
Operación	F = 0
Descripción	Esta instrucción borra un registro específico, poniendo sus bits a cero
Comentarios	Ninguno
Registro STATUS	Modifica el bit Z y lo pone a 1 (ya que el resultado de la operación es 0).
Ejemplo	Tenemos un registro que se llama dato y que vale 3F. Ponemos: CLRF dato Ahora dato vale 00
Ciclos de máquina	1

CLRW

Acción	Borra el acumulador
Sintaxis	**CLRW**

Funcionamiento	Clear W
Hexadecimal	01 8f
Bits (OPCODE)	00 0001 0xxx xxxx
Operación	W = 0
Descripción	Esta instrucción borra el registro W solamente
Comentarios	Donde pone xxx… en la instrucción en hexadecimal, significa que noo importa qué valor puede contener
Registro STATUS	Modifica el bit Z y lo pone a 1 (ya que el resultado de la operación es 0).
Ejemplo	Tenemos el acumulador cargado con el valor 3F. Ponemos: CLRF dato Ahora W vale 00
Ciclos de máquina	1

COMF

Acción	Complementa el registro F
Sintaxis	**COMF f,d**
Funcionamiento	Complement f
Hexadecimal	09 ff
Bits (OPCODE)	00 1001 dfff ffff
Operación	d = NOT f (d puede ser W ó f).
Descripción	Esta instrucción complementa un registro, es decir, los ceros los convierte en unos, y los unos en ceros.
Comentarios	ninguno
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo	Supongamos que tenemos un registro fdenominado regist = 00111011; cuando es aplicada la instrucción tenemos que los 0 cambian a valores 1 y los valores 1 cambian a 0 obteniéndose un registro invertido. El resultado será regist = 11000100.
Ciclos de máquina	1

DECF

Acción	Decrementa el registro f
Sintaxis	**DECF f,d**
Funcionamiento	Decrement f
Hexadecimal	03 ff
Bits (OPCODE)	00 0011 dfff ffff
Operación	d = f – 1 (d puede ser W ó f).
Descripción	Esta instrucción decrementa en una sola unidad el registro "f".
Comentarios	ninguno
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo	Nuestro registro se llama regist = 5; cuando se aplica la instrucción DECF f,0 el resultado será W=4. Por el contrario, si aplicamos la instrucción DECF f,1 el resultado será regist = 4.
Ciclos de máquina	1

DECFSZ

Acción	Decrementa el registro f, y si el resultado es cero, se salta una instrucción.
Sintaxis	**DECFSZ f,d**
Funcionamiento	Decrement f, skip if 0
Hexadecimal	0B ff

Bits (OPCODE)	00 1011 dfff ffff
Operación	d = f – 1, si d = 0 SALTA (d puede ser W ó f).
Descripción	Esta instrucción decrementa el contenido del registro direccionado por el parámetro f, y si el resultado es 0 salta la instrucción siguiente. Si no, sigue con su curso habitual
Comentarios	Aquí nos enfrentamos ante la primera instrucción que plantea una condición, y que modifica el curso del PC.
Registro STATUS	No modifica ningún bit de estado
Ejemplo	DECFSZ VALOR, W INSTRUCCION 1 INSTRUCCIÓN 2 Si el contenido del registro VALOR al decrementarlo es igual a 0, se guarda el resultado en el acumulador y sigue con la INSTRUCCION2, saltándose la INSTRUCCION1. Si el resultado que guardamos en W no es 0, sigue con la INSTRUCCION1 y después con la INSTRUCCION2 (no se salta la inmediata siguiente).
Ciclos de máquina	1 (2) Si tiene que saltar ocupa dos ciclos

INCF

Acción	Suma una unidad al registro f
Sintaxis	**INCF f,d**
Funcionamiento	Increment f
Hexadecimal	0A ff
Bits (OPCODE)	00 1010 dfff ffff
Operación	d = f + 1 (d puede ser W ó f).
Descripción	Esta instrucción incrementa en una sola unidad el registro "f".
Comentarios	Ninguno
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo	Si tenemos un registro llamado DIA = 7. Aplicando la instrucción INCF DIA, 0, tendremos W = 8 y DIA = 7. Si aplicamos esta otra INCF DIA, 1, tendremos DIA = 8.
Ciclos de máquina	1

INCFSZ

Acción	Incrementa en 1 a f, y si f= 0 salta la siguiente instrucción
Sintaxis	**INCFSZ f,d**
Funcionamiento	Increment f, Skip if 0
Hexadecimal	0F ff
Bits (OPCODE)	00 1111 dfff ffff
Operación	d = f + 1, si d = 0 SALTA (d puede ser W ó f).
Descripción	Esta instrucción incrementa en una sola unidad el registro "f", en la cual si el resultado “d” es igual a cero, entonces salta la instrucción siguiente.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	Ejecutamos las siguiente sinstrucciones: INCFSZ VALOR, W ; el resultado se almacenará en W INSTRUCCION 1 ; salta aquí si W≠0 INSTRUCCIÓN 2 ; salta aquí si W=0 Si el contenido del registro VALOR es igual a 0 al incrementarlo en una unidad, se guarda el resultado en el acumulador y sigue con la INSTRUCCION2, saltándose la INSTRUCCION1. Si el resultado que guardamos en W no es 0, sigue con la INSTRUCCION1 y después con la INSTRUCCION2 (no se salta la

	inmediata siguiente).
Ciclos de máquina	1 (2) Si tiene que saltar ocupa dos ciclos

IORWF

Acción	Operación lógica OR entre el acumulador y un registro
Sintaxis	**IORWF f,d**
Funcionamiento	Inclusive Or W with F
Hexadecimal	04 ff
Bits (OPCODE)	00 0100 dfff ffff
Operación	d = W OR f (d puede ser W ó f).
Descripción	Esta instrucción realiza una operación lógica OR inclusivo entre el acumulador W y el registro direccionado por el parámetro f. El parámetro d determina donde se almacenará el resultado de la operación. Si no se pone nada, el valor por defecto es 1 y se guarda en f.
Comentarios	La operación OR inclusivo suele llamarse OR a secas, pero se pone así para diferenciarla de la Suma Exclusiva que veremos más adelante. La tabla de verdad de la suma lógica es la que sigue: **ENTRADA** | | **SALIDA** **f** | **W** | **S** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1 La operación lógica que describe esta instrucción es esta: **F + W = S** Se puede ver que basta con que uno de los dos registros tenga un uno para que la salida sea un uno también.
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo 1	Supongamos que W= 00001111 y f = 11110000 antes de ejecutar la instrucción IORWF f,d, y después obtenemos: **ENTRADA** | | | **RESULTADO** | **W** | **f** | **W + f** Bit 7 | 0 | 1 | 1 Bit 6 | 0 | 1 | 1 Bit 5 | 0 | 1 | 1 Bit 4 | 0 | 1 | 1 Bit 3 | 1 | 0 | 1 Bit 2 | 1 | 0 | 1 Bit 1 | 1 | 0 | 1 Bit 0 | 1 | 0 | 1
Ejemplo 2	En este segundo ejemplo tenemos que W = 01110011 y f = 00101001, antes de ejecutar la instrucción IORWF f,d, y después obtenemos que: **ENTRADA RESULTADO**

Bit 3 0 1 1

	Bit 3 0 1 1 Bit 2 0 0 0 Bit 1 1 0 1 Bit 0 1 1 1 Al igual que el caso anterior, el microcontrolador compara bit a bit los dos registros, dos a dos, obteniendo el resultado expresado en W + f
Ciclos de máquina	1

MOVF

Acción	Mueve el contenido de un registro al acumulador o al propio registro
Sintaxis	**MOVF f,d**
Funcionamiento	Move f
Hexadecimal	08 ff
Bits (OPCODE)	00 1000 dfff ffff
Operación	d = f (d puede ser W ó f).
Descripción	Esta instrucción mueve el contenido del registro f en el mismo registro f o en W. D determina el destino del resultado. Si no se pone nada, el valor por defecto es 1 y se guarda en f.
Comentarios	Se suele usar para mover datos al acumulador. El hecho de que se pueda mover sobre sí mismo no es otro que para mirar el resultado en el registro STATUS
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.

Ejemplo	Tenemos el registro EDAD = 38
	MOVF EDAD, 0 ; hace que W = 38.
	MOVF EDAD, 1 ; hace que EDAD = 38.
	MOVF EDAD, W ; hace que W = 38.
	MOVF EDAD ; hace que EDAD = 38.

Ciclos de máquina	1

MOVWF

Acción	Mueve el acumulador al registro f
Sintaxis	**MOVWF f**
Funcionamiento	Move W to f
Hexadecimal	00 ff
Bits (OPCODE)	00 0000 1fff ffff
Operación	f = W
Descripción	Esta instrucción copia el contenido del acumulador W en el registro direccionado por el parámetro f.
Comentarios	No existe el parámetro d (lógico, o no? En este caso no se puede copiar el acumulador sobre sí mismo :p)
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	Si queremos escribir el valor 10H en el registro TMR0, que está situado en la dirección 01h, tendremos que cargar primero el valor en el acumulador y después copiarlo al registro. MOVWF 10H ; cargar el valor 10H en el acumulador. MOVWF 01H ; copia el acumulador en la dirección 01H. O escrito de otra manera: MOVWF 10H ; cargar el valor 10H en el acumulador. MOVWF TMR0 ; copia el acumulador en el registro TMR0.
Ciclos de máquina	1

NOP

Acción	No opera

Sintaxis	**NOP**
Funcionamiento	No Operation
Hexadecimal	00 00
Bits (OPCODE)	00 0000 0xx0 0000
Operación	Ninguna
Descripción	Esta instrucción no reliza ninguna ejecución, pero sirve para gastar un ciclo de máquina, equivalente a 4 de reloj
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	Si usamos un cristal de cuarzo de 4 Mhz. en el oscilador, podremos obtener un retardo igual a un microsegundo por cada instrucción NOP que insertemos en el código del programa: RETARDO NOP NOP NOP RETURN Cada vez que llamemos a la subrutina RETARDO, obtendremos 3 microsegundos de demora.
Ciclos de máquina	1

RLF

Acción	Rota a la izquierda el registro f
Sintaxis	**RLF f,d**
Funcionamiento	Rotate Left through Carry f
Hexadecimal	0D ff
Bits (OPCODE)	00 1101 dfff ffff
Operación	d = << 1 (d puede ser W ó f).
Descripción	Esta instrucción rota a la izquierda todos los bits del registro direccionado por el parámetro fpasando por el bit CARRY del registro STATUS (desde los bits menos significativos a los más significativos). Es como si multiplicáramos por dos el contenido del registro. Veamos el registro fde forma gráfica: El bit D7 pasa al CARRY del registro STATUS, el contenido del CARRY pasa al D0, el D0 al D1, etc.
Comentarios	Ninguno
Registro STATUS	Modifica el bit **C** (**CARRY**).
Ejemplo	Tenemos el registro VALOR = 00000001 y aplicamos la instrucción RLF VALOR Entonces el resultado será VALOR = 00000010 y el bit C = 0. Si tenemos el registro VALOR = 10000000 y aplicamos la instrucción RLF VALOR El resultado será VALOR = 00000000 y el bit C = 1.
Ciclos de máquina	1

RRF

Acción	Rota a la derecha el registro f
Sintaxis	**RRF f,d**
Funcionamiento	Rotate Right through Carry f
Hexadecimal	0C ff
Bits (OPCODE)	00 1100 dfff ffff
Operación	d = f >> 1 (d puede ser W ó f).
Descripción	Esta instrucción rota a la derecha todos los bits del registro direccionado por el parámetro fpasando por el bit CARRY del registro STATUS (desde los bits más significativos a los menos significativos). Es como si dividiéramos por dos el contenido del registro. Veamos el registro fde forma gráfica: El bit C del registro STATUS pasa al D7, el D0 pasa al bit C, el D1 al D0,

	etc. El bit d determina el destino del resultado. Si no se pone nada, el valor por defecto es 1 y se guarda en f.
Comentarios	Ninguno
Registro STATUS	Modifica el bit **C** (**CARRY**).
Ejemplo	Si tenemos el registro VALOR = 00000001 y aplicamos la instrucción RRF VALOR Entonces el resultado será VALOR = 00000000 y el bit C = 1. Si tenemos el registro VALOR = 10000000 y aplicamos la instrucción RRF VALOR El resultado será VALOR = 01000000 y el bit C = 0.
Ciclos de máquina	1

SUBWF

Acción	Resta el contenido del registro W el registro f
Sintaxis	**SUBWF f,d**
Funcionamiento	Subtract W from f
Hexadecimal	02 ff
Bits (OPCODE)	00 0010 dfff ffff
Operación	d = f – W (d puede ser W ó f).
Descripción	Esta instrucción resta el valor contenido en el acumulador W del valor contenido en el registro direccionado por el parámetro f. El parámetro ddetermina el destino. Si no se pone nada el valor por defecto será 1 y se almacenará en f.
Comentarios	Ninguno
Registro STATUS	Modifica los bits Z, DC y C. Z vale 1 si el resultado de la operación es 0. DC vale 1 si el resultado de la operación es un número superior a 15. C vale 1 si el resultado de la operación es positivo o el bit 7 del registro que contiene el resultado vale 0. En caso contrario C vale 0 (resultado negativo).
Ejemplo	Según sean los valores de W y el registro DATO, si aplicamos SUBWF DATO obtendremos diferentes resultados en el bit CARRY. Si DATO = 3 y W = 2; el resultado será DATO = 1 y C = 1. Si DATO = 2 y W = 2; el resultado será DATO = 0 y C = 1. Si DATO = 1 y W = 2; el resultado será DATO = FFH y C = 0. Vemos que C = 1 porque el resultado es positivo y C = 0 cuando el resultado es negativo.
Ciclos de máquina	1

SWAPF

Acción	Invierte los dos nibbles que forman un byte dentro de un registro
Sintaxis	**SWAPF f,d**
Funcionamiento	Swap nibbles in f
Hexadecimal	0E ff
Bits (OPCODE)	00 1110 dfff ffff
Operación	f = 0123 SWAP 4567 de f
Descripción	Esta instrucción intercambia el valor de los 4 bits más significativos (D7-D4) contenidos en el registro f, con los 4 bits menos significativos (D3-D0) del mismo. El parámetro d determina el destino. Si no se pone nada, el valor por defecto es 1 y se guarda en f.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	Tenemos VALOR = 00001111 y W = 00000000. Aplicamos la instrucción SWAPF SWAPF VALOR ; VALOR = 11110000B y W = 00000000B.

	SWAPF VALOR, W ; VALOR = 00001111B y W = 11110000B. Con la primera instrucción modificamos el valor del registro DATO, y en la segunda instrucción modificamos el valor del acumulador sin que varíe el registro DATO
Ciclos de máquina	1

XORWF

Acción	Operación lógica OR-Exclusiva
Sintaxis	**XORWF f,d**
Funcionamiento	Exclusive OR W with f
Hexadecimal	06 ff
Bits (OPCODE)	00 0110 dfff ffff
Operación	d = W OR f
Descripción	Esta instrucción efectúa la operación lógica XOR (OR exclusivo) entre el valor contenido en el acumulador W y el valor contenido en el registro direccionado por el parámetro f. El parámetro d determina el destino. Si no se pone nada el valor por defecto es 1 y se guarda en f
Comentarios	Al igual que las otras operaciones lógicas, la suma exclusiva está tratada en el capítulo de electrónica digital. No obstante expongo las tablas de verdad correspondientes a esta operación

ENTRADA		**SALIDA**
f	**W**	**S**
0	0	0
0	1	1
1	0	1
1	1	0

La salida únicamente se podrá a nivel alto cuando las dos entradas sean distintas. Esto es útil cuando tenemos una suma lógica en la que 1 + 1 es 10 y nos llevamos 1.
Esta operación algebraicamente se expresa así:
S = f + W

Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo 1	Tenemos dos registros que se corresponden con los siguientes valores W= 00001111 y f = 11110000 antes de ejecutar la instrucción XORWF f,d. Una vez ejecutada obtenemos la siguiente tabla de verdad sobre los dos registros:

	ENTRADA		**RESULTADO**
	W	**f**	**W + f**
Bit 7	0	1	1
Bit 6	0	1	1
Bit 5	0	1	1
Bit 4	0	1	1
Bit 3	1	0	1
Bit 2	1	0	1
Bit 1	1	0	1
Bit 0	1	0	1

El PIC compara dos a dos los bits de los registros

Ejemplo 2	Ahora W = 01110011 y f = 00101001 antes de ejecutar la instrucción XORWF f,d y después obtenemos:

	ENTRADA		**RESULTADO**
Bit 7	0	0	0

	Bit 7 0 0 0 Bit 6 1 0 1 Bit 5 1 1 1 Bit 4 1 0 1 Bit 3 0 1 1 Bit 2 0 0 0 Bit 1 1 0 1 Bit 0 1 1 0
Ciclos de máquina	1

BCF

Acción	Pone a cero el bit b del registro f
Sintaxis	**BCF f,b**
Funcionamiento	Bit Clear f
Hexadecimal	1b ff
Bits	01 00bb bfff ffff
Operación	F(b) = 0
Descripción	Esta instrucción pone a cero un bit que hayamos elegido de un registro determinado.
Comentarios	No debemos olvidar que en la numeración de los bits también se tiene en cuenta el 0. Si tratamos con un registro especial, podemos poner el nombre del bit correspondiente.
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	BCF PORTA, RA4 ; pone a 0 el bit RA4 del registro PORTA BCF PORTA, 4 ; igual, si no conocemos en nombre del bit Si en el PORTA tenemos como valor inicial 11111111, después de aplicar el ejemplo anterior, PORTA = 11101111.
Ciclos de máquina	1

BSF

Acción	Pone a uno el bit b del registro f
Sintaxis	**BSF f,b**
Funcionamiento	Bit Set f
Hexadecimal	1b ff
Bits (OPCODE)	01 01bb bfff ffff
Operación	F(b) = 1
Descripción	Esta instrucción pone a uno un bit que hayamos elegido de un registro determinado.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	BSF PORTA, RA0 ; pone a 1 el bit RA0 del registro PORTA BSF PORTA, 0 ; igual, si no conocemos en nombre del bit Si en el PORTA tenemos como valor inicial 00000000, después de aplicar el ejemplo anterior, PORTA = 00000001.
Ciclos de máquina	1

BTFSC

Acción	Comprueba un bit b del registro f, y salta la instrucción siguiente si este es cero
Sintaxis	**BTFSC f,b**
Funcionamiento	Bit Test, Skip if Clear

Hexadecimal	1b ff
Bits (OPCODE)	01 10bb bfff ffff
Operación	F(b) = 0? SI, salta una instrucción
Descripción	Esta instrucción comprueba el valor del bit b en el registro f, y si b = 0 entonces se salta la siguiente instrucción. Si b = 1 no salta y sigue con su ejecución normal.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado
Ejemplo	BTFSC PORTA, 2 INSTRUCCIÓN 1 INSTRUCCIÓN 2 Si en PORTA tenemos como valor inicial 11111011, el programa continúa con la instrucción 2, saltándose la instrucción 1 Si en PORTA tenemos el valor 00000100, el programa sigue con la instrucción 1 y después la instrucción 2
Ciclos de máquina	1 (2) Si tiene que saltar ocupa dos ciclos

BTFSS

Acción	Comprueba un bit b del registro f, y salta la instrucción siguiente si este es uno
Sintaxis	**BTFSC f,b**
Funcionamiento	Bit Test, Skip if Set
Hexadecimal	1b ff
Bits (OPCODE)	01 11bb bfff ffff
Operación	F(b) = 1? SI, salta una instrucción
Descripción	Esta instrucción comprueba el valor del bit b en el registro f, y si b = 1 entonces se salta la siguiente instrucción. Si b = 0 no salta y sigue con su ejecución normal.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado
Ejemplo	BTFSS PORTB, 7 INSTRUCCIÓN 1 INSTRUCCIÓN 2 Si en PORTB tenemos como valor inicial 10000000, el programa continúa con la instrucción 2, saltándose la instrucción 1. Si tenemos el valor 01111111, el programa sigue con la instrucción 1 y después la instrucción 2.
Ciclos de máquina	1 (2) Si tiene que saltar ocupa dos ciclos

OPERACIONES CON LITERALES Y DE CONTROL

ADDLW

Acción	Suma a W un literal
Sintaxis	ADDLW
Funcionamiento	Add literal to W
Hexadecimal	3E kk
Bits (OPCODE)	11 111x kkkk kkkk
Operación	W = W + k
Descripción	Esta instrucción suma un valor de un literal al contenido del registro W y lo guarda en W.
Comentarios	Es igual que su homologo manejando registros
Registro STATUS	Modifica los bits Z, DC y C. Z vale 1 si el resultado de la operación es 0. DC vale 1 si el resultado de la operación es un número superior a 15. C vale 1 si el resultado de la operación es positivo o el bit 7 del registro que contiene el resultado vale 0. En caso contrario C vale 0 (resultado negativo).

Ejemplo	MOVLW 3 ; carga el acumulador W con el valor 3. ADDLW 1 ; suma 1 al acumulador. Al final el acumulador tendrá el valor 4.
Ciclos de máquina	1

ANDLW

Acción	Realiza la operación AND entre un literal y W
Sintaxis	**ANDLW k**
Funcionamiento	AND W with k
Hexadecimal	39 ff
Bits (OPCODE)	11 1001 kkkk kkkk
Operación	W = W AND k
Descripción	Esta instrucción realiza una operación lógica AND entre el contenido de W y k. El resultado se guarda siempre en el acumulador W
Comentarios	ninguno
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo	Si cargamos el acumulador con el número binario 10101010 y hacemos un AND con el binario 11110000, nos quedará el resultado de la operación en el acumulador W. MOVLW 10101010 ANDLW 11110000 El resultado de la operación queda en W = 10100000.
Ciclos de máquina	1

CALL

Acción	Llama a una subrutina en la dirección k
Sintaxis	**CALL k**
Funcionamiento	Call subroutine
Hexadecimal	2k kk
Bits (OPCODE)	10 0kkk kkkk kkkk
Operación	CALL → k...RETURN → PC+1.
Descripción	Esta instrucción llama a un grupo de instrucciones (subrutina) que comienzan en la dirección k, donde k puede ser un valor numérico o una etiqueta. Siempre termina con la instrucción de retorno (RETURN o RETLW). Definición de subrutina: son un grupo de instrucciones que forman un programa dentro del programa principal y que se ejecutan cuando las llama el programa principal. Utilidad: sirven para utilizarlas varias veces en cualquier parte del programa, sin necesidad de tener que copiar las mismas instrucciones, con el consiguiente ahorro de memoria. Funcionamiento: cuando un programa ejecuta una instrucción CALL, guarda en el stack el valor del registro PC+1 (PC = Program Counter) de manera que al regresar de la subrutina continúa con la instrucción siguiente recuperándola del stack, ejecutando la instrucción de retorno RETURN o RETLW. Limitaciones: en el PIC16F84 tenemos disponibles 8 niveles de stack, por lo que el número máximo de CALL reentrantes (instrucciones CALL que contengan otra instrucción CALL) queda limitado a 8.
Comentarios	Definición de subrutina: son un grupo de instrucciones que forman un programa dentro del programa principal y que se ejecutan cuando las llama el programa principal. Utilidad de las subritunas: sirven para utilizarlas varias veces en cualquier parte del programa, sin necesidad de tener que copiar las mismas instrucciones, con el consiguiente ahorro de memoria.

	Funcionamiento: cuando un programa ejecuta una instrucción CALL, guarda en el stack el valor del registro PC+1 (PC = Program Counter) de manera que al regresar de la subrutina continúa con la instrucción siguiente recuperándola del stack, ejecutando la instrucción de retorno RETURN o RETLW. Limitaciones: en el PIC16F84 tenemos disponibles 8 niveles de stack, por lo que el número máximo de CALL reentrantes (instrucciones CALL que contengan otra instrucción CALL) queda limitado a 8.
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	PRINCIPAL: etiqueta que identifica una dirección de memoria. RETARDO: etiqueta que identifica el comienzo de una subrutina. BUCLE: etiqueta que identifica una dirección de memoria. PRINCIPAL CALL RETARDO BTFSC PORTB, RB0 GOTO PRINCIPAL * * * RETARDO CLRF CONTADOR BUCLE DECFSZ CONTADOR, 1 GOTO BUCLE RETURN En este listado vemos que la subrutina RETARDO salta a un grupo de instrucciones que forman un bucle y cuando éste termina regresa para seguir con la instrucción siguiente al salto (BTFSC...).
Ciclos de máquina	2

CLRWDT

Acción	Pone el temporizador WDT a cero.
Sintaxis	**CLRWDT**
Funcionamiento	Clear WatchDog Timer
Hexadecimal	00 64
Bits (OPCODE)	00 0000 0110 0100
Operación	WDT = 0
Descripción	Esta instrucción se utiliza cuando programamos el PIC con la opción Watch Dog habilitada. Para evitar el reset del PIC, el programa debe contener cíclicamente la instrucción CLRWDT para ponerlo a cero. Si no se pone a cero a tiempo, el WDT interpretará que se ha bloqueado el programa y ejecutará un reset para desbloquearlo.
Comentarios	ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	Bucle CLRWDT * * * GOTO Bucle
Ciclos de máquina	1

GOTO

Acción	Salto incondicional a k.
Sintaxis	**GOTO k**
Funcionamiento	Go to address (label)
Hexadecimal	28 kk
Bits (OPCODE)	10 1kkk kkkk kkkk

Operación	Salto → k
Descripción	Esta instrucción ejecuta un salto del programa a la dirección k. El parámetro k puede ser un valor numérico o una etiqueta.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	INSTRUCCIÓN 1 GOTO ABAJO INSTRUCCIÓN 3 INSTRUCCIÓN 4 INSTRUCCIÓN 5 ABAJO INSTRUCCIÓN 6 Primero se ejecuta la instrucción 1, después GOTO y continúa con la instrucción 6 saltándose las instrucciones 3, 4 y 5.
Ciclos de máquina	2

IORLW

Acción	Operación lógica OR entre el acumulador y un literal
Sintaxis	**IORWF f,d**
Funcionamiento	Inclusive OR W with I
Hexadecimal	
Bits (OPCODE)	
Operación	W = W OR I
Descripción	Esta instrucción realiza una operación lógica OR inclusivo entre el acumulador W y un literal. El resultado siempre se guarda en el acumulador.
Comentarios	La operación OR inclusivo suele llamarse OR a secas, pero se pone así para diferenciarla de la Suma Exclusiva que veremos más adelante. La tabla de verdad de la suma lógica es la que sigue: **ENTRADA** | | **SALIDA** **f** | **W** | **S** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1 La operación lógica que describe esta instrucción es esta: **F + W = S** Se puede ver que basta con que uno de los dos registros tenga un uno para que la salida sea un uno también.
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo	Supongamos que W= 00001111 y I = f0. Ejecutanado la instrucción IORWF f,d, obtenemos: **ENTRADA** | **RESULTADO** | **W** | **I** | **W + I** Bit 7 | 0 | 1 | 1 Bit 6 | 0 | 1 | 1 Bit 5 | 0 | 1 | 1 Bit 4 | 0 | 1 | 1 Bit 3 | 1 | 0 | 1 Bit 2 | 1 | 0 | 1 Bit 1 | 1 | 0 | 1 Bit 0 | 1 | 0 | 1
Ciclos de	2

máquina	

MOVLW

Acción	Copia el contenido de un literal al acumulador
Sintaxis	**MOVLW f**
Funcionamiento	Move literal to W
Hexadecimal	30 kk
Bits (OPCODE)	11 00xx kkkk kkkk
Operación	W = f
Descripción	Esta instrucción asigna al acumulador W el valor del literal k, el cual debe estar comprendido entre 0 y 255.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	Si tenemos el acumulador a cero o con cualquier valor, y queremos que contenga el que le asignemos nosotros directamente entonces usaremos esta instrucción: W = 0. Valor a asignar = 100. Instrucción: MOVLW 100 El acumulador valdrá 100 (W = 100). Con distinto valor de partida del acumulador: W = 225. MOVLW 100 El acumulador valdrá 100 (W = 100).
Ciclos de máquina	1

RETFIE

Acción	Retorno de la llamada a una subrutina
Sintaxis	**RETFIE**
Funcionamiento	Return From Interrupt
Hexadecimal	00 09
Bits (OPCODE)	00 0000 0000 1001
Operación	FIN INTERRUPCIÓN
Descripción	Esta instrucción devuelve el control al programa principal después de ejecutarse una subrutina de gestión de una interrupción.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado
Ejemplo	ORG 00H BUCLE GOTO BUCLE ; bucle infinito. ORG 04H; vector de interrupción. RETFIE ; retorna de la interrupción Este código de programa ejecuta un bucle infinito. Si habilitamos una de las interrupciones del 16F84, en cuanto ésta se produzca pasará el control al programa situado en la dirección 04h y la instrucción RETFIE regresa de la interrupción. Al ejecutarse una interrupción, el bit GIE del registro INTCON se pone a 0 y así evita que otra interrupción se produzca mientras ya está con una en marcha. Con la instrucción RETFIE ponemos de nuevo el bit GIE a 1 para así atender de nuevo a futuras interrupciones.
Ciclos de máquina	2

RETLW

Acción	Retorno de subrutina y carga literal k en el acumulador
Sintaxis	**RETLW**

Funcionamiento	Return with Literal in W
Hexadecimal	34 kk
Bits (OPCODE)	11 01xx kkkk kkkk
Operación	RETORNO con W = k
Descripción	Esta instrucción retorna de una subrutina al programa principal, cargando el acumulador W con el literal k. Es la última instrucción que forma una subrutina, al igual que RETURN, con la diferencia que carga en W el valor de k.
Comentarios	¿Y para qué me sirve regresar de una subrutina con un determinado literal en el acumulador? Nos será muy útil al programar con TABLAS.
Registro STATUS	No modifica ningún bit de estado
Ejemplo	CALL SUBRUT1 ; llama a Subrut1. MOVWF DATO 1 ; carga W en Dato1. CALL SUBRUT2 ; llama a Subrut2. MOVWF DATO2 ; carga W en Dato2. * * SUBRUT1 RETLW 0A ; carga W = 0A y retorna. SUBRUT2 RETLW 0B ; carga W = 0B y retorna.
Ciclos de máquina	2

RETURN

Acción	Retorno de una subrutina.
Sintaxis	**RETURN**
Funcionamiento	Return from subroutine
Hexadecimal	00 08
Bits (OPCODE)	00 0000 0000 1000
Operación	RETORNO
Descripción	Esta instrucción retorna de una subrutina al programa principal en la instrucción siguiente a la llamada de la subrutina, tomando el valor almacenado en el stack para continuar. Es la última instrucción que forma una subrutina (al igual que RETLW).
Comentarios	El procedimiento es siempre el mismo. Así, tenemos que crear la subrrutina y darle el nombre para poder ser llamada; al final de la subrrutina se debe escribir la instrucción denominada RETURN. Entonces podemos concluir que una subrrutina esta constituida por un conjunto de instrucciones demarcadas por un nombre que se encuentra al inicio y la instrucción RETURN que se encuentra al final demarcando el final de la subrrutina. Estos mismos pasos debemos seguirlos para la instrucción RETLW
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	CALL COMPARA ; llama a Compara. INSTRUCCION1 ; vuelve aquí cuando se INSTRUCCION2 ; ejecuta return * * COMPARA INSTRUCCIÓN R1 INSTRUCCIÓN R2 RETURN Aquí llamamos a la subrutina COMPARA, se ejecutan las instrucciones R1 y R2 y con el RETURN regresa a la instrucción siguiente al CALL y ejecuta las instrucciones 1, 2 y sigue con el programa.
Ciclos de máquina	2

SLEEP

Acción	Paso a modo de bajo consumo
Sintaxis	**SLEEP**

Funcionamiento	Go into Standby Mode
Hexadecimal	00 63
Bits (OPCODE)	00 0000 0110 0011
Operación	EN ESPERA.
Descripción	Esta instrucción detiene la ejecución del programa, deja el PIC en modo suspendido y el consumo de energía es mínimo. No ejecuta ninguna instrucción hasta que sea nuevamente reinicializado (reset) o surja una interrupción. Durante este modo, el contador del Watch Dog sigue trabajando, y si lo tenemos activado el PIC se reseteará por este medio.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	No creo que haga falta… ;)
Ciclos de máquina	1

SUBLW

Acción	Resta al literal k el valor del acumulador.
Sintaxis	**SUBLW k**
Funcionamiento	Substract W from Literal
Hexadecimal	3C kk
Bits (OPCODE)	11 110x kkkk kkkk
Operación	W = k - W
Descripción	Esta instrucción resta al literal k el valor almacenado en W y el resultado se guarda en el acumulador.
Comentarios	ninguno
Registro STATUS	Modifica los bits Z, DC y C. Z vale 1 si el resultado de la operación es 0. DC vale 1 si el resultado de la operación es un número superior a 15. C vale 1 si el resultado de la operación es positivo o el bit 7 del registro que contiene el resultado vale 0. En caso contrario C vale 0 (resultado negativo).
Ejemplo	MOVLW 10 ; carga el acumulador W con el valor 10. SUBLW 15 ; resta a 15 el valor del acumulador. Al final el acumulador tendrá el valor W = 5.
Ciclos de máquina	1

XORLW

Acción	Operación lógica OR exclusivo entre el acumulador y el literal k
Sintaxis	**XORLW k**
Funcionamiento	Exclusive OR Literal with W
Hexadecimal	3A kk
Bits (OPCODE)	11 1010 kkkk kkkk
Operación	W = W XOR k
Descripción	Esta instrucción realiza un OR exclusivo entre el contenido del acumulador W y el valor del literal k. El resultado se guarda siempre en el acumulador (recuerda que k es un literal, no un registro).
Comentarios	Al igual que las otras operaciones lógicas, la suma exclusiva está tratada en el capítulo de electrónica digital. No obstante expongo las tablas de verdad correspondientes a esta operación

ENTRADA		**SALIDA**
f	**W**	**S**
0	0	0
0	1	1
1	0	1
1	1	0

La salida únicamente se podrá a nivel alto cuando las dos entradas sean distintas. Esto es útil cuando tenemos una suma lógica en la que 1 + 1

	es 10 y nos llevamos 1. Esta operación algebraicamente se expresa así: **S = f + W**
Registro STATUS	Modifica el bit Z. Z vale 1 si el resultado de la operación es 0.
Ejemplo	W = 01110011 y k = 00101001 antes de ejecutar la instrucción XORWF k **ENTRADA RESULTADO** **W k W + k** Bit 7 0 0 0 Bit 6 1 0 1 Bit 5 1 1 1 Bit 4 1 0 1 Bit 3 0 1 1 Bit 2 0 0 0 Bit 1 1 0 1 Bit 0 1 1 0
Ciclos de máquina	1

INSTRUCCIONES DE LA GAMA BAJA

Entre estas instrucciones no se han incluido dos rutinas que aparte de no pertenecer a las 35 instrucciones de la gama media, no pueden ser clasificadas en ninguna de las categorías expuestas anteriormente, aunque normalmente son acogidas dentro de las instrucciones con literales y de control. Estas instrucciones son OPTION y TRIS. La razón por la cual no pertenecen a estas 35 instrucciones es por que fueron creadas pensando en la gama baja, ya que carece de 4 de las instrucciones de la gama media: ADDLW, RETFIE, RETURN y SUBLW. No por ello se ha prohibido a la gama media disponer de estas instrucciones:

OPTION

Acción	Guarda el valor del acumulador en el registro OPTION
Sintaxis	**OPTION**
Funcionamiento	-
Hexadecimal	00 62
Bits (OPCODE)	00 0000 0110 0010
Operación	OPTION = W
Descripción	Esta instrucción guarda en el registro especial OPTION el valor contenido en el acumulador W
Comentarios	ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	MOVLW 10H ; carga el acumulador con el valor 10h. OPTION ; carga el registro OPTION con el acumulador. Esta instrucción existe para mantener la compatibilidad con los PIC producidos con anterioridad, y como en el futuro podría dejar de implementarse, Microchip aconseja realizar el ejemplo anterior de esta otra forma: BSF STATUS, RP0 ; activa el banco 1. MOVLW 10H ; carga el acumulador con 10h MOVWF OPTION_REG ; carga OPTION con el acumulador.
Ciclos de máquina	1

TRIS

Acción	Guarda el acumulador en uno de los registros de TRIS.

Sintaxis	**TRIS f**
Funcionamiento	-
Hexadecimal	00 6F
Bits (OPCODE)	00 0000 0110 1111
Operación	TRIS de f = W.
Descripción	Esta instrucción guarda el valor del acumulador W en uno de los registros especiales de TRIS que indicamos en el parámetro f. Los registros TRIS determinan el funcionamiento como entrada y salida de las líneas I/O del PIC.
Comentarios	ninguno
Registro STATUS	No modifica ningún bit de estado.
Ejemplo	MOVLW 16h ; carga el acumulador W con el valor 16h TRIS PORTA ; carga el registro PORTA con el acumulador. Esta instrucción existe para mantener la compatibilidad con los PIC producidos anteriormente, y como en el futuro podría dejar de implementarse, Microchip aconseja realizar el ejemplo anterior de esta otra forma (aunque ocupa más memoria...): BSF STATUS, RP0 ; activa el banco 1. MOVLW 16h ; carga el acumulador con el valor 16h MOVWF TRISA ; carga el registro PORTA con W.
Ciclos de máquina	1

INSTRUCCIONES ESPECIALES

Existe un conjunto de instrucciones especiales diseñadas para facilitar las operaciones a la hora de diseñar nuestros algoritmos. Estas instrucciones pueden ser implementadas con una, dos o tres de las instrucciones de la gama media. La mayoría de ellas se basa en las operaciones con los acarreos y con los bits del registro status en general. Este cuadro sólo debe servir de referencia, y no debemos usarlo en el caso de que estemos empezando. Sólo lo usaremos si vemos muy claro el funcionamiento de las instrucciones, pero es recomendable usar la forma equivalente, que tiempo de acomodarnos ya tendremos.

Otra cosa que debemos tener en cuenta es que no por reducir algoritmos a una sola expresión, vamos a ahorrar ciclos de máquina.

Mnemónico	Parámetros	Descripción	Traducción	Operación Equivalente	Banderas
ADDCF	f, d	Add Carry to File	Sumar acarreo a f	BTFSC 3,0 INCF f,d	Z
ADDDCF	f, d	Add Digit Carry to File	Sumar acarreo de digito a f	BTFSC 3,1 INCF f,d	Z
B	K	Branch	Saltar a una etiqueta	GOTO k	-
BC	K	Branch on Carry	Saltar a una etiqueta si hay acarreo	BTFSC 3,0 GOTO k	-
BDC	K	Branch on Digit Carry	Saltar a una etiqueta si hay acarreo de digito	BTFSC 3,1 GOTO k	-
BNC	K	Branch on No Carry	Saltar a una etiqueta si no hay acarreo	BTFSS 3,0 GOTO k	-
BNDC	K	Branch on No Digit Carry	Saltar a una etiqueta si no hay acarreo de digito	BTFSS 3,1 GOTO k	-

BNZ	K	Branch on No Zero	Saltar a una etiqueta si no hay cero	BTFSS GOTO	3,2 k	-
BZ	K	Branch on Zero	Saltar a una etiqueta si hay cero	BTFSC GOTO	3,2 k	-
CLRC		Clear Carry	Poner a cero acarreo	BCF	3,0	-
CLRDC		Clear Digit Carry	Poner a cero acarreo de digito	BCF	3,1	-
CLRZ		Clear Zero	Poner a cero el flag Zero	BCF	3,2	-
LCALL	K	Long CALL	Llamada larga a una etiqueta	BSF/BCF 0A,3 BSF/BCF 0A,4 CALL k		- - -
LGOTO	K	Long GOTO	Salto largo a una etiqueta	BSF/BCF 0A,3 BSF/BCF 0A,4 GOTO		- - - k
MOVFW	F	Move File to W	Mover registro a W	MOVF	f,0	Z
NEGF	f, d	Negate File	Negar un registro	COMF INCF	f,1 f,d	Z
SETC		Set Carry	Poner a uno el acarreo	BSF	3,0	-
SETDC		Set Digit Carry	Poner a uno el acarreo de digito	BSF	3,1	-
SETZ		Set Zero	Poner a uno el Zero	BSF	3,2	-
SKPC		Skip on Carry	Saltar si hay acarreo	BTFSS	3,0	-
SKPDC		Skip on Digit Carry	Saltar si hay acarreo de digito	BTFSS	3,1	-
SKPNC		Skip on No Carry	Saltar si no hay acarreo	BTFSC	3,0	-
SKPNDC		Skip on No Digit Carry	Saltar si no hay acarreo de digito	BTFSC	3,1	-
SKPNZ		Skip on Non Zero	Saltar si no hay Zero	BTFSC	3,2	-
SKPZ		Skip on Zero	Saltar si hay Zero	BTFSS	3,2	-
SUBCF	f,d	Substract Carry from File	Restar acarreo del registro	BTFSC DECF	3,0 f,d	Z
SUBDCF	f,d	Substract Digit Carry from File	Restar acarreo de dígito del	BTFSC DECF	3,1 f,d	Z

			registro			
TSTF	f	Test File	Probar registro	MOVF	f,1	Z

Resumen del cuadro de instrucciones

Instrucciones orientadas a los Bytes

Mnemónico	Parámetros	Descripción	Ciclos	Banderas
ADDWF	f, d	Add W and f	1	C, DC, Z
ANDWF	f, d	AND W with f	1	Z
CLRF	f	Clear f	1	Z
CLRW	-	Clear W	1	Z
COMF	f, d	Complement f	1	Z
DECF	f, d	Decrement f	1	Z
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	None
INCF	f, d	Increment f	1	Z
INCFSZ	f, d	Increment f, Skip if 0	1(2)	None
IORWF	f, d	Inclusive OR W with f	1	Z
MOVF	f, d	Move f	1	Z
MOVWF	f	Move W to f	1	None
NOP	-	No Operation	1	None
RLF	f, d	Rotate left f through carry	1	C
RRF	f, d	Rotate right f through carry	1	C
SUBWF	f, d	Subtract W from f	1	C, DC, Z
SWAPF	f, d	Swap nibbles in f	1	None
XORWF	f, d	Exclusive OR W with f	1	Z

Instrucciones orientadas a Bits

Mnemónico	Parámetros	Descripción	Ciclos	Banderas
BCF	f, b	Bit Clear f	1	None
BSF	f, b	Bit Set f	1	None
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	None
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	None

Operaciones con literales y de control

Mnemónico	Parámetros	Descripción	NroCic.	Banderas
ADDLW	k	Add literal and W	1	C, DC, Z
ANDLW	k	AND literal with W	1	Z
CALL	k	Call subroutine	2	
CLRWDT	-	Clear Watchdog Timer	1	TO,PD
GOTO	k	Go to address	2	None
IORLW	k	Inclusive OR literal with W	1	Z
MOVLW	k	Move literal to W	1	None
RETFIE	-	Return from interrupt	2	None
RETLW	k	Return with literal in W	2	None
RETURN	-	Return from Subroutine	2	None

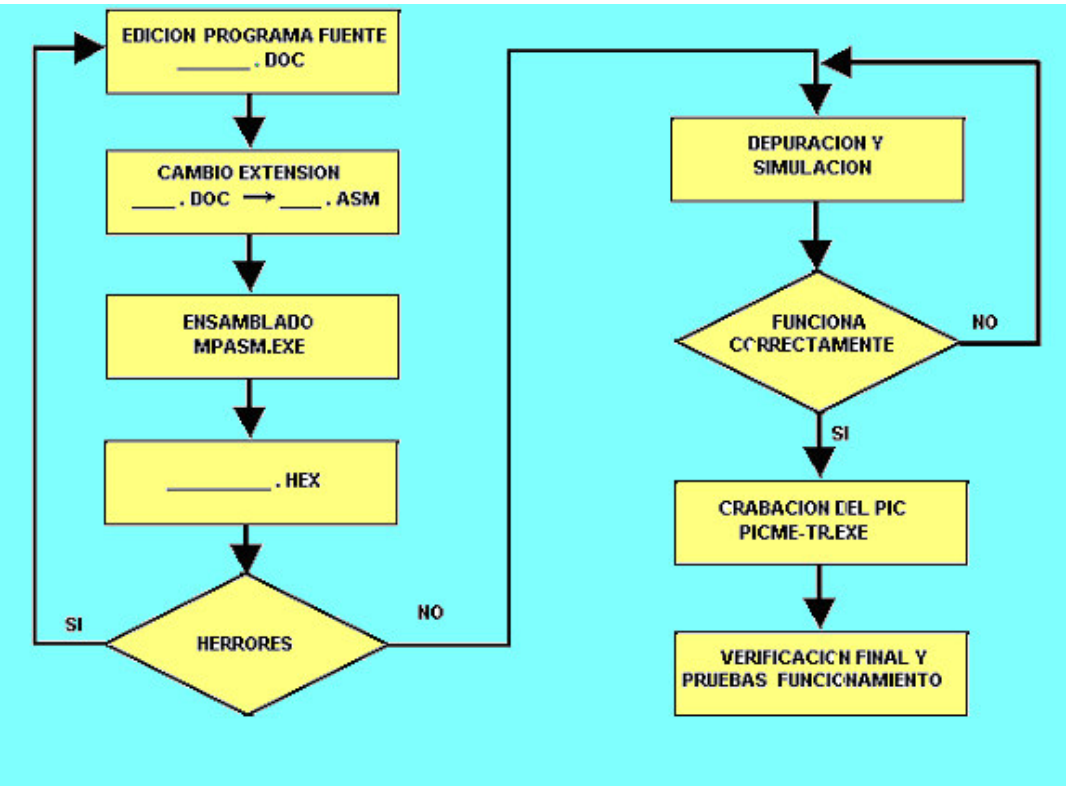
SLEEP	-	Go into standby mode	1	TO,PD
SUBLW	k	Subtract W from literal	1	C, DC, Z
XORLW	k	Exclusive OR literal with W	1	Z

Microchip recomienda no utilizar las instrucciones **TRIS** y **OPTION**, con el fin de mantener la compatibilidad con el PIC16CXX.

CAPITULO 4

EDICIÓN EN ENSAMBLADOR Y GRABACION

Procedimiento de grabación del PIC.

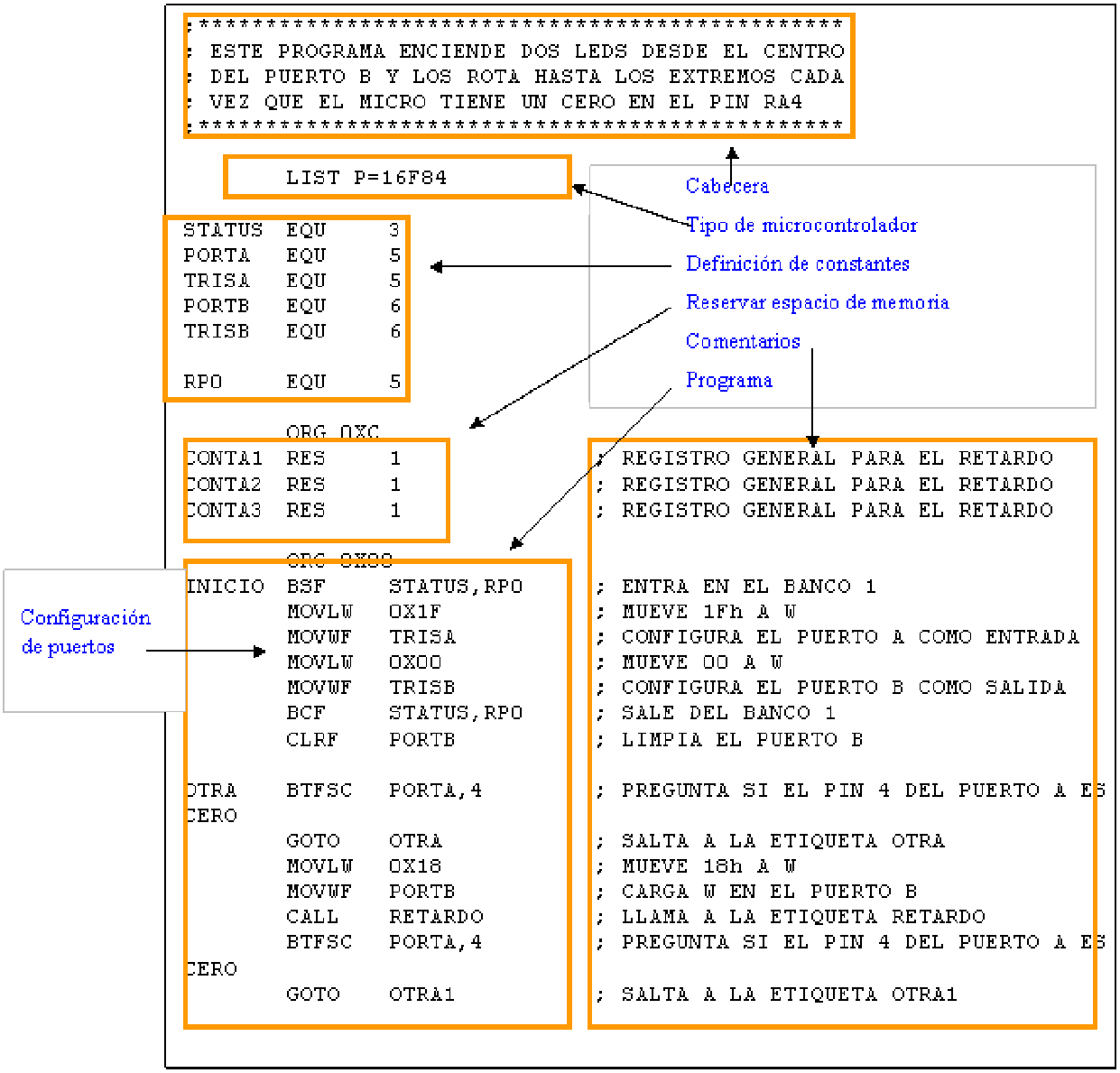


ORGANIGRAMA DEL PROCEDIMIENTO DE PROGRAMACIÓN DEL PIC

Proceso de grabación del PIC
-La edición del programa fuente puede realizarse con un editor de texto, Ej. **--------.DOC**
-Se debe cambiar la extensión del fichero a _____ .ASM
-El ensamblado se realiza mediante un programa ensamblador en este caso el MPASM.EXE
-El fichero ensamblado tiene la extensión _____ .EXE, en el caso de contener errores se debe editar nuevamente, el fichero _____ . ERR nos informara sobre los errores cometidos.
-Al no contener errores se puede realizar la depuración y simulación, o directamente proceder a la grabación.

-El volcado y grabación del PIC se realizara conectando la tarjeta grabadora mediante su conexión con el PC a través del port serie, USB, o en nuestro caso el port paralelo, y ejecutando el PICME_TR.EXE
-Finalmente se comprueba la correcta grabación y se realizan las pruebas de funcionamiento.

ESTRUCTURA DE UN PROGRAMA EN ENSAMBLADOR



Edición de un programa en ensamblador
Consta de los siguientes elementos o partes que deben ser codificadas:
- Comentario descriptivo del programa (opcional, pero recomendable)
- Definir el microcontrolador que se usará (con las directrices LIST e INCLUDE).
- Introducir las opciones de compilación (que serán vistas más adelante)(opcional)
- Establecer las constantes que se usarán (con la directriz EQU).
- Reservar espacios de memoria (directriz RES) (si es necesario)
- Configurar los puertos
- Desarrollar el programa

-Poner comentarios

Tipos de ficheros utilizados
Edición **.DOC** Es el listado del programa con editor en el cual se especifica las subrutinas .INC y directivas a utilizar
Fuente **.ASM** **.SRC** Cambio de extensión, de edición a fuente para ensamblar
Listado **.LST** Es el listado del programa completo incluyendo las distintas instrucciones incluidas en subrutinas y generadas por directivas
Objeto .OBJ
Ejecutable **.HEX** Es el fichero en HEX realmente ejecutable por el PIC
Referencias Cruzadas **.XRF**
Error .ERR
Depuración **.COD**
Inicialización **.INI**

Ejercicio:

;Leer el estado de los 5 interruptores del Port A(RA4-RA0) y reflejar ;el estado de los mismos sobre los leds RB4-RB0 conectados al Port B

```
        List    p=16F84         ;Tipo de procesador
        include "P16F84.INC"  ;Define los registros internos


        org     0x00            ;Vector de Reset
        goto    Inicio

        org     0x05            ;Salva el vector de interrupción


Inicio  clrf    PORTB           ;Borra los latch de salida
        bsf     STATUS,RP0      ;Selecciona banco 1
        clrf    TRISB           ;Puerta B se configura como salida
        movlw   b'00011111'
        movwf   TRISA           ;Puerta A se configura como entrada
        bcf     STATUS,RP0      ;Selecciona banco 0


Loop    movf    PORTA,W         ;Leer las entradas RA0-RA4
        movwf   PORTB           ;Reflejar en las salidas
        goto    Loop            ;Bucle sin fin

        end                     ;Fin de edición del programa fuente
```

Ejercicio:

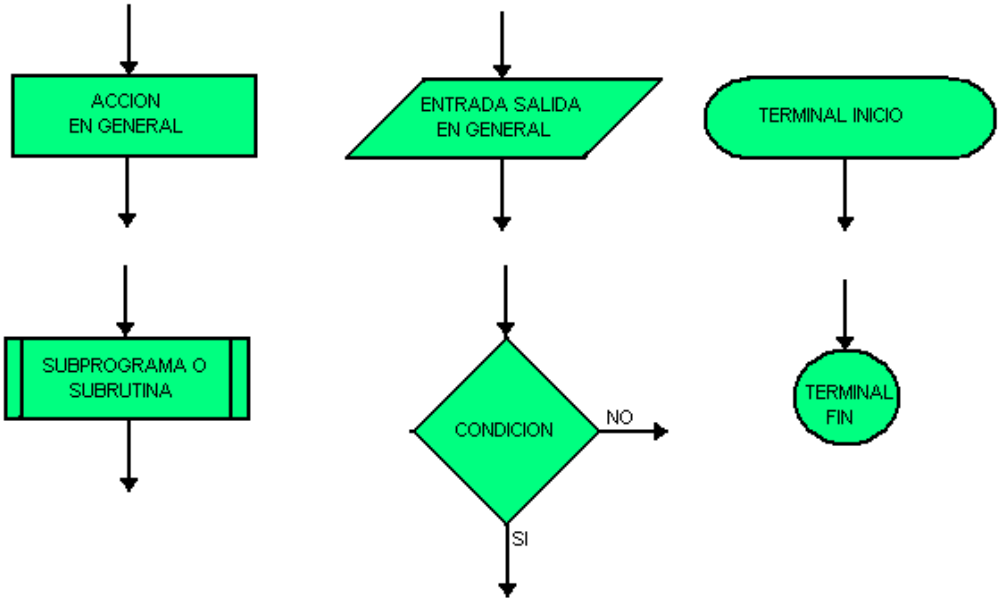
;Leer el estado de los 5 interruptores del Port A(RA4-RA0) y realiza la función and con 01010101 y refleja el resultado sobre los leds RB4-RB0 conectados al Port B

Ejercicio:

;Leer el estado de los 5 interruptores del Port A(RA4-RA0) y realiza la función or con con el valor 01010101 depositado en el registro “ parámetro” y refleja el resultado sobre los leds RB4-RB0 conectados al Port B

CAPITULO 5.
DIRECTIVAS PROGRAMACIÓN

Organigramas de programación



SIMBOLOGIA GENERAL DE LOS ORGANIGRAMAS

Programación Assembler de los PIC.
El **ensamblador** convierte los **mnemónicos de las instrucciones** del programa a su correspondiente código binario para ser decodificadas y ejecutadas por el microprocesador..
Un **programa fuente** en ensamblador contiene dos tipos de sentencias: las instrucciones y las directivas.
Las **directivas** son comandos insertados que controlan de forma mas automática el proceso de ensamblado, indicando al ensamblador como tratar las instrucciones y los datos,
Las instrucciones se utilizan para la realización del programa, y las directivas sólo son utilizadas para el ensamblado.

Etiquetas (Label)
Es cada uno de los puntos de salto del programa a los que se puede acceder a través de la instrucción adecuada, se sitúan en el primer campo de la instrucción.
En el caso de las subrutinas, se inicia en la instrucción de la etiqueta y finaliza con la instrucción de retorno (RETURN o RETLW), a la siguiente instrucción desde donde se realizo la llamada.

Subrutina.
Está formada por un conjunto de instrucciones que esta escrita una sola vez y se puede acceder a ella desde distintos puntos del programa mediante la instrucción de llamada CALL.
La MACRO se diferencia de la subrutina en que se repite tantas veces como se utilice la macro dentro del programa principal, dan mayor rapidez de ejecución pero utilizan mas memoria.

Macros.
Mediante un nombre simbólico se puede representar a la agrupación de un conjunto de

instrucciones que aparecen repetidas veces en el transcurso del programa. Mediante este procedimiento se puede simplificar la edición del programa principal.

Principales DIRECTIVAS utilizadas en el proceso de ensamblado

Estos comandos se usan para simplificar la tarea de programar, automatizando el proceso indicando al ensamblador como tratar las instrucciones y los datos del programa.

Campos de una directiva.

Los campos de las directivas son similares a las de sentencia de instrucción:
[nombre] nombre_directiva [operandos] [comentario]
Como en las sentencias de instrucción sólo es necesario el campo de nombre_directiva.

Directivas de definición de símbolos

EQU (EQUivalence): Asigna un valor a una expresión de nombre simbólico.

```
Ej.
contador_1   EQU     0x22          ;  Asignar 22Hexadecimal a la palabra Contador_1
temp         EQU          12
DATO         EQU          22
Bank_1       EQU          BSF STATUS,RP0
clockrate    EQU          .4000000       ; frecuencia del cristal
fclk         EQU          clockrate/4    ; frecuencia del reloj interno
W            EQU  0
F            EQU          1
```

Pueden formar parte y estar **contenidas en el fichero de Instrucción INCLUDE** que se incluye al principio del programa.

Directivas de control del ensamblador.

ORG (ORiGin): Indica la posicion de memoria de inicio de ensamblado, a partir de la posición de memoria en que se situarán las otras instrucciones

```
Ej:
ORG    0x00   ;Inicia el programa en la posición cero
GOTO  INICIO  ;Salta el vector de interrupción situado en posc 4, y continua en posc 5
ORG    0x05
INICIO  (Primera instrucción del programa); Posición 5 para el Inicio del progm-instrucc
```

END: Indica el final del fichero fuente debe situarse al final, para indicar al ensamblador que el programa ha finalizado. Esta siempre debe estar presente, aunque el flujo del programa acabe en un bucle.

TD: Equivalente y simplifica la sucesión de una serie de instrucciones **retlw**

RADIX: Indica que la base de numeración asignada por defecto es Hexadecimal.

Directivas de definición de segmentos y procedimientos.

INCLUDE " nombre_fichero.INC": Define los registros internos y las etiquetas de manejo del programa. También indica qué archivos se deben tener en consideración al compilar el código. Debe colocarse al principio, y tiene la siguiente sintaxis:

```
EJEMPLO
 include          "P16F84.INC"
; P16F84A.INC  Standard Header File, Version 2.00    Microchip Technology, Inc.
```

```
        NOLIST
; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F84 microcontroller.  These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;               C:\ MPASM MYFILE.ASM /PIC16F84A
;       2. LIST directive in the source file
;               LIST   P=PIC16F84A
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;       Revision History
;
;==========================================================================

;Rev:   Date:    Reason:

;1.00   2/15/99 Initial Release

;==========================================================================
;
;       Verify Processor
;
;==========================================================================

        IFNDEF __16F84A
          MESSG "Processor-header file mismatch.  Verify selected processor."
        ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                       EQU     H'0000'
F                       EQU     H'0001'

;----- Register Files------------------------------------------------------

INDF                    EQU     H'0000'
TMR0                    EQU     H'0001'
PCL                     EQU     H'0002'
STATUS                  EQU     H'0003'
FSR                     EQU     H'0004'
PORTA                   EQU     H'0005'
PORTB                   EQU     H'0006'
EEDATA                  EQU     H'0008'
EEADR                   EQU     H'0009'
PCLATH                  EQU     H'000A'
INTCON                  EQU     H'000B'

OPTION_REG              EQU     H'0081'
TRISA                   EQU     H'0085'
TRISB                   EQU     H'0086'
```

```
EECON1                  EQU     H'0088'
EECON2                  EQU     H'0089'

;----- STATUS Bits --------------------------------------------------------

IRP                     EQU     H'0007'
RP1                     EQU     H'0006'
RP0                     EQU     H'0005'
NOT_TO                  EQU     H'0004'
NOT_PD                  EQU     H'0003'
Z                       EQU     H'0002'
DC                      EQU     H'0001'
C                       EQU     H'0000'

;----- INTCON Bits --------------------------------------------------------

GIE                     EQU     H'0007'
EEIE                    EQU     H'0006'
T0IE                    EQU     H'0005'
INTE                    EQU     H'0004'
RBIE                    EQU     H'0003'
T0IF                    EQU     H'0002'
INTF                    EQU     H'0001'
RBIF                    EQU     H'0000'

;----- OPTION_REG Bits ----------------------------------------------------

NOT_RBPU                EQU     H'0007'
INTEDG                  EQU     H'0006'
T0CS                    EQU     H'0005'
T0SE                    EQU     H'0004'
PSA                     EQU     H'0003'
PS2                     EQU     H'0002'
PS1                     EQU     H'0001'
PS0                     EQU     H'0000'

;----- EECON1 Bits --------------------------------------------------------

EEIF                    EQU     H'0004'
WRERR                   EQU     H'0003'
WREN                    EQU     H'0002'
WR                      EQU     H'0001'
RD                      EQU     H'0000'

;=========================================================================
;
;       RAM Definition
;
;=========================================================================

        __MAXRAM H'CF'
        __BADRAM H'07', H'50'-H'7F', H'87'

;=========================================================================
;
;       Configuration Bits
;
;=========================================================================

_CP_ON                  EQU     H'000F'
```

```
_CP_OFF                 EQU     H'3FFF'
_PWRTE_ON                 EQU     H'3FF7'
_PWRTE_OFF                EQU     H'3FFF'
_WDT_ON                 EQU     H'3FFF'
_WDT_OFF                EQU     H'3FFB'
_LP_OSC                 EQU     H'3FFC'
_XT_OSC                 EQU     H'3FFD'
_HS_OSC                 EQU     H'3FFE'
_RC_OSC                 EQU     H'3FFF'
```

También se pueden crear archivos propios con funciones, definiciones y subrutinas de uso frecuente, evitando tenerlas que copiar de forma repetitiva.
Deben estar incluidas en archivos del tipo _____ . INC

LIST p =tipo_procesador
Indica el tipo de microcontrolador o PIC a utilizar, para que el compilador reconozca el PIC que tiene que programar. Se debe colocar a continuación del "include".
Ej. p=16f84

 # DEFINE asigna un nombre simbólico a una instrucción, o cadena en sustitución de un texto, como si se tratara de crear pequeñas macros

Ej.
```
#DEFINE   LED  PORT B,4           ; a LED le asigna el Bit 4 del Port B
#DEFINE   LEER    BSF  PORTA,1     ; A LEER  le asocia la instrucción  BSF  PORTA,1
#define   CERO    STATUS,2        ; Mediante la palabra CERO, accede al bit 2 de Status.
#define    BANCO1BSF   OPTION,RP0   ;Acceso a Banco 1
#define    BANCO0BCF OPTION,RP0   ;Acceso a Banco 0
```

Para DEFINE es imprescindible utilizar del carácter almohadilla(#), mientras que para INCLUDE, se puede prescindir de ella.

TITLE
Su sintaxis es TITTLE "Nombre del código" El compilador toma en consideración el título de su código, cuyo nombre hará constar en los archivos .lst.

MACRO
La macro se inicia con la declaración de MACRO, y termina con ENDM.
Ej:
```
MACRO           JumpLabel       ;Salta aquí
btfsc     STATUS, Z       ;Pregunta si el
goto      JumpLabel       ;Flag está a uno
ENDM                ;ahora lo invocamos
movf      AnyReg, w       ;Línea de código
xorwf     Constante       ;Compara la variable con Constante
jz        ThisLabel       ;Salta a la macro
```

ENDM.
Define el final de la MACRO

CBLOCK
Define el inicio de dirección para un bloque de constantes

ENDC
Define el final de ENDC

Ejemplo

```
LCDVAR                  EQU       0x0C ; Dirección física memoria datos
```

```
CBLOCK      LCDVAR              ; Define dirección inicio bloque de las variables
            LCD_TEMP_1  ; Primera variable
            LCD-TEMP-2  ; Segunda variable
ENDC
```

RESUMEN DIRECTIVAS

Directiva	Descripción	Sintaxis
__BADRAM	Especifica las localizaciones inválidas de la RAM.	__badram <expr>
BANKISEL	Genera código para la selección del banco de RAM para un direccionamiento indirecto.	bankisel <etiqueta>
BANKSEL	Genera código para la selección de banco RAM.	banksel <etiqueta>
CBLOCK	Define un bloque de constantes.	cblock [<expr>]
CODE	Comienzo de sección de código ejecutable.	[<etiqueta>] code [<dirección>]
CONFIG	Especifica los bits de configuración.	config <expr>
CONSTANT	Declara símbolos constantes.	constant <etiqueta>[=<expr>,... ...,<etiqueta>[=<expr>]
DATA	Crea datos numéricos y textos.	[<etiqueta>]data<expr>[,<expr>,., expr>] [<etiqueta>]data“<cadena_de_texto>” [,“<cadena_de_texto>”, ...]
DB	Declara datos de un byte.	[<etiqueta>] db<expr>[,<expr>,...,<expr>] [<etiqueta>] db“<texto>”[,“<texto>”,...]
DE	Define datos EEPROM.	[<etiqueta>] de<expr>[,<expr>,...,<expr>] [<etiqueta>] de“<texto>”[,“<texto>”,...]
#DEFINE	Define etiquetas.	define <nombre> [<valor>] define <nombre> [<arg>,...,<arg>]<valor>
DT	Define una tabla.	[<etiqueta>] dt<expr>[,<expr>,...,<expr>] [<etiqueta>] dt“<texto>”[,“<texto>”,...]
DW	Declara datos de una palabra.	[<etiqueta>]dw<expr>[,<expr>,...,<expr>] [<etiqueta>]dw“<texto>”[,“<texto>”,...]
ELSE	Comienzo de bloque alternativo a IF.	else
END	Fin del bloque de programa.	end
ENDC	Fin de un bloque de definición de constantes.	endc
ENDIF	Fin de un bloque condicional.	endif
ENDM	Fin de la definición de una macro.	endm
ENDW	Fin de un lazo while	endw
EQU	Define y ensambla constantes.	<etiqueta> equ <expr>
ERROR	Emisión de un mensaje de error.	error “<cadena_de_texto>”
ERRORLEVEL	Fija niveles de error.	errorlevel 0 | 1 | 2 <+ | -><nummsj>
EXITM	Sale de una macro.	exitm
EXPAND	Expansión de macros.	expand
EXTERN	Declara una etiqueta externa.	extern <etiqueta>[,<etiqueta>]
FILL	Ocupa memoria.	[<etiqueta>] fill <expr>, <cuenta>
GLOBAL	Exporta una etiqueta definida.	global <etiqueta> [, <etiqueta>]
IDATA	Comienza una sección de datos inicializados.	[<nombre>] idata [<dirección>]
IDLOCS	Especifica localizaciones ID.	idlocs <expr>
IF	Comienza un bloque de ensamblado de código condicional.	if <expr>
IFDEF	Ejecuta si el símbolo ha sido definido.	ifdef <etiqueta>
IFNDEF	Ejecuta si el símbolo no ha sido definido.	ifndef <etiqueta>
#INCLUDE	Incluye archivos fuente adicionales.	include <<archivo_include>> | “<archivo_include>”
LIST	Listado de opciones.	list [<opción_list>, ..., <opción_list>]
LOCAL	Declara una variable local de una macro.	local <etiqueta> [,<etiqueta>]
MACRO	Define una macro.	<etiqueta> macro [<arg>, ..., <arg>]
MAXRAM	Especifica la dirección máxima de RAM.	maxram <expr>
MESSG	Crea mensaje definido por el usuario	messg “<texto_del_mensaje>”
NOEXPAND	Fin de la expansión de macros.	noexpand
NOLIST	Desactiva las opciones de salida.	nolist
ORG	Fija el origen del programa.	<etiqueta> org <expr>
PAGE	Inserta una salida de página.	page
PAGESEL	Genera el código para la selección de la página de ROM.	pagesel <etiqueta>

PROCESSOR	Indica tipo de procesador.	processor <tipo_de_procesador>
RADIX	Especifica la base de numeración por defecto.	radix <radix_defecto>
RES	Reserva memoria.	[etiqueta>] res <unidades_de_memoria>
SET	Define y ensambla variables.	<etiqueta> set <expr>
SPACE	Inserta líneas en blanco en el listado.	space <expr>
SUBTITLE	Especifica el subtítulo del programa.	subtitle "<texto_de_subtitulo>"
TITLE	Especifica el título del programa.	title "<texto_de_titulo>"
UDATA	Comienza una sección de datos no inicializados.	[<nombre>] udata [<dirección>]
UDATA_OVR	Comienza una sección de datos no inicializados reutilizables.	[<nombre>] udata_ovr [<dirección>]
UDATA_SHR	Comienza una sección de datos no inicializados compartidos.	[<nombre>] udata_shr [<dirección>]
#UNDEFINE	Borra una etiqueta de substitución.	#undefine <etiqueta>
VARIABLE	Declara un símbolo variable.	variable <etiqueta>[=<expr>, ..., <etiqueta> [=<expr>]]
WHILE	Ejecuta un lazo mientras la condición sea verdad	while <expr>

.

6-Test-bit, salto y contadores

BITS DE TEST CON CONDICION DE SALTO

BTFSC

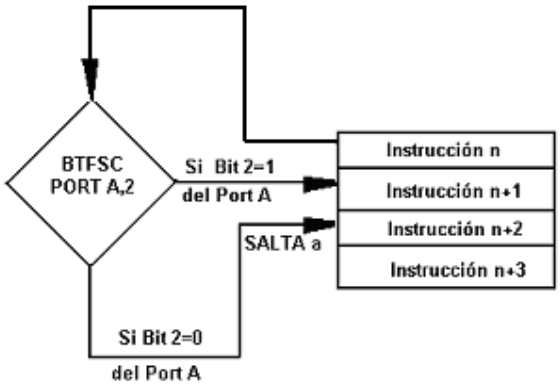
Acción	Comprueba un bit b del registro f, y salta la instrucción siguiente si este es cero
Sintaxis	**BTFSC f,b**
Funcionamiento	Bit Test, Skip if Clear
Hexadecimal	1b ff
Bits (OPCODE)	01 10bb bfff ffff
Operación	F(b) = 0? SI, salta una instrucción
Descripción	Esta instrucción comprueba el valor del bit b en el registro f, y si b = 0 entonces se salta la siguiente instrucción. Si b = 1 no salta y sigue con su ejecución normal.

Ejemplo

BTFSC PORTA, 2

Si en PORTA tenemos 11111011, Bit 2 = 0 el programa salta a la instrucción n+2.

Si en PORTA tenemos 00000100, Bit 2 = 1 el programa no salta, sigue con la instrucción n+1.



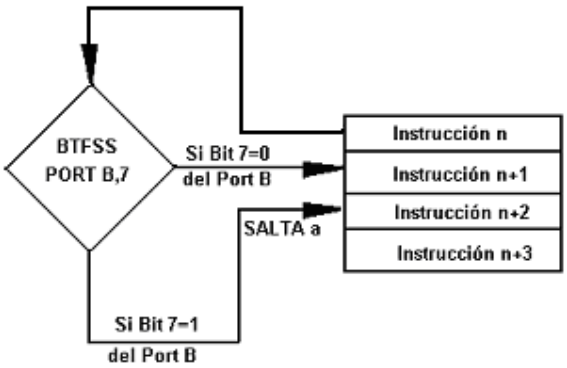
BTFSS

Acción	Comprueba un bit b del registro f, y salta la instrucción siguiente si este es uno
Sintaxis	**BTFSC f,b**
Funcionamiento	Bit Test, Skip if Set
Hexadecimal	1b ff
Bits (OPCODE)	01 11bb bfff ffff
Operación	F(b) = 1? SI, salta una instrucción
Descripción	Esta instrucción comprueba el valor del bit b en el registro f, y si b = 1 entonces se salta la siguiente instrucción. Si b = 0 no salta y sigue con su ejecución normal.
Comentarios	Ninguno
Registro STATUS	No modifica ningún bit de estado

Ejemplo

```
BTFSS    PORTB,   7
```

Si en PORTB tenemos como valor inicial 10000000, Bit 7=1 el programa salta a la instrucción n+2.
Si en PORTB tenemos como valor inicial 00000000, Bit 7=0 el program no salta y sigue con la instrucción n+1



Ejercicio:

; RB0 refleja ;el estado de RA0, y RB1 el complemento de RA0

```
                List    p=16F84                 ;Tipo de procesador
                  Include  "P16F84.INC"         ;Def. de registros internos

                org     0x00            ;Vector de Reset
                goto    Inicio
                org     0x05            ;Salva el vector de interrupción

Inicio  clrf    PORTB                   ;Pone a 0 los latch del Port B.
                bsf     STATUS,RP0      ;Selecciona banco 1 para configuración
                clrf    TRISB           ;Con 0 se configura Port B como salida
                movlw   b'00011111'
                movwf   TRISA           ;Con 1 se configura Port A como entrada
                bcf     STATUS,RP0      ;Selecciona banco 0

Loop            clrwdt                  ;Refresca el WDT timer
                btfsc   PORTA,0                 ;Chequea Bit 0, del Port A
                goto    RA0_es_1            ;Si RA 0 es igual a 1, salta a etiqueta RA0_es_1
```

```
                bcf     PORTB,0         ;Si RA 0 igual a 0, pone la  línea 0 del Port B a nivel
bajo 0
                bsf     PORTB,1             ;Si RA 0 igual a 0, pone la linea 1 del Port B a nivel
alto 1
                goto    Loop                ;Vuelve a inicio de programa

RA0_es_1        bsf     PORTB,0             ; SiRA 0 igual a 1, pone la línea 0 del Port B a nivel
alto 1
                bcf     PORTB,1             ; Si  RA 0 igual a 1, pone la línea 1 del Port B a
nivel bajo 0
                goto    Loop            ;Vuelve a inicio de programa

                end                     ;Vuelve al bucle principal.
```

Ejercicio:

;Según el estado de los interruptores RA0 y RA1, activar los leds RB0 a RB7 conectados a la puerta B, de acuerdo a la ;tabla de verdades.

;	RA1	RA0	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
;	--	--	--	--	--	--	--	--	--	--
;	0	0	1	0	1	0	1	0	1	0
;	0	1	0	1	0	1	0	1	0	1
;	1	0	0	0	0	0	1	1	1	1
;	1	1	1	1	1	1	0	0	0	0

```
                List    p=16F84                 ;Tipo de procesador
                Include "P16F84.INC"            ;Defin. de registros internos

                org     0x00                    ;Vector de Reset
                goto    Inicio
                org     0x05                    ;Salva el vector de interrupción

Inicio  clrf    PORTB                   ;Borra los latch de salida
                bsf     STATUS,RP0              ;Selecciona banco 1, para
                        configurar líneas de Entrada/Salida
                clrf    TRISB                   ;Puerta B se configura como salida
                movlw   b'00011111'             ;1 configuran líneas entrada
                movwf   TRISA                   ;Puerta A se configura como entrada
                bcf     STATUS,RP0              ;Selecciona banco 0

Loop:           clrwdt                          ;Refrescar el WDT
                btfsc   PORTA,0                         ;Chequea el estado de PORTA, 0
                goto    RA0_es_1                ;Si está a "1" salta a RA0_es 1
                btfsc   PORTA,1                         ;Chequea  estado de RA1
                goto    Estan_a_10              ;Si está a "1" salta a Estan_a_10
movlw   b'10101010'             ;Valores de salida para Estan a 00
movwf   PORTB                   ;Coloca 10101010 en Port B
                goto Loop                       ;Vuelve al inicio de programa

Estan_a_10      movlw   b'00001111'             ;Valores de salida para Estan a 10
                movwf   PORTB                   ;Salida de la secuencia 00001111
                goto    Loop                    ;Vuelve al inicio de programa

RA0_es_1        btfsc   PORTA,1                         ;Chequea el estado de PORTA,1
                goto    Estan_a_11              ;Si está a "1" salta a Estan_a_11
                movlw   b'01010101'             ;Valores de salida para Estan a 01
                movwf   PORTB                   ;Salida de 01010101 por el Port B
                goto    Loop                    ;Vuelve al inicio de programa

Estan_a_11      movlw   b'11110000'             ;Valores de salida para Estan a 11
                movwf   PORTB                   ;Salida de 11110000   por el Port B
```

```
goto     Loop                ;Vuelve al inicio de programa

end                          ;Fin del programa fuente
```

CONTADORES CON CONDICION DE SALTO

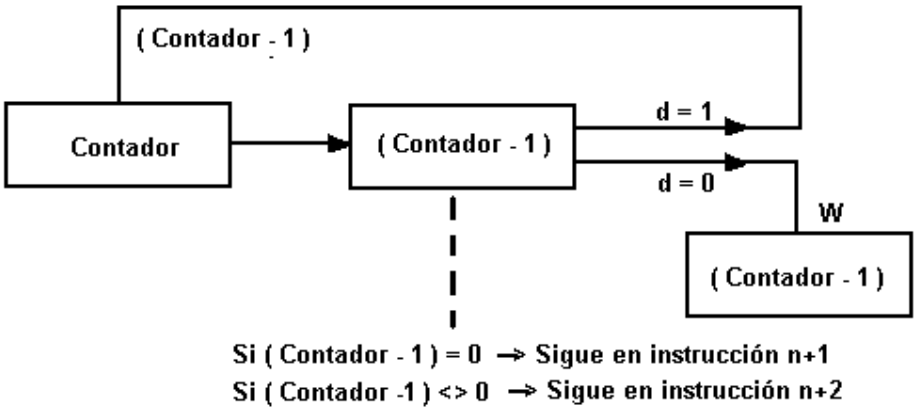
Z es el Zero Bit de Registro STATUS.
Cuando el contador alcanza un valor igual a 0 -> Z = 1
Si el resultado tiene un valor distinto de cero -> Z = 0

DECFSZ

Acción	Decrementa el registro f, y si el resultado es cero, se salta una instrucción.
Sintaxis	**DECFSZ f,d**
Funcionamiento	Decrement f, skip if 0
Hexadecimal	0B ff
Bits (OPCODE)	00 1011 dfff ffff
Operación	d = f – 1, si d = 0 SALTA (d puede ser W ó f).
Descripción	Esta instrucción decrementa el contenido del registro direccionado por el parámetro f, y si el resultado es 0 salta la instrucción siguiente. Si no, sigue con su curso habitual
Registro STATUS	No modifica ningún bit de estado

Ejemplo
DECFSZ contador, d
Decrementa en una unidad el valor del registro denominado contador contador = (contador -1)
Si d=1 el resultado se guarda en contador.
Si d=0 el resultado se guarda en W, contador no se altera

Si al decrementar en 1 el valor del contador queda igual a 0, salta a la INSTRUCCION n+2.
Si al decrementar en 1 el valor de contador queda distinto de cero , no salta sigue en la INSTRUCCION n+1



INCFSZ

Acción	Incrementa en 1 a f, y si f= 0 salta la siguiente instrucción
Sintaxis	**INCFSZ f,d**
Funcionamiento	Increment f, Skip if 0
Hexadecimal	0F ff
Bits (OPCODE)	00 1111 dfff ffff
Operación	d = f + 1, si d = 0 SALTA (d puede ser W ó f).
Descripción	Esta instrucción incrementa en una sola unidad el registro "f", en la cual si el resultado “d” es igual a cero, entonces salta la instrucción siguiente.
Comentarios	Ninguno

Registro STATUS	No modifica ningún bit de estado.

Ejemplo
INCFSZ contador , d
Se incrementa al valor del contador en una unidad. contador = (contador +1)
El resto sigue el mismo procedimiento que la instrucción DECFSZ

Ejercicio:

```
;Programa que muestra en el display el numero de interruptores
;que estan activados.

            list    p=16c84
            list    c=132
ra          equ     5                   ;direcci¢n del puerto A
rb          equ     6                   ;direcci¢n del puerto b
trisa       equ     5                   ;direcci¢n del registro
                                        ;de estado del puerto A
trisb       equ     6                   ;direcci¢n del registro
                                        ;de estado del puerto B
status      equ     3                   ;direcci¢n del registro de estado
contador    equ     0x0c                ;contador

            org     0                   ;comenzar en  vector de reset
            goto    inicio
            org     4                   ;   "     "   " interrupciones
            goto    inicio              ;para mandarlos a inicio

            org     5                   ;seguido de vector interrupcio
inicio      bsf     status,5            ;activar página 1
            clrf    trisb               ;declara Port B como salida
            movlw   b'00011111'     ;
            movwf   trisa           ;declara PortA como entrada
            bcf     status,5            ;activar página 0
bucle       movlw   0x00
            movfw   contador
            btfsc   ra,4            ;testear ra4
            incf    contador,1      ;si es 1, incrementar contador
            btfsc   ra,3            ;testear ra3
            incf    contador,1   ;si es 1, incrementar contador
            btfsc   ra,2            ;...
            incf    contador,1   ;si es 1, incrementar contador
            btfsc   ra,1
            incf    contador,1    ;si es 1, incrementar contador
            btfsc   ra,0
            incf    contador,1     ;si es 1, incrementar contador
test5       decfsz  contador,1
            goto    test4           ;si el resultado de decfsz no
                                    ; es cero testea  contador 4
            goto    num1            ;si el resultado de la decfsz
                                    ;es cero salta a num5
test4       decfsz  contador,1
            goto    test3
            goto    num2

test3       decfsz  contador,1
            goto    test2
            goto    num3

test2       decfsz  contador,1
            goto    test1
```

```
            goto    num4

test1       decfsz  contador,1
            goto    num0
            goto    num5

num5        movlw   b'01101101'     ;cinco en 7segmento carga en W
            movwf   rb
            goto    bucle

num4        movlw   b'01100110'     ;4 en 7seg. a W
            movwf   rb
            goto    bucle

num3        movlw   b'01001111'     ;3 en 7seg. a W
            movwf   rb
            goto    bucle

num2        movlw   b'01011011'     ;2 en 7seg. a W
            movwf   rb
            goto    bucle

num1        movlw   b'00000110'     ;1 en 7seg. a W
            movwf   rb
            goto    bucle

num0        movlw   b'00111111'     ;0 en 7seg al W
            movwf   rb
            goto    bucle

            end
```

Ejemplo:
;Programa que muestra en el display el numero de interruptores
;que estan activados.

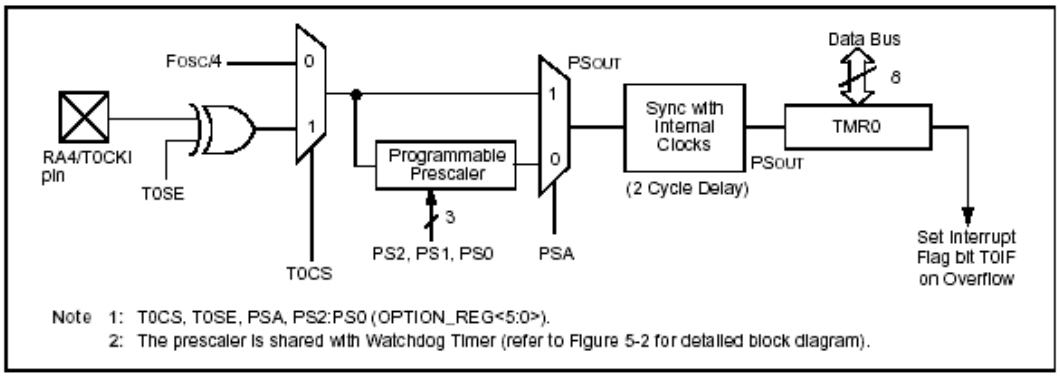
```
            list    p=16c84
            list    c=132
ra          equ     5               ;direcci¢n del puerto A
rb          equ     6               ;direcci¢n del puerto b
trisa       equ     5               ;direcci¢n del registro
                                    ;de estado del puerto A
trisb       equ     6               ;direcci¢n del registro
                                    ;de estado del puerto B
status      equ     3               ;direcci¢n del registro de estado
contador    equ     0x0c            ;contador

            org     0               ;comenzar en el vector de reset
            goto    inicio
            org     4               ;   "     "   "    "    " interrupciones
            goto    inicio          ;para mandarlos a inicio

            org     5               ;seguido del vector de interrupciones
inicio      bsf     status,5        ;activar p gina 1
            clrf    trisb           ;declarar puerto b como todo de salida
            movlw   b'00011111'     ;
            movwf   trisa           ;declarar el puerto A (RA0-RA4) de entrada
            bcf     status,5        ;activar pagina 0
```

```
bucle     movlw   0x00
          movfw   contador
          btfsc   ra,4            ;testear ra4
          incf    contador,1      ;si es 1, incrementar contador
          btfsc   ra,3            ;testear ra3
          incf    contador,1      ;si es 1, incrementar contador
          btfsc   ra,2
          incf    contador,1      ;si es 1, incrementar contador
          btfsc   ra,1
          incf    contador,1      ;si es 1, incrementar contador
          btfsc   ra,0
          incf    contador,1      ;si es 1, incrementar contador
test5     decfsz  contador,1
          goto    test4          ;si el resultado de la decfsz
                                 ;no es cero testea a ver si contador=4
          goto    num1           ;si el resultado de la decfsz
                                 ;es cero salta a num5
test4     decfsz  contador,1
          goto    test3
          goto    num2

test3     decfsz  contador,1
          goto    test2
          goto    num3

test2     decfsz  contador,1
          goto    test1
          goto    num4

test1     decfsz  contador,1
          goto    num0
          goto    num5

num5       movlw   b'01101101'    ;cinco en 7 segmentos se carga en W
          movwf   rb
          goto    bucle

num4       movlw   b'01100110'    ;4 en 7seg. a W
          movwf   rb
          goto    bucle

num3       movlw   b'01001111'    ;3 en 7seg. a W
          movwf   rb
          goto    bucle

num2       movlw   b'01011011'    ;2 en 7seg. a W
          movwf   rb
          goto    bucle

num1       movlw   b'00000110'    ;1 en 7seg. a W
          movwf   rb
          goto    bucle

num0       movlw   b'00111111'    ;0 en 7seg al W
          movwf   rb
          goto    bucle

          end
```

7-TEMPORIZADORES

La temporización se realiza a partir de un oscilador, se puede seleccionar el propio oscilador con frecuencia de entrada Fosc/4, o las señales de reloj que provengan a través de la entrada RA4/TOCK1.

Diagrama de bloques del TIMER0

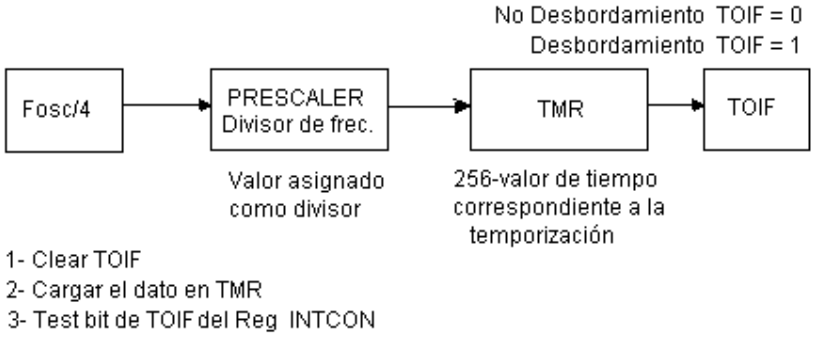


La señal de oscilación, es dividida en el PRESCALER, y posteriormente aplicada al TMR0 que actua como si fuera un contador de impulsos.
Cuando se alcanza el valor de desbordamiento de TMR0 se produce la señal de cambio de estado en el Bit TOIF, (Bit 2 del Registro INTCON) .

Cuando TOIF=1 Indica desbordamiento de TMR0
TOIF=0 Indica no desbordamiento de TMR0

El Clear de TOIF se produce por software

Su programación se realiza mediante el registro OPTION.
Basicamente se deben seleccionar:
Procedencia de señales de CK
Relación de división del Preescaler.
Selección de TMR0
Cargar el TMR con el complemento del valor correspondiente a la temporización



1- Clear TOIF
2- Cargar el dato en TMR
3- Test bit de TOIF del Reg INTCON

Calculo de la temporización
de la temporización = (256-Valor asignado a TMR0) (Valor preescaler) (Tiempo de ciclo Osc/4)
Supongamos cargando el TMR0 con 78, Preescaler 110 y Osc 4MHz se obtiene
Tiempo total = (256-78) (128) (1µseg) = 22,78 mseg

Registro OPTION.

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
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: 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 is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

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

Ejemplo
;El programa realiza temporización intermitente de 0.5 segundos, y de :forma complementaria sobre los led 0 y 1 del ;PortB.
;Se supone una frecuencia de trabajo del PIC de 4 MHz, por lo que el TMR0 evoluciona ;cada 1uS(4Tosc=1uS).
;Al TMR0 se le carga con 250 - 2 (su complemento, 7) y se selecciona ;un preescaler de 8. La temporización así ;obtenida es de unos 1990 uS. ;Si esta se repite 250 veces, se obtiene una temporización total en ;torno a los ;500000uS.

```
                List    p=16F84         ;Tipo de procesador
                Include "P16F84.INC"    ;Define registros internos

Contador        equ     0x10            ;Variable para el contador
                org     0x00            ;Vector de Reset
                goto    Inicio
                org     0x05            ;Salva el vector de interrupción

Inicio  bsf     STATUS,RP0              ;Selecciona banco 1 de datos
                movlw   b'11000010'
                movwf   OPTION_REG      ;Configura preescaler  8 asignado a TMR0
                clrf    TRISB           ;Configura portB como salida
                bcf     STATUS,RP0      ;Selecciona banco 0 de datos


Led0            bcf   PORTA,0
                bsf   PORTA,1
                call  tiempo
```

```
Led1            bcf   PORTA,1
                bsf   PORTA,0
                call  tiempo

Tiempo movlw  .250
                movwf  Contador       ;Inicia variable contador
Bucle1  clrf    INTCON                ;Desconecta flag de TMR0 e interrupciones
                movlw  .7
                movwf  TMR0           ;Carga el TMR0 con complemento de 250

Bucle2  btfss   INTCON,T0IF           ;Fin del TMR0 (flag T0IF=1) ??
                goto    Bucle2        ;No, esperar
                decfsz Contador,F     ;Si.Repetir las veces de contador
                goto    Bucle1
                return

                end                   ;Fin del programa fuente
```

Ejemplo

```
;Generación de ondas cuadradas de diferentes frecuencias variando el ;valor del TMR0
;La línea de salida RB0 cambiará de estado a una frecuencia ,determinada por el valor
introducido mediante los 3 interruptores, RA0-RA2.
;
;RA2    RA1     RA0     Frecuencia      Periodo         Semiperiodo
;---    ---     ---     ---------- -------             -----------
;0      0       0       0 KHz           ---             ---
;0      0       1       1 KHz           1000 uS                 500 uS
;0      1       0       2 KHz           500 uS          250 uS
;0      1       1       3 KHz           333 uS          166 uS
;1      0       0       4 KHz           250 uS          125 uS
;1      0       1       5 KHz           200 uS          100 uS
;1      1       0       6 KHz           166 uS          83 uS
;1      1       1       7 KHz           143 uS          71 uS
;
;En el tratamiento de interrupción que provocará el desbordamiento del TMR0, se puede
;apreciar como se salva el W y el registro de estado, para recuperarlos posteriormente.
;Es lo que se llama "salvar el contexto"
;

                List    p=16F84                 ;Tipo de procesador
                include "P16F84.INC"    ;Define de registros internos

Valor           equ     0x0c            ;Variable de frecuencia
W_Temp          equ     0x0d            ;W temporal
Status_Temp     equ     0x0e            ;Registro de estado temporal

                org     0x00            ;Vector de Reset
                goto    Inicio
                org     0x04            ;Vector de interrupción
                goto    Interrupcion

;*****************************************************************
;Tabla: esta rutina devuelve el valor a cargar en el TMR0 según la frecuencia selec-
;cionada. Partiendo de una frecuencia general de 4 MHz, el TMR0 evoluciona cada 1 uS.
;Se selecciona una preescaler de 4. El valor a cargar en TMR0 se obtiene de dividir el
;semiperiodo de la frecuencia deseada entre el preescaler. Al valor obtenido se le
;resta 2 por motivos de sincronismo interno del PIC, se convierte a hex. y se comple-
```

```
;menta.

Tabla:      addwf   PCL,F       ;Calcula desplazamiento de la tabla
            retlw   0x00        ;0 KHz
            retlw   0x86        ;1 KHz
            retlw   0xc5        ;2 KHz
            retlw   0xda        ;3 KHz
            retlw   0xe4        ;4 KHz
            retlw   0xea        ;5 KHz
            retlw   0xee        ;6 KHz
            retlw   0xf1        ;7 KHz


Interrupcion movwf  W_Temp              ;Salva el W
            swapf   STATUS,W
            movwf   Status_Temp ;Salva el registro de estado

            movf    Valor,W
            movwf   TMR0        ;Recarga el TMR0
            bcf     INTCON,T0IF ;Desactiva el flag TMR0
            movlw   b'00000001'
            xorwf   PORTB,F             ;Bascula RB0

            swapf   Status_Temp,W
            movwf   STATUS              ;Recupera el registro de estado
            swapf   W_Temp,F
            swapf   W_Temp,W    ;Recupera el registro W

            retfie

Inicio       clrf   PORTB       ;Borra los latch de salida
            bsf     STATUS,RP0  ;Selecciona banco 1
            clrf    TRISB       ;Puerta B se configura como salida
            movlw   b'00011111'
            movwf   TRISA       ;Puerta A se configura como entrada
            movlw   b'00000001'
            movwf   OPTION_REG  ;Preescaler de 4 para el TMR0
            bcf     STATUS,RP0  ;Selecciona banco 0
            movlw   b'00100000'
            movwf   INTCON      ;Interrupción TMR0 habilitada

Loop        clrwdt              ;Refrescar el WDT
            movf    PORTA,W
            andlw   b'00000111'
            btfss   STATUS,Z    ;RA0-RA2 = 0 ??
            goto    Salida_On   ;No, salida de frecuencia
            bcf     INTCON,GIE  ;Si, interrupciones OFF, frecuencia OFF
            goto    Loop

Salida_On   call    Tabla       ;Determina valor a cargar en TMR0
            movwf   Valor       ;Carga la variable
            bsf     INTCON,GIE  ;Interrupciones ON
            goto    Loop

            end                 ;Fin del programa fuente
```

Ejemplo
;Temporización para rotación secuencial
;Se desea realiza una rotación secuencial en el encendido de cada led contectados a la

```
;puerta B en la Trainer. Si RA0 = 0, la rotación seráde derecha a izquierda y viceversa.
;Cada led permanece encendido 0.25 segundos (250 mS)

                List    p=16F84         ;Tipo de procesador
                include "P16F84.INC"    ;Definiciones de registros internos

Contador        equ     0x0c            ;Variable para la temporización

                org     0x00            ;Vector de Reset
                goto    Inicio

                org     0x05            ;Salva el vector de interrupción

;*****************************************************************
;Delay es una rutina que realiza una temporización de 250 mS que es el tiempo en que han de
permanecer encendido cada uno de los leds.

;Si el PIC trabaja a una frecuencia de 4MHz, el TMR0 evoluciona cada uS. Si se desea
temporizar 25000 uS (25mS) con un preescaler de 128, el TMR0 deberá contar 195 eventos
(195 *128 = 24960). El valor 195 equivale a 0xc3 y, como el TMR0 es ascendente, habrá que
cargar su complemento (0x3c). Esta temporización habrá que repetirla 10 veces para
conseguirel total deseado (250000)

Delay           movlw   .10
                movwf   Contador        ;Carga el contador con 10
Delay_0         bcf     INTCON,T0IF     ;Desconecta el flag de rebosamiento del TMR0
                movlw   0x3c
                movwf   TMR0            ;carga el TMR0
Delay_1         clrwdt                  ;Refresco del WDT
                btfss   INTCON,T0IF     ;Rebasamiento del TMR0 ??
                goto    Delay_1         ;Todavía no han pasado los 25 mS
                decfsz  Contador,F      ;Decrementa contador. Se ha repetido 10 veces ?
                goto    Delay_0         ;Todavía no, temporiza 25 ms
                return                  ;Ahora si

Inicio          clrf    PORTB           ;Borra los latch de salida
                bsf     STATUS,RP0      ;Selecciona banco 1
                clrf    TRISB           ;Puerta B se configura como salida
                movlw   b'00011111'
                movwf   TRISA           ;Puerta A se configura como entrada
                movlw   b'00000110'
                movwf   OPTION_REG      ;Preescaler de 128 para el TMR0
                bcf     STATUS,RP0      ;Selecciona banco 0

                bsf     STATUS,C        ;Activa el carry
Loop            call    Delay           ;Temporiza 250mS
                btfsc   PORTA,0         ;Está a 0 RA0 ??
                goto    A_Dcha          ;No, rotación a derecha
A_Izda          rlf     PORTB,F         ;Si, rotación a izquierda
                goto    Loop
A_Dcha          rrf     PORTB,F         ;Rotación a derechas
                goto    Loop

                end                     ;Fin del programa fuente
```

11- INTERRUPCIONES

Las interrupciones son desviaciones asíncronas del flujo de control del programa originadas por diversos sucesos, que no están bajo el control de las instrucciones del programa.

Estos sucesos pueden ser internos o externos al sistema y en diseños industriales son un recurso muy importante para atender contecimientos físicos en tiempo real.
Cuando se produce una interrupción se detiene la ejecución del programa en curso, se salva la dirección actual en la pila y se carga el PC con la dirección del Vector de Interrupción , que es una dirección reservada de la memoria de código.
En el PIC16C84 el Vector de Interrupción está sitiado en la dirección 0004 H dela memoria de programa, donde empieza la Rutina de Servicio de la Interrupción. En general, en dicha dirección se suele colocar una instrucción de salto incondicional GOTO que salta directamente a la dirección del programa en la que comienza la rutina de servicio de la interrupción.

La rutina de servicio comienza guardando en la memoria de datos los registros específicos que va a emplear y por lo tanto pueden alterar su contenido. Antes del retorno al programa principal se recuperan los valores guardados y se restaura el estado del procesador.

Otra cuestión importante es averiguar cual de las posibles causas ha motivado la interrupción en este caso. Para ello se exploran los señalizadores de las fuentes de interrupción que están almacenados en el registro INTCON que se explicará más adelante.

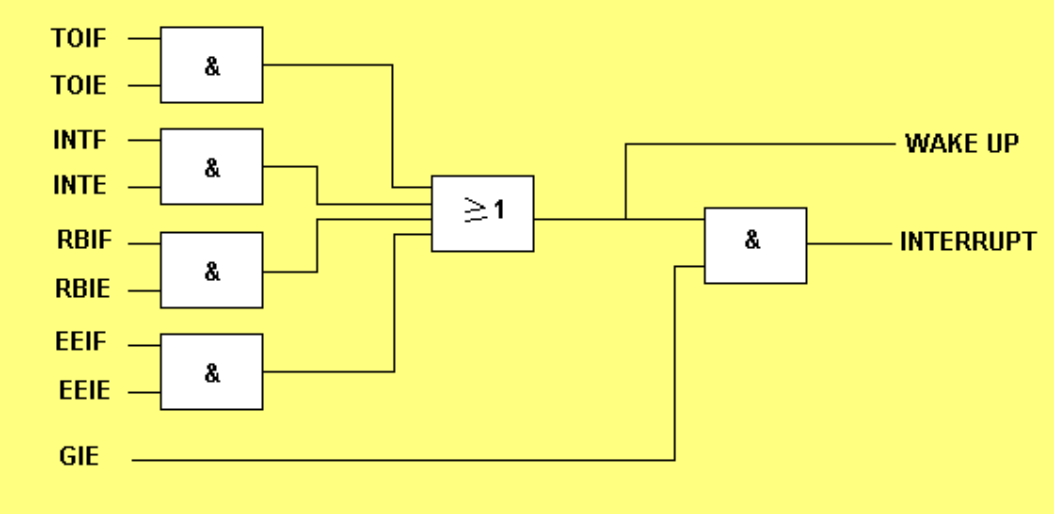
El PIC16C84 posee un bit GIE (*Global Interrupt Enable*) que cuando vale 0 prohibe todas las interrupciones. Lo primero que debe hacer la rutina de servicio es poner a 0 este bit para evitar que se aniden interrupciones.
En el retorno final de la interrupción (Instrucción RETFIE), GIE pasa a valer 1 automáticamente para permitir de nuevo las interrupciones. Sin embargo, los señalizadores de interrupción no se borran de forma automática y hay que desactivarlos por programa (de no hacerlo así en lasiguiente interrupción tendríamos problemas para esclarecer su origen).

CAUSAS DE LA INTERRUPCIÓN
El PIC16C84 tiene cuatro posibles causas de interrupción.
1. Activación del Pin RB0/INT.
2. Desbordamiento del temporizador TMR0.
3. Cambio de estado de una de las entradas RB4 - RB7 de la puerta B.
4. Finalización de la escritura en la EEPROM de datos.

Estas solicitudes se atienden siempre que el bit GIE tenga valor 1.
Cada una de las fuentes de interrupción anteriores dispone de un señalizador o *flag* que es un bit que se pone automáticamente a 1 cuando se produce la interrupción. Además cada interrupción tiene también un bit de permiso, que permite o prohibe la solicitud de esa interrupción.

REGISTRO DE CONTROL DE INTERRUPCIÓN INTCON

GIE: Permiso global de interrupciones. Si está activo (1) permite la solicitud de interrupciones cuyos bits de permiso individuales también lo permitan. Si está a 0 prohibe todas las interrupciones.



- **EEIE: Permiso de interrupción por fin de escritura en la EEPROM.** Si está a 0 prohibe esta interrupción y si está a 1 la permite.
TOIE: Permiso de interrupción por desbordamiento del TMR0. Si está a 0 prohibe esta interrupción y si está a 1 la permite.
- **INTE: Permiso de interrupción por activación de la patilla RB0/INT.** Si está a 0 prohibe esta interrupción y si está a 1 la permite.
- **RBIE: Permiso de interrupción por cambio de estado en RB4 - RB7.** Si está a 0 prohibe esta interrupción y si está a 1 la permite.
TOIF: Señalizador del desbordamiento del TMR0. Se pone automáticamente a 1 cuando el TMR0 pasa del valor FF H a 00 H.
INTF: Señalizador de activación de la patilla RB0/INT. Se pone a uno al activarse la patilla RB0/INT, indicando solicitud de interrupción externa.
RBIF: Señalizador de cambio de estado en las patillas RB4/RB7. Se pone a 1 cuando cambia el estado de alguna de estas líneas.

Organigrama de las operaciones de una interrupción en el PIC16C84.
Interrupción
PC PILA
GIE = 0
PC = 0004
(DIRECCIÓN VECTOR DE INTERRUPCIÓN)
RUTINA DE SERVICIO DE INTRRUPCIONES (RSI)SE ALMACENAN LOS REGISROS A MODIFICAR
DETERMINAR LA CAUSA DE LA INTERRUPCIÓN
SALTO A LA RUTINA DE SERVICIO CORRESPONDIENTE
SE RESTAURAN LOS VALORES DE REGISTROS ALMACENADOS
SE BORRA EL SEÑALIZADOR DE LA INTERRUPCIÓN
RETORNO (RETFIE)
PC PILA
GIE = 1

INTERRUPCIÓN EXTERNA INT
Esta fuente de interrupción es muy importante para atender acontecimientos externos en tiempo real. Cuando ocurre algún acontecimiento que el microcontrolador debe gestionar se activa la pata RB0/INT y se hace una solicitud de interrupción. De forma automática el bit INTF

= 1 y si el bit de permiso INTE = 1 y por supuesto el bit de permiso global GIE valen 1, se puede atender a la interrupción.

5.4. INTERRUPCIÓN DE DESBORDAMIENTO DEL CONTADOR TMR0

Se produce cuando el TMR0 se desborda y pasa del valor FF H a 00 H. En esta situación el señalizador OTIF se pone automáticamente a 1. Si además, el bit de permiso de esta interrupción TOIE y el bit de permiso global GIE, están puestos a 1 se atiende la interrupción. Si no se recarga el TMR0 sigue contado de 00 H a FF H. En cualquier momento se puede leer o escribir este registro, pero cada vez que se recarga se pierden dos ciclos de reloj. Cuando se recarga con el valor N (en decimal) cuenta un número de ciclos igual a 256 - N.

INTERRUPCIÓN POR CAMBIO DE ESTADO EN LAS LÍNEAS RB7:RB4 DE LA PUERTA B

Esta interrupción está expresamente diseñada para atender la pulsación de una tecla en un teclado matricial conectado dichas señales. En este caso, con cuatro líneas se puede controlar un teclado matricial de 16 teclas. Cada vez que cambia el estado lógico de alguna de estas entradas, el señalizador RBIF se pone a 1 y si los dos bits de permiso RBIE = GIE = 1 se autoriza la interrupción.

REINICIALIZACIÓN O RESET

El PIC16C84 tiene cinco causas por las que se provoca la reinicialización o reset del micro. Al producirse un reset se carga el valor 000 H (Vector de Reset) en el PC y los registros específicos toman un valor conocido. Las causas que motivan esta situación son las siguientes:

1. Conexión de la alimentación.
2. Activación de la para MCLR# en funcionamiento normal.
3. Activación de la pata MCLR# en estado de reposo.
4. Desbordamiento del perro guardián en funcionamiento normal.
5. Desbordamiento del perro guardián en el estado de reposo.

En la tabla 1 se presentan el estado lógico que adquieren los bits de los registros SFR de la memoria de datos cuando se provoca un reset por alguna de estas causas posibles.

APLICACIÓN PRÁCTICA: UN CONTADOR CONTROLADO POR INTERRUPCIÓN

Mediante un control por interrupción simulado mediante un pulsador que, al activarse, detendrá la cuenta, y la volverá a cero cuando de suelte el botón.

El problema aparecido es el hecho de que el siete segmentos implementado en esa ocasión controlaba mediante el bit 7 del puerto b el punto decimal, absolutamente innecesario para esta experiencia, mientras que ocupaba el bit 0 del mismo puerto para el *segmento a* del 7 segmentos, que es el único pin disponible para controlar directamente una interrupción externa. Se ha resuelto eliminando el punto decimal y desplazando un bit cada uno de los otros segmento, con lo que observará la tabla de los mismos cambiada.

```
; Programa interrupcion.asm
; Contamos hasta 0x5f.
; El valor del contador se visualizará en 8 diodos LED conectados al puerto B
; a partir de la patilla 1, sin gestión de punto decimal
; Preparado para PIC16F84
; Velocidad del reloj: 4 MHz
; Ciclo de instrucción: 1 MHz = 1 microsegundo
; Interrupciones: A través de PB.0, para detener y recomenzar la cuenta.
; Perro guardián: Desactivado
; Tipo de Reloj: XT
; Protección del código: Desactivado
; ************************************************************
;
LIST P = 16F84 ;Indicamos el modelo de PIC a utilizar
; Definición de registros
portb EQU 0x06 ;Hemos conectado el teclado al puerto B
;La dirección 0x06 corresponde al registro PORTB (puerto B)
; en el banco1
TRISB EQU 0X06 ; y TRISB en banco 1
```

```
estado EQU 0X03 ; La dirección del registro de estado es la 0x03
pc EQU 0x02 ; Contador de Programa, dirección de memoria actual de programa
intcon EQU 0x0B ; Registro gestionador de interrupciones
opcion EQU 0x01 ; Registro OPTION. Recordar que está en el banco 1.
; Definición de bits
banco EQU 0X05 ; Bit del registro de estado correspondiente al banco de datos
Z EQU 0X02 ; Bit indicador de que el registro W está a cero
int EQU 0x00 ; Bit de interrupción externa, es el 0 en el puerto B.
intdeg EQU 0x06 ; Bit 6 de OPTION, que indica si la interrupción PB0 es por nivel
alto.
intf EQU 0x01 ; Bit 1 de INTCON, flag de interrupción por PB0.
inte EQU 0x04 ; Bit 4 de INTCON, habilitador de interrupción por PB0.
GIE EQU 0x07 ; Bit 7 de INTCON, habilitador de interrupciones.
; Definición de constantes
w EQU 0 ; Destino de operación = w
f EQU 1 ; Destino de operación = registro
; Definición de variables
contador EQU 0X0C ; Contador
digito EQU 0X0D ; Para almacenar el dígito
; Comienzo del programa.
ORG 0X00 ; Cubrimos el vector de reset
GOTO inicio ; Saltamos a la primera dirección tras el vector de interrupción
ORG 0x04 ; Vector de interrupción
GOTO RSI
; ********** Inicialización de variables **********
ORG 0X05
inicio BSF estado,banco ; Cambiamos a la segunda página de memoria
CLRF TRISB ; Programa la puerta B como de todo salidas
BSF TRISB,int ; Salvo la pata de interrupción PB0, que es de entrada
BSF opcion,intdeg ; Interrupción PB0 cuando esté a nivel alto.
BCF estado,banco ; Volvemos a la página 0.
BCF intcon,intf ; Borramos el flag de interrupción por PB0.
BSF intcon,GIE ; Habilitamos las interrupciones.
BSF intcon,inte ; Habilitamos la interrupción por PB0.
CLRF portb ; Apaga el display, por si había residuos
CLRF contador ; Borra el contador (dirección 0x0C)
CLRW ; Borramos el registro W
; ********** Cuerpo Principal **********
Reset CLRF digito ; Comienza a contar por el 0
Siguien MOVF digito,w ; Coloca el siguiente dígito a evaluar en W
CALL Tabla ; Llama a la subrutina para coger el dato
; y hacer la conversión decimal-7 segmentos
MOVWF portb
Pausa DECFSZ contador ; Decrementa contador y repite
GOTO Pausa ; hasta que valga 0
INCF digito,f ; Incrementa el valor del dígito al siguiente
MOVF digito,w ; Pone el valor del dígito en W
XORLW 0x0A ; Chekea si el dígito sobrepasa el valor 9
BTFSC estado,Z ; Comprobando con un xor si W vale 0 (Z=1)
GOTO Reset ; Si Z=1 resetea el dígito y comienza de nuevo la cuenta
GOTO Siguien ; En caso contrario, continua la cuenta
; ********** La tabla queda definida aquí **********
Tabla ADDWF pc,f ; Suma al contador de programa el valor de offset, es decir,
; el valor del dígito. Así se genera un PC distinto
; según su valor,
; asegurando que vaya a la línea correcta de la tabla
RETLW 0x7F ; 0 en código 7 segmentos (desplazado a la izquierda)
RETLW 0x0C ; 1 en código 7 segmentos (desplazado a la izquierda)
RETLW 0xB6 ; 2 en código 7 segmentos (desplazado a la izquierda)
RETLW 0x9F ; 3 en código 7 segmentos (desplazado a la izquierda)
```

```
RETLW 0xCC ; 4 en código 7 segmentos (desplazado a la izquierda)
RETLW 0xDA ; 5 en código 7 segmentos (desplazado a la izquierda)
RETLW 0xFA ; 6 en código 7 segmentos (desplazado a la izquierda)
RETLW 0x0F ; 7 en código 7 segmentos (desplazado a la izquierda)
RETLW 0xFF ; 8 en código 7 segmentos (desplazado a la izquierda)
RETLW 0xDF ; 9 en código 7 segmentos (desplazado a la izquierda)
RSI BTFSS intcon,intf ; Si no es interrumpido por PB0, volver al programa
RETFIE
pulsado BTFSC portb,0 ; Retenemos hasta que se suelte el pulsador
GOTO pulsado
MOVLW 0xFF ; Puesto que se habrá de incrementar
MOVWF digito ; Ponemos el marcador a FF
BCF intcon,intf ; Borramos la bandera de interrupción
BSF intcon,inte ; Y rehabilitamos la interrupción por PB0
RETFIE
END
```

8-Watch Dog, Sleep , Wake Up

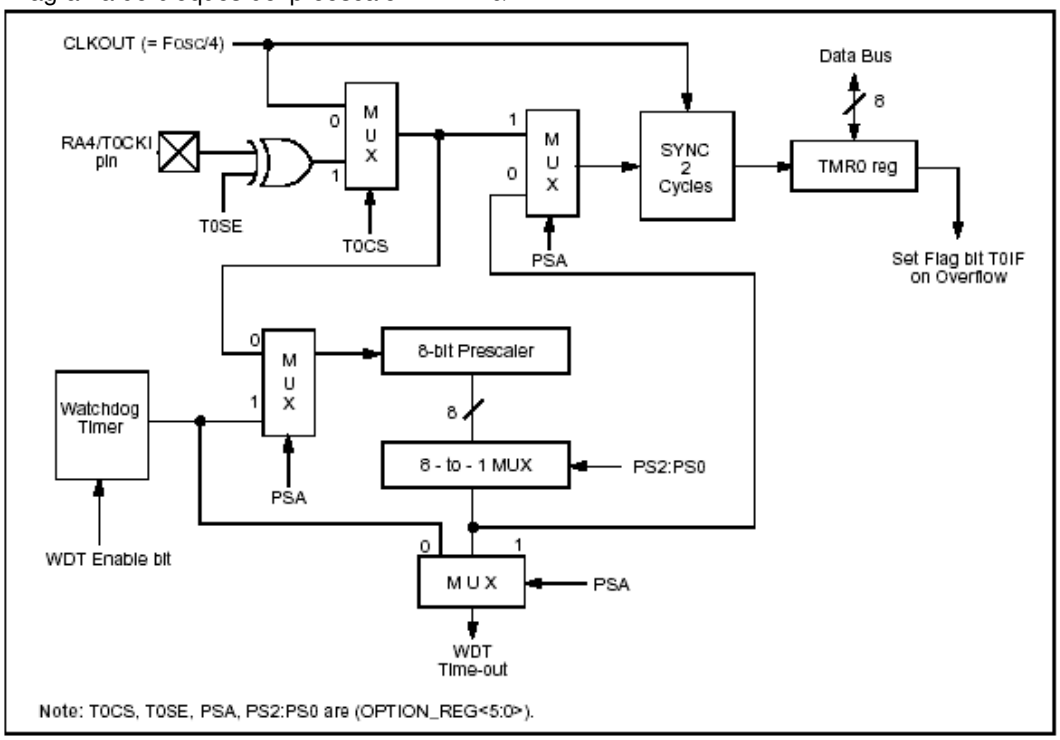
Función del Watch Dog (WDT)

Vigila el programa que no se cuelge. Se trata de un contador que al desbordarse a causa de fallo en la ejecución del programa, pueda realizar un Reset general y automáticamente reiniciar todo el sistema. Se puede ajustar su temporización mediante la programación del Prescaler, para las señales que recibe de su propio oscilador RC, con impulsos de 18 ms de tiempo periodico.

Cuando se desborda el Watch Dog se produce un Reset (Wake Up) del PIC, Para evitar este desbordamiento se debe realizar el refresco del Watch Dog cada cierto tiempo, colocando a 0 la temporización inicial, mediante las instrucciones CLRWDT.

Se deben situar estas instrucciones en puntos estratégicos del programa, con el fin de que no exista tiempo suficiente para el desbordamiento del WDT. En el caso de que se cuelgue el programa, entraría en un bucle que podríamos considerar infinito, pero en esta caso al no refrescarse el WDT se produce su desbordamiento y se reinicia el sistema.

Diagrama de bloques del preescaler TIMER0/WDT



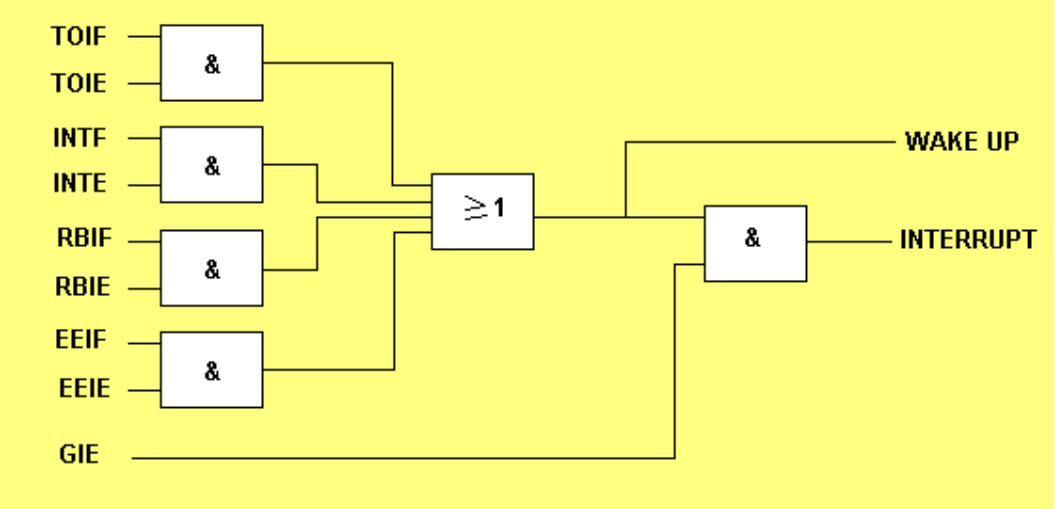
Programación del WDT
-Mediante el Bit PSA (PSA=1) del registro Option, se asigna el divisor de frecuencia al WDT
-Mediante la programación PS2, PS1, PS0, se realiza el ajuste del tiempo para el desbordamiento del WDT.

CLRWDT. Borra el valor de WDT reinicia su valor de **temporización a 0.**

WDTE. Desactivación del WDT poniendo a 0 el Bit 3 de OPTION, PSA no asigna el Prescaler a WDT

SLEEP y WAKE UP
Sleep. Modo de trabajo en bajo consumo.
Durante determinados momentos del programa como son esperas de acontecimientos externos, etc. el microcontrolador puede apagar gran parte de sus circuitos con el fin de ahorrar energía, pasando a esta situación mediante la instrucción de **Sleep.** Mediante una interrupción el microcontrolador reanuda su trabajo Wake Up.
En esta condición en el Registro STATUS, los Bits toman los valores, 1 -> TO y 0 -> PD.

Wake Up. **El microcontrolador despierta** y vuelve al funcionamiento normal **al ocurrir alguno de los eventos**:
-Pin MCLR activado a nivel bajo.
-Desbordamiento del Watchdog.
-Mediante una de las interrupcines.



Al despertarse el microcontrolador, si GIE esta desactivado se ejecuta la siguiente instrucción a la de sep. Cuando GIE = 1, se salta al vector de Interrupción que ocupa la dirección 004H.

Bits del registro STATUS utilizados por Watchdog y Sleep.
PD, Power Down, Flag de bajo consumo
/PD = 0 Cuando se ejecuta Sleep y entrar en reposo.
/PD = 1 Al conectar la alimentación Vdd, o ejecutar **clrwdt**
TO, Timer Out, Flag indicador de fin de temporizador
/TO = 0 Al desbordar Watchdog
/TO = 1 Al conectar la alimentación Vdd, o ejecutar **clrwdt o sleep.**

Ejemplo

;El modo "sleep" y el "wake-up" (despertar) mediante el watch-dog Timer (WDT)
;Este ejemplo pretende mostrar el empleo de la instrucción SLEEP para poner al PIC en el

;modo standby de bajo consumo. El despertar del mismo se producirá cada vez que el WDT ;rebase.
;En ese momento se producirá un incremento del valor de la puerta B que actuará como ;contador binario y nuevamente se volverá a la situaciónd de standby.
;El preescaler se asociará al WDT y estará comprendido entre 1 y 128, dependiendo del ;estado lógico de los interruptores RA0-RA2.
;El valor nominal del WDT es de 18mS. Es decir, con un preescaler de 1, el pic ;"despertará" cada 18mS, con un prrescaler de 128, lo hará cada 2,3 segundos.

```
                List    p=16F84             ;Tipo de procesador
                include "P16F84.INC"  ;Definiciones de registros internos

                org     0x00          ;Vector de Reset
                goto    Inicio
                org     0x05          ;Salva vector de interrupción

Inicio      clrf        PORTB         ;Borra los latch de salida
                bsf     STATUS,RP0    ;Selecciona banco 1
                clrf    TRISB         ;Puerta B se configura como salida
                movlw   b'00011111'
                movwf   TRISA         ;RA0-RA4 entradas
                movlw   b'00001000'
                movwf   OPTION_REG    ;Preescaler de 1 para el WDT
                bcf     STATUS,RP0    ;Selecciona banco 0


Loop            sleep                 ;Modo Standby

                incf    PORTB,F             ;Incrementa el contador binario sobre la puerta
B
                movf    PORTA,W
                andlw   b'00000111'   ;Lee el estado de los interruptores RA0-RA2
                iorlw   b'00001000'
                bsf     STATUS,RP0    ;Selecciona el banco 1
                movwf   OPTION_REG    ;Ajusta valor del preescaler
                bcf     STATUS,RP0    ;Selecciona el banco 1
                goto    Loop          ;Volver al modo Standby

                end                   ;Fin del programa fuente
```

Ejemplo
;Máquina de emvasado, con aplicación del WDT.
;Dos relés "M1" (RB0) y "M2" (RB1) gobiernan dos motores que arrastran dos cintas ;transportadoras. "M1" (RB0) transporta piezas y "M2" (RB1) embalajes. Un sensor "DP" ;(RA1) ;detecta el paso de piezas y, otro "DE" (RA2), detecta el correcto posicionamiento de un ;emvase. Al detectarse el paso de 10 piezas, el emvase se considera lleno, se activa una ;señal acústica "A" (RB2) y, la cinta que transporta embalajes, se desplaza hasta situar ;un nuevo emvase vacío. En este momento se desactiva la señal acústica "A"(RB2) y ;nuevamente avanza la cinta de piezas repitiéndose así el ciclo. Un interruptor "I" ;(RA0) activa o no a todo el sistema.

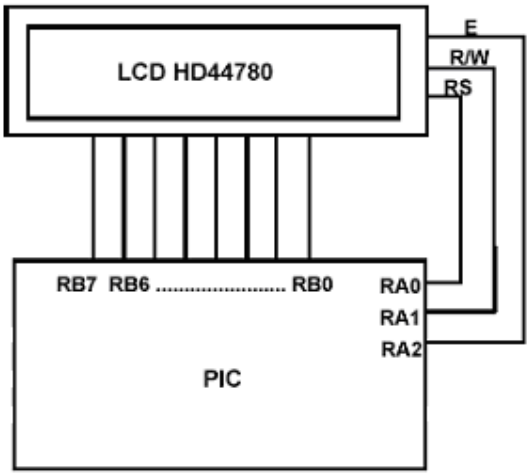
```
                List    p=16F84             ;Tipo de procesador
                include "P16F84.INC"  ;Definiciones de registros internos

Conta_pieza     equ     0x0c          ;Variable de Nº de piezas
```

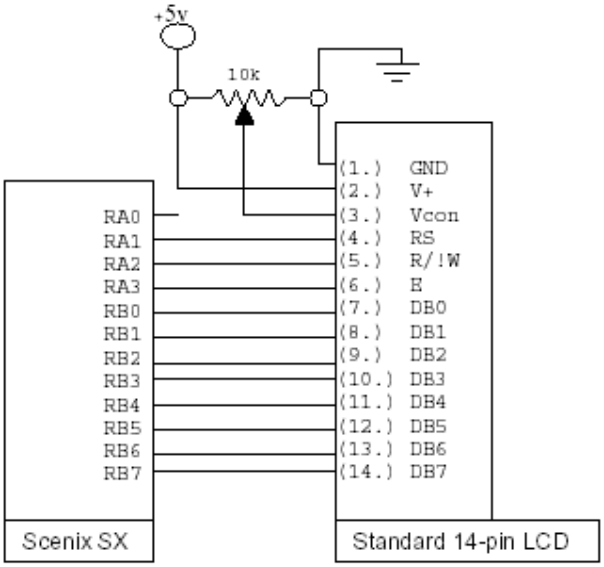
```
                org     0x00            ;Vector de Reset
                goto    Inicio

                org     0x05            ;Salva el vector de interrupción

;*****************************************************************
;
;Delay_20_ms es una rutina que realiza una temporización de 20 mS con objeto de eliminar
;el "efecto rebote" que poseen los elementos electromecánicos.
;Si el PIC trabaja a una frecuencia de 4MHz, el TMR0 evoluciona cada uS. Si se desea ;tempo-
rizar 20000 uS (20mS) con un preescaler de 128, el TMR0 deberá contar 156 eventos ;(156
*128 = 19968). El valor 156 equivale a 0x9c y, como el TMR0 es ascendente, habrá ;que cargar
su complemento (0x63)

Delay_20_ms:    bcf     INTCON,T0IF   ;Desconecta el flag de rebosamiento del TMR0
                movlw   0x63
                movwf   TMR0          ;carga el TMR0
Delay_20_ms_1           clrwdt  ;Refresco del WDT
                btfss   INTCON,T0IF   ;Rebasamiento del TMR0 ??
                goto    Delay_20_ms_1         ;Todavía no
                return                ;Ahora si

Inicio  clrf    PORTB           ;Borra los latch de salida
                bsf     STATUS,RP0    ;Selecciona banco 1
                clrf    TRISB         ;Puerta B se configura como salida
                movlw   b'00011111'
                movwf   TRISA         ;Puerta A se configura como entrada
                movlw   b'00000110'
                movwf   OPTION_REG    ;Preescaler de 128 para el TMR0
                bcf     STATUS,RP0    ;Selecciona banco 0

Loop_0 clrf     PORTB           ;Desconecta las salidas
Loop:           clrwdt                ;Refrescar el WDT
                movlw   d'10'
                movwf   Conta_pieza   ;Inicia variable con Nº de piezas
                btfss   PORTA,0               ;Chequea estado del interruptor I
                goto    Loop_0        ;Está en OFF, sistema parado

                call    Delay_20_ms   ;Elimina rebotes

                bcf     PORTB,0               ;Avance de piezas en OFF
                bsf     PORTB,1               ;Avance de emvases en ON

No_emvase       clrwdt                ;Refresca el WDT
                btfss   PORTA,2               ;Embase en posición ?
                goto    No_emvase         ;No.call      Delay_20_ms
;Elimina rebotes
                bsf     PORTB,0               ;Si. Avance de piezas en ON
                bcf     PORTB,1               ;Avance de emvases en OFF
                bcf     PORTB,2               ;Acústico en OFF

;Esta secuencia de instrucciones, espera que una pieza pase por el ;sensor "DP" en su
;totalidad

Pieza_0         clrwdt                ;Refresca el WDT
                btfss   PORTA,1               ;Llega la pieza ??
                goto    Pieza_0               ;Todavía no call Delay_20_ms
                                      ;Elimina rebotes

Pieza_1         clrwdt                ;Ahora si. Refresca el WDT
                btfsc   PORTA,1               ;Paso de pieza frente al sensor DP
```

```
                goto    Pieza_1             ;Todavía no     call    Delay_20_ms
                                            ;Elimina rebotes
                decfsz  Conta_pieza,F       ;Ahora si. Decr. Cont. piezas
                goto    Pieza_0             ;Esperar el paso de una nueva pieza
                bcf     PORTB,0             ;Han pasado 10,avance piezas en OFF
                bsf     PORTB,1             ;Avance de emvases en ON
                bsf     PORTB,2             ;Acústico en ON


Si_emvase       clrwdt                      ;Refresca el WDT
                btfsc   PORTA,2             ;Sigue el  emvase posicionado ??
                goto    Si_emvase           ;Todavía si
                call    Delay_20_ms
                goto    Loop                ;Ahora no
                end                         ;Fin del programa fuente
```

9-Visualización. Display LCD.

Una de las formas utlilizadas para visualizar los mensajes de microcontrolador es mediante las pantallas de LCD.
Entre los modelos utilizados estan los LM054,LM016l,LM020L,LM041L,etc, que a su vez utilizan internamente el microcontrolador HD44780.





Ejemplos de conexionado de un Microcontrolador al Display con HD44780

Función de cada uno de los Pins.
DB7-DB0 Lineas de transferencia. Pueden tranferrir los Comandos de Control de configuración de las distintas opciones de trabajo, o los Datos a visualizar en el Display, segun el estado de la linea RS.
RS Selección modo Control o Datos. RS=0 ->Modo control, RS=1 -> Modo Datos. Los comandos pueden enviarse en grupos de 4 0 8 Bits
Vecon Regula el contraste del Display
E Activación, E=1 Activado, E=0 Desactivado
R/W Lectura Escritura, R/W = 1 Lectura, R/W =0 Escritura.

PIN	NAME	OPERATION
1	Vss	(-) Ground
2	Vcc	(+) Power
3	Vee	Contrast Adjust. Connect to Potentiometer
4	RS	Data/!Instruction... 0 = Instruction input, 1 = Data input
5	R/!W	Read/!Write... 0 = Write, 1 = Read
6	E	Enable signal. Active High (Read). Negative edge triggers input latch (Write).
7	DB0	Data Bus Line 0 (LSB)
8	DB1	Data Bus Line 1
9	DB2	Data Bus Line 2
10	DB3	Data Bus Line 3
11	DB4	Data Bus Line 4
12	DB5	Data Bus Line 5
13	DB6	Data Bus Line 6
14	DB7	Data Bus Line 7 (MSB)

La pantalla del Display, esta controladas por su propio microcontrolador , y constan de una matriz de caracteres(Normalmente de 5x7 , o 5x10 puntos), distribuidos un una o varias lineas de 16 a 40 caracteres cada una.
Memoria DDRAM La información a visualizar en el display se guarda en la zona de memoria DDRAM(Data Display RAM)de 80 Bytes, 40 por cada una de las dos lineas, Fila 0 posiciones 00H-27H,Fila1 de 40H a 67H, pudiendo visualizar simultaneamente en pantalla 32 caracteres (16 por linea). Para la correcta visualización se deben tener en cuenta las posiciones de los caracteres y su correspondiente dirección en la DDRAM
Memoria CGROM (Character Generator ROM)almacena la tabla de caracteres que pueden ser visualizados, Para visualizar cada caracter se debe recibir por las lineas de datos su caracter correspondiente. Se pueden definir nuevos caracteres guardandolos en la CGRAM (Character Generator RAM)

Modo LECTURA. Tambien denominada Busy Flag, informa que el microcontrolador del LCD esta ocupado, y no puede aceptar nueva información hasta que el bit DB7 de Datos no pase a 0.
Estos modos se controlan mediante las señales RS, R/W y DB7

Comandos de control.
Permiten seleccionar las diferentes opciones de trabajo del display, mediante la orden enviada por las lineas DB7-DB0.
Cada uno de los Bits dentro de las palabra de control representan una opción especifica de trabajo

Instruction	Code RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{cp} or f_{osc} is 270 kHz)
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 µs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 µs
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 µs
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs

Table 6 Instructions (cont)

Instruction	Code RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{cp} or f_{osc} is 270 kHz)
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 × 10 dots, F = 0: 5 × 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable										DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses	Execution time changes when frequency changes Example: When f_{cp} or f_{osc} is 250 kHz, $37\ \mu s \times \frac{270}{250} = 40\ \mu s$

Note: — indicates no effect.

* After execution of the CGRAM/DDRAM data write or read instruction, the RAM address counter is incremented or decremented by 1. The RAM address counter is updated after the busy flag turns off. In Figure 10, t_{ADD} is the time elapsed after the busy flag turns off until the address counter is updated.

Direccionamiento Absoluto

Por defecto los caracteres se posicionan a partir del inicio de la DDRAM, añadiendo un offset de 00H a 27H, se pueden situar en cualquier posicion de la pantalla.

Direccionamiento Relativo

Manteniendo una posicion fija de cursor, a medida que los caracteres aparecen en pantalla, se pueden ir desplazando a derecha o izquierda.

Procedimiento para la programación del LCD

-Inicializacion: 8 o 4 Bits Datos, numero de lineas del LCD , caracteres por linea y matriz de puntos por caracter
-Programación de la presentación en el LCD mediante comandos
-Datos a visualizar

LCD__.INC Existen librerias de subrutinas especificas y en INCLUDE para facilitar la programación de las correspodientes pantallas de visualización.

Istruccion retlw
Su función es colocar una constante o literal en el Registro W. para posteriormente ser enviada al Display. Colocandolas una a continuación de otra se pueden formar secuencias de caracteres o datos.
Es similar al return de una subrutina, cuya unica unica función sea colocar una literal en W.
Ej.
rtlw 0x4d
rtlw 0x61
rtlw 0x10
rtlw 0x0f

Directiva DT
Simplifica el uso de varias instrucciones retlw de una tabla
El ejemplo anterior puede ser substituido por:
DT 0x4d, 0x61, 0x10, 0x0f

Ejemplo de manejo de la pantalla LCD HD44780

;Este ejemplo pretende introducirnos en el manejo de la pantalla LCD, para la ;visualización de diferentes mensajes (p.e. Hola).
;Debe recordarse que las líneas RA0-RA2 actúan ahora como salida de señales de control ;hacia el LCD. Al estar conectadas con sendos interruptores en la MicroPIC Trainer, ;estos deben estar a nivel lógico "1" permanentemente.

```
                List    p=16F84             ;Tipo de procesador
                include "P16F84.INC"  ;Definiciones de registros internos

Lcd_var         equ     0x0c            ;Variables (2) de las rutinas de manejo del LCD

                org     0x00            ;Vector de Reset
                goto    Inicio
                org     0x05            ;Salva vector de interrupción

                include "LCD_Cxx.inc"  ;Incluye las rutinas de manejo del LCD

Inicio      clrf        PORTB           ;Borra los latch de salida
                bsf     STATUS,RP0      ;Selecciona banco 1
                clrf    TRISB           ;Puerta B se configura como salida
                movlw   b'00011000'
                movwf   TRISA           ;RA0-RA2 salidas, RA3-RA4 entradas

                bcf     STATUS,RP0      ;Selecciona banco 0

                call    LCD_INI             ;Secuencia de inicio del LCD
                movlw   b'00001111'
```

```
                call    LCD_REG                 ;Envía instrucción: LCD ON, Cursor ON y blink
ON
                movlw   'H'
                call    LCD_DATO        ;Visualiza H
                movlw   'o'
                call    LCD_DATO        ;Visualiza o
                movlw   'l'
                call    LCD_DATO        ;Visualiza l
                movlw   'a'
                call    LCD_DATO        ;Visualiza a
                movlw   ' '
                call    LCD_DATO        ;Visualiza blanco


Loop            sleep                   ;Puesta en Standby
                goto    Loop            ;Vuelta a standby

                end                     ;Fin del programa fuente
```

LCD_CXX.INC RUTINAS DE MANEJO DEL LCD HD44780

```
;**********LCD_CXX.INC  RUTINAS DE MANEJO DEL LCD**************

;El conjunto de rutinas que se presentan a continuación permiten realizar las tareas
;de control del módulo de visualización LCD. Este fichero se debe incluir en los futuros
;programas fuente mediante la directiva INCLUDE
;
;Permiten adaptarse a los nuevos tiempos de las nuevas pantallas LCD de Winteck
;
;Las rutinas están adaptadas a las velocidades  de los nuevos módulo de winteck, en las
;cuales el tiempo de activación (Tc) de la señal ENABLE es de 40 uS.

#define ENABLE      bsf PORTA,2       ;Activa señal E
#define DISABLE     bcf PORTA,2       ;Desactiva señal E
#define LEER        bsf PORTA,1       ;Pone LCD en Modo RD
#define ESCRIBIR    bcf PORTA,1       ;Pone LCD en Modo WR
#define OFF_COMANDO  bcf PORTA,0      ;Desactiva RS (modo comando)
#define ON_COMANDO   bsf PORTA,0      ;Activa RS (modo dato

        CBLOCK          Lcd_var                 ;Inicio de las variables. Será la primera

                Lcd_Temp_1    ;direccón libre disponible
                Lcd_Temp_2
                ENDC


;*****************************************************************************
;UP_LCD: Configuración PIC para el LCD.
;
UP_LCD          bsf     STATUS,RP0      ;Banco 1
            clrf    PORTB           ;RB <0-7> salidas digitales
            movlw   b'00011000'
                movwf    PORTA          ;RA0-RA2 salidas, RA3-RA4 entradas
            bcf     STATUS,RP0      ;Banco 0
            OFF_COMANDO             ;RS=0
            DISABLE                 ;E=0
            return
;
;*****************************************************************************
```

```
;LCD_BUSY: Lectura del Flag Busy y la dirección.
;
LCD_BUSY       LEER                ;Pone el LCD en Modo RD
           bsf    STATUS,RP0
           movlw  H'FF'
           movwf  PORTB          ;Puerta B como entrada
           bcf    STATUS,RP0     ;Selecciona el banco 0
           ENABLE               ;Activa el LCD
           nop
LCD_BUSY_1     btfsc  PORTB,7        ;Chequea bit de Busy
           goto   LCD_BUSY_1           ;
           DISABLE              ;Desactiva LCD
           bsf    STATUS,RP0
           clrf   PORTB          ;Puerta B salida
           bcf    STATUS,RP0
           ESCRIBIR             ;Pone LCD en modo WR
           return

;*****************************************************************************
;LCD_E: Pulso Enable. En los  LCD's esta señal debe estar a "0" unos 40uS antes de
;volver a activarse a "1".
;
LCD_E          ENABLE              ;Activa E
                 nop
           DISABLE             ;Desactiva E
                 movlw  .14
                 movwf  Lcd_Temp_1
LCD_E_1                decfsz  Lcd_Temp_1,F ;Pierde unos 40 uS para la constante de tiempo
Tc
                 goto   LCD_E_1                ;de los nuevos módulos LCD de winteck
                 return
;
;*****************************************************************************
;LCD_DATO: Escritura de datos en DDRAM o CGRAM. Envía el dato presente en el W
;
LCD_DATO       OFF_COMANDO           ;Desactiva RS (modo comando)
           movwf  PORTB          ;Valor ASCII a sacar por portb
           call   LCD_BUSY       ;Espera a que se libere el LCD
           ON_COMANDO              ;Activa RS (modo dato).
           goto   LCD_E          ;Genera pulso de E

;*****************************************************************************
;LCD_REG: Escritura de comandos en el LCD. Envía el comando presente en el W

LCD_REG        OFF_COMANDO           ;Desactiva RS (modo comando)
           movwf  PORTB          ;Código de comando.
           call   LCD_BUSY       ;LCD libre?.
           goto   LCD_E          ;SI.Genera pulso de E.

;*****************************************************************************

;LCD_INI: inicialización del LCD enviando el comando "Function Set" 3 veces consecutivas
;con un intervalo de unos 5 mS. El LCD queda borrado y el cursor en la primera posición

LCD_INI                movlw  b'00111000'
                 call   LCD_REG                ;Código de instrucción
                 call   LCD_DELAY    ;Temporiza
                 movlw  b'00111000'
                 call   LCD_REG                ;Código de instrucción
                 call   LCD_DELAY    ;Temporiza
```

```
                movlw   b'00111000'
                call    LCD_REG                 ;Código de instrucción
                call    LCD_DELAY     ;Temporiza
                movlw   b'00000001'   ;Borra LCD y Home.
        call    LCD_REG
        return

;*****************************************************************************
;LCD_DELAY: Rutina de temporización de unos 5 mS. Se emplean las variables Lcd_Temp_1
;y LCD_Temp_2 en lugar del TMR0. Este queda libre para las aplicaciones del usuario

LCD_DELAY:  clrwdt
            movlw       .10
            movwf       Lcd_Temp_1
             clrf       Lcd_Temp_2
LCD_DELAY_1:            decfsz  Lcd_Temp_2,F
                goto    LCD_DELAY_1
                decfsz  Lcd_Temp_1,F
                goto    LCD_DELAY_1
                return

;*****************************************************************************
;*****************************************************************************
Ejemplo
;Este ejemplo pretende introducirnos en el manejo de la pantalla LCD,
;para la visualización de diferentes mensajes (p.e. Hola).
;
;Debe recordarse que las líneas RA0-RA2 actúan ahora como salida de
;señales de control hacia el LCD. Al estar conectadas con sendos
;interruptores en la MicroPIC Trainer, estos deben estar a nivel
;lógico "1" permanentemente.


                List    p=16F84                 ;Tipo de procesador
                Include "P16F84.INC"    ;Definiciones de registros internos

 Lcd_var        equ     0x0c            ;Variables (2) de rutinas  del LCD

                org     0x00            ;Vector de Reset
                goto    Inicio
                org     0x05            ;Salva vector de interrupción

                include "LCD_Cxx.inc"   ;Rutinas de manejo del LCD

;*****************Configuracion de pins*************************
 Inicio  clrf   PORTB           ;Borra los latch de salida
                bsf     STATUS,RP0      ;Selecciona banco 1
                clrf    TRISB           ;Puerta B se configura como salida
                movlw   b'00011000'
                movwf   TRISA           ;RA0-RA2 salidas, RA3-RA4 entradas

                bcf     STATUS,RP0      ;Selecciona banco 0

;****************Transferir comandos y datos***********************
                call    LCD_INI                 ;Secuencia de inicio del LCD
                movlw   b'00001111'
                call    LCD_REG                 ;Envía instrucc LCD ON,Cursor ON,Blink ON

                movlw   'H'
                call    LCD_DATO        ;Visualiza H
```

```
                movlw   'o'
                call    LCD_DATO        ;Visualiza o
                movlw   'l'
                call    LCD_DATO        ;Visualiza l
                movlw   'a'
                call    LCD_DATO        ;Visualiza a
                movlw   ' '
                call    LCD_DATO        ;Visualiza blanco

Loop            sleep                   ;Puesta en Standby
                goto    Loop            ;Vuelta a standby

                end                     ;Fin del programa fuente
```

Ejemplo

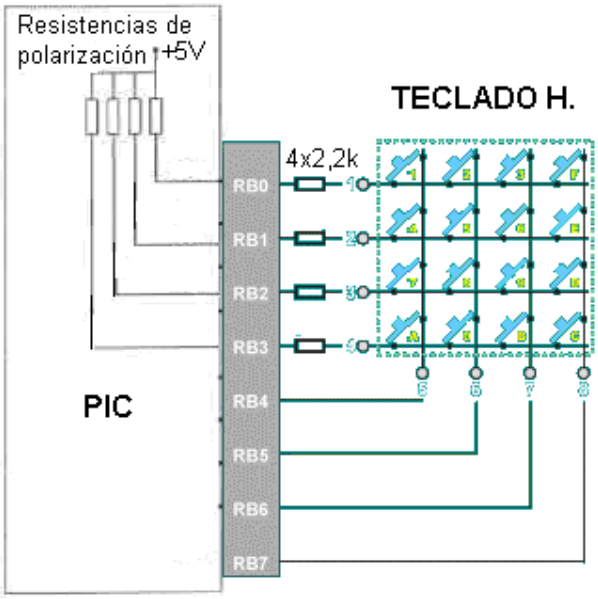
```
;Este ejemplo pretende introducirnos en el manejo de la pantalla LCD, para la ;visualizaciónde diferentes mensajes.
;
;Debe recordarse que las líneas RA0-RA2 actúan ahora como salida de señales de control
;hacia el LCD. Al estar conectadas con sendos interruptores en la MicroPIC Trainer, ;estos deben estar a nivel lógico "1" permanentemente.

                List    p=16F84                 ;Tipo de procesador
                include "P16F84.INC"    ;Definiciones de registros internos

Lcd_var                 equ     0x0c            ;Variables (2) de las rutinas de manejo del LCD
Delay_Cont      equ     0x0e            ;Variable para la temporización
Temporal_1      equ     0x0f            ;Variable temporal
Temporal_2      equ     0x10            ;Variable temporal

                org     0x00            ;Vector de Reset
                goto    Inicio
                org     0x05            ;Salva vector de interrupción

                include "LCD_Cxx.inc"   ;Incluye las rutinas de manejo del LCD

;*****************************************************************************
;
;Según el valor contenido en el registro W, se devuelve el carácter a visualizar

Tabla_Mensajes          movwf PCL               ;Calcula el desplazamiento sobre la tabla

Mens_0                  equ     $               ;Mens_0 apunta al primer carácter del ;mensaje 0
                retlw   'H'
                retlw   'o'
                retlw   'l'
                retlw   'a'
                retlw   0x00            ;Ultimo carácter del mensaje 0

Mens_1                  equ     $               ;Mens_1 apunta al primer carácter ;del mensaje 1
                retlw   'A'
                retlw   'd'
                retlw   'i'
                retlw   'o'
                retlw   's'
                retlw   0x00            ;Ultimo carácter del mensaje 1
```

```
;*******************************************************************************
;
;Delay_var: Esta rutina de propósito general realiza una temporización variable
;entre 50 mS y 12.8". Se emplea un preescaler de 256 y al TMR0 se le carga con 195.
;La velocidad de trabajo es de 4Mhz y por tanto el TMR0 se incrementa cada uS. De
;esta forma, el TMR0 debe contar 195 eventos que, con un preescaler de 256 hace una
;intervalo total de 50000 uS/50 mS (195 * 256). El valor 195 hay que expresarlo
;en Hex. (c3) y como el TMR0 es ascendente habrá que cargar su complemento (3C hex.)
;Dicho intervalo de 50 mS se repite tantes veces como indique la variable "Delay_cont",
;es por ello que el delay mínimo es de 50 mS ("Delay_cont=1) y el máxima de 12.8"
;(Delay_cont=255).

Delay_var:      bcf     INTCON,T0IF   ;Desconecta el flag de rebosamiento
                movlw   0x3c          ;Complemento hex. de 195
                movwf   TMR0          ;carga el TMR0
Intervalo       clrwdt                ;Refrescar el WDT
                btfss   INTCON,T0IF   ;Rebasamiento del TMR0 ??
                goto    Intervalo     ;Todavía no
                decfsz  Delay_Cont,F  ;Decrementa contador de intervalos
                goto    Delay_var     ;Repite el intervalo de 50 mS
                return

;*******************************************************************************************
;
;Mensaje: Esta rutina visualiza en el LCD el mensaje cuyo inicio está  indicado en
;el acumulador. El fin de un mensaje se determina mediante el código 0x00

Mensaje         movwf   Temporal_1      ;Salva posición de la tabla
Mensaje_1       movf    Temporal_1,W    ;Recupera posición de la tabla
            call    Tabla_Mensajes  ;Busca caracter de salida
            movwf   Temporal_2      ;Guarda el caracter
            movf    Temporal_2,F
            btfss   STATUS,Z        ;Mira si es el último
            goto    No_es_ultimo
            return
No_es_ultimo    call    LCD_DATO        ;Visualiza en el LCD
            incf    Temporal_1,F    ;Siguiente caracter
            goto    Mensaje_1

;*******************************************************************************
;
;
Inicio  clrf    PORTB                   ;Borra los latch de salida
                bsf     STATUS,RP0      ;Selecciona banco 1
                clrf    TRISB           ;Puerta B se configura como salida
                movlw   b'00011000'
                movwf   TRISA           ;RA0-RA2 salidas, RA3-RA4 entradas

                movlw   b'00000111'
                movwf   OPTION_REG  ;Preescaler de 256 para el TMR0
                bcf     STATUS,RP0  ;Selecciona banco 0

                call    LCD_INI             ;Secuencia de inicio del LCD
                movlw   b'00001100'
                call    LCD_REG             ;Envía instrucción: LCD ON, Cursor ;OFF y
blink OFF

Loop            movlw   b'00000001'
                call    LCD_REG             ;Borra LCD y Home (colocar cursor ;en 1ª
posición)
                movlw   Mens_0
```

```
                call    Mensaje             ;Visualiza el mensaje 0
                movlw   .20
                movwf   Delay_Cont
                call    Delay_var       ;Temporiza 2 segundos
                movlw   b'00000001'
                call    LCD_REG             ;Borra LCD y Home (colocar cursor ;en 1ª
posición)
                movlw   Mens_1
                call    Mensaje             ;Visualiza el mensaje 1
                movlw   .20
                movwf   Delay_Cont
                call    Delay_var       ;Temporiza 2 segundos
                goto    Loop

                end                     ;Fin del programa fuente
```

10-Teclado Hexadecimal.

Descripción: Dispositivo de entrada de datos que consta de 16 teclas o pulsadores, dispuestos e interconectados en forma matricial filas y columnas. Al pulsar una tecla se realiza el contacto de una fila con una columna. La detección de tecla pulsada se realiza mediante la exploración del teclado

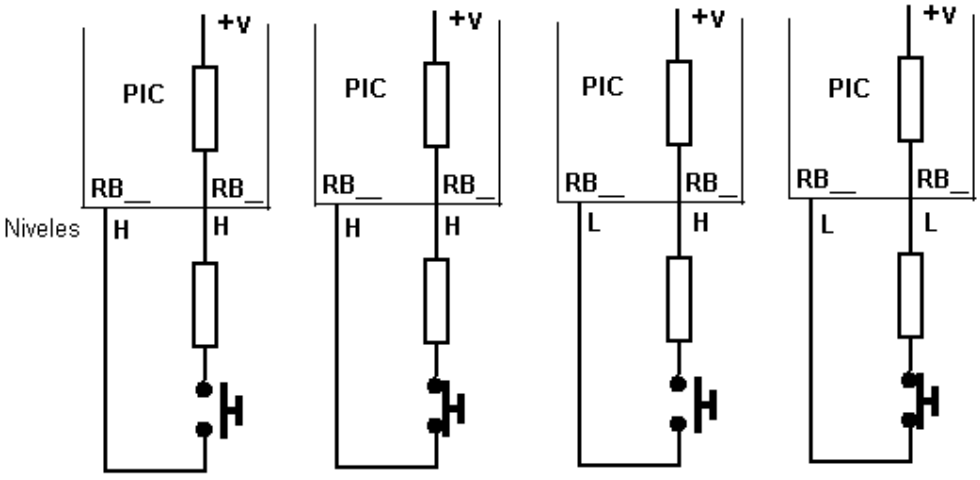


Funcionamiento. Durante la fase de exploración del teclado, cuatro lineas del Port B (RB7-4) es configurada como salida y las otras cuatro como entrada (RB3-RB0), en configuración pull-up.
En los terminales de salida se introducen secuencialialmente señales de 4 Bits de los cuales tres estan a nivel alto y uno a nivel bajo.

Conociendo el codigo de los cuatro Bits de salida sabemos cual columna se encuentra a nivel bajo, y por la entrada a nivel bajo podermos conocer la fila de la tecla pulsada.
En caso de la tecla pulsada podemos conocer el valor de RB7-RB0, y a cada valor se le hace corresponder un determinado código guardado en la tabla ROM.

El valor de entrada se carga en W, y produce un desplazamiento en la dirección del PCL, de tal forma que mediante una subrutina el PCL apunta a la dirección de la tabla ROM correspondiente al código asignado al valor de la entrada.
Se carga este código en W para su proceso y se produce el retorno de subrutina.

Ejemplos de los casos para la obtención del nivel bajo en las entradas del PIC



Tablas de datos en memoria ROM
Consta de una serie de datos ordenados secuencialmente, que para su lectura se salta a una subrutina que suma el valor del contador de programa PCL al contenido del registro W, y el resultado se deposita en PCL mediante la instrucción addwf PCL,F
PCL apunta a la dirección del dato concreto dentro de la tabla y lo coloca en W para poder ser procesado, y se produce el retorno de la subrutina.

; Ejemplo, la tabla será de 3 entradas y 6 salidas:

```
;
;               C  B  A  | S5 S4  S3   S2 S1   S0
;               ----------------------------------------------
;               0  0  0  |  0  0   1   0  1   0 ; (Configuración 0).
;               0  0  1  |  0  0   1   0  0   1 ; (Configuración 1).
;               0  1  0  |  1  0   0   0  1   1 ; (Configuración 2).
;               0  1  1  |  0  0   1   1  1   1 ; (Configuración 3).
;               1  0  0  |  1  0   0   0  0   0 ; (Configuración 4).
;               1  0  1  |  0  0   0   1  1   1 ; (Configuración 5).
;               1  1  0  |  0  1   0   1  1   1 ; (Configuración 6).
;               1  1  1  |  1  1   1   1  1   1 ; (Configuración 7).
;
; Las entradas C, B, A se conectarán a las líneas del puerto A: RA2(C), RA1(B) yRA0 (A).
; Las salidas se obtendrán del puerto B: RB5 (S5), RB4 (S4), RB3 (S3), RB2 (S2),RB1 (S1)
; y RB0 (S0).
;
; ZONA DE DATOS *****************************************************************

        __CONFIG   _CP_OFF &  _WDT_OFF & _PWRTE_ON & _XT_OSC
        LIST       P=16F84A
        INCLUDE  <P16F84A.INC>
```

```
; ZONA DE CÓDIGOS ******************************************************************

        ORG     0                       ; El programa comienza en la dirección 0.
Inicio
        bsf     STATUS,RP0              ; Acceso al Banco 1.
        clrf    TRISB                   ; Las líneas del Port B se configuran como salida.
        movlw   b'00011111'             ; Las 5 lín. del Port A se configuran como entrada.
        movwf   TRISA
        bcf     STATUS,RP0              ; Acceso al Banco 0.
Principal
        movf    PORTA,W                     ; Lee el valor de las variables de entrada.
        andlw   b'00000111'             ; Se queda con los tres bits bajos de entrada.
        call    TablaVerdad             ; Obtiene la configuración de salida.
        movwf   PORTB                   ; Se visualiza por el puerto de salida.
        goto    Principal

; Subrutina "TablaVerdad" ---------------------------------------------------------
;
TablaVerdad
        addwf   PCL,F
        retlw   b'00001010'             ; (Configuración 0).
        retlw   b'00001001'             ; (Configuración 1).
        retlw   b'00100011'             ; (Configuración 2).
        retlw   b'00001111'             ; (Configuración 3).
        retlw   b'00100000'             ; (Configuración 4).
        retlw   b'00000111'             ; (Configuración 5).
        retlw   b'00010111'             ; (Configuración 6).
        retlw   b'00111111'             ; (Configuración 7).

        END
```

La directiva DT
Simplifica la programación, evitando la utilización repetitiva de la directiva retlw

Ejemplo

```
; Simplificar el ejercicio anterior  utilizando la directiva DT.
;
; ZONA DE DATOS ********************************************************************

        __CONFIG   _CP_OFF &  _WDT_OFF & _PWRTE_ON & _XT_OSC
        LIST    P=16F84A
        INCLUDE  <P16F84A.INC>

; ZONA DE CÓDIGOS ******************************************************************

        ORG     0
Inicio
        bsf     STATUS,RP0
        clrf    PORTB
        movlw   b'00011111'
        movwf   PORTA
        bcf     STATUS,RP0
Principal
        movf    PORTA,W                 ; Lee el valor de las variables de entrada.
        andlw   b'00000111'             ; Se queda con los tres bits de entrada.
        call    TablaVerdad             ; Obtiene la configuración de salida.
        movwf   PORTB                   ; Se visualiza por el puerto de salida.
        goto    Principal
; Subrutina "TablaVerdad" ---------------------------------------------------------
```

```
;
TablaVerdad
        addwf   PCL,F
        DT      0x0A, 0x09, 0x23, 0x0F, 0x20, 0x07, 0x17, 0x03F ; Configuraciones
                                                                ; de salida.
        END

LIBRERIAS PARA MANEJO DEL TECLADO
;***************************** Librería "TECLADO.INC" ***********************************
;
;
; Librería de subrutinas para la gestión de un teclado organizado en una matriz de 4x4 y
; conectado al Puerto B según la disposición siguiente
;
;                   RB4 RB5 RB6 RB7
;                    ^   ^   ^   ^
;                   |----|----|----|----|
;         RB0 -->|  0 |  1 |  2 |  3 |
;                   |----|----|----|----|
;         RB1 -->|  4 |  5 |  6 |  7 |
;                   |----|----|----|----|
;         RB2 -->|  8 |  9 | 10 | 11 |
;                   |----|----|----|----|
;         RB3 -->| 12 | 13 | 14 | 15 |
;                   |----|----|----|----|
;
; Los números que se han dibujado dentro de cada cuadrado son el orden de las teclas
; que no tienen por qué coincidir con lo serigrafiado sobre ellas. El paso del número de
;orden de la tecla al valor que hay serigrafiado sobre la misma se hace con una tabla de
; conversión.
; ZONA DE DATOS ********************************************************************
;
        CBLOCK
        Tecl_TeclaOrden                 ; Orden de la tecla a chequear.
        ENDC

Tecl_UltimaTecla        EQU     d'15'           ; Valor de orden de última tecla utilizada.

; Subrutina "Teclado_LeeHex" ***************************************************
;
; Cada tecla tiene asignado un número de orden que es contabilizado en la variable
; Tecl_TeclaOrden. Para convertir a su valor según el tipo de teclado en concreto se
; utiliza una tabla de conversión.
; A continuación se expone la relación entre el número de orden de la tecla y los
; valores correspondientes para el teclado hexadecimal más utilizado.
;
;
;
;
;               ORDEN DE TECLA:                 TECLADO HEX. UTILIZADO:
;                   0   1   2   3               1  2  3  F
;                   4   5   6   7               4  5  6  E
;                   8   9  10  11               7  8  9  D
;                  12  13  14  15               A  0  B  C
;
; Así, en este ejemplo, la tecla "7" ocupa el orden 8, la tecla "F" ocupa el orden 3 y ;la tecla "9"
el orden 10.

; Si cambia el teclado también hay cambiar de tabla de conversión.
;
```

```
; Entrada:      En (W) el orden de la tecla pulsada.
; Salida:       En (W) el valor hexadecimal para este teclado concreto.
;
Teclado_LeeHex
        call    Teclado_LeeOrdenTecla       ; Lee el Orden de la tecla pulsada.
        btfss   STATUS,C            ; ¿Pulsa alguna tecla?, ¿C=1?
        goto    Tecl_FinLeeHex              ; No, por tanto sale.
        call    Tecl_ConvierteOrdenEnHex; Lo convierte en su valor real mediante tabla.
        bsf     STATUS,C            ; Vuelve a posicionar el Carry, porque la
Tecl_FinLeeHex                              ; instrucción "addwf PCL,F" lo pone a "0".
        return
;
Tecl_ConvierteOrdenEnHex                    ; Según el teclado utilizado resulta:
        addwf   PCL,F
        DT      1h,2h,3h,0Fh        ; Primera fila del teclado.
        DT      4h,5h,6h,0Eh        ; Segunda fila del teclado
        DT      7h,8h,9h,0Dh        ; Tercera fila del teclado.
        DT      0Ah,0h,0Bh,0Ch              ; Cuarta fila del teclado.
Teclado_FinTablaHex
;
; Esta tabla se sitúa al principio de la librería con el propósito de que no supere la
; posición 0FFh de memoria ROM de programa. De todas formas, en caso que así fuera
; visualizaría el siguiente mensaje de error en el proceso de ensamblado:
;
        IF (Teclado_FinTablaHex > 0xFF)
                ERROR       "Atención: La tabla ha superado el tamaño de la página de los"
                MESSG       "primeros 256 bytes de memoria ROM. NO funcionará
correctamente."
        ENDIF

; Subrutina "Teclado_Inicializa" ----------------------------------------------------
;
; Esta subrutina configura las líneas del Puerto B según la conexión del teclado
; y comprueba que no hay pulsada tecla alguna al principio.

Teclado_Inicializa
        bsf     STATUS,RP0          ; Configura las líneas del puerto:
        movlw   b'11110000'         ; <RB7:RB4> entradas, <RB3:RB0> salidas
        movwf   PORTB
        bcf     OPTION_REG,NOT_RBPU     ; Habilita resistencia de Pull-Up del Puerto B.
        bcf     STATUS,RP0          ; Acceso al banco 0.
;       call    Teclado_EsperaDejePulsar
;       return
;
; Subrutina "Teclado_EsperaDejePulsar" ----------------------------------------------
;
;Permanece en esta subrutina mientras siga pulsada la tecla.
;
Teclado_ComprobacionEQU     b'11110000'

Teclado_EsperaDejePulsar:
        movlw   Teclado_Comprobacion; Pone a cero las cuatro líneas de salida del
        movwf   PORTB               ; Puerto B.
Teclado_SigueEsperando
        call    Retardo_20ms        ; Espera a que se estabilicen los niveles de tensión.
        movf    PORTB,W                 ; Lee el Puerto B.
        sublw   Teclado_Comprobacion; Si es lo mismo que escribió es que ya no pulsa
        btfss   STATUS,Z            ; tecla alguna.
        goto    Teclado_SigueEsperando
        return
```

```
;
; Subrutina "Teclado_LeeOrdenTecla" ----------------------------------------------
;
; Lee el teclado, obteniendo el orden de la tecla pulsada.
;
; Salida:En (W) el número de orden de la tecla pulsada. Además Carry se pone a "1" si
;               se pulsa una tecla ó a "0" si no se pulsa tecla alguna.
;
Teclado_LeeOrdenTecla:
        clrf    Tecl_TeclaOrden         ; Todavía no ha empezado a chequear el teclado.
        movlw   b'11111110'             ; Va a chequear primera fila.
Tecl_ChequeaFila                        ; (Ver esquema de conexión).
        movwf   PORTB                   ; Activa la fila correspondiente.
Tecl_Columna1
        btfss   PORTB,4                         ; Chequea la 1ª columna buscando un cero.
        goto    Tecl_GuardaValor        ; Sí, es cero y por tanto guarda su valor y sale.
        incf    Tecl_TeclaOrden,F       ; Va a chequear la siguiente tecla.
Tecl_Columna2                           ; Repite proceso para las siguientes
        btfss   PORTB,5                         ; columnas.
        goto    Tecl_GuardaValor
        incf    Tecl_TeclaOrden,F
Tecl_Columna3
        btfss   PORTB,6
        goto    Tecl_GuardaValor
        incf    Tecl_TeclaOrden,F
Tecl_Columna4
        btfss   PORTB,7
        goto    Tecl_GuardaValor
        incf    Tecl_TeclaOrden,F
;
; Comprueba si ha chequeado la última tecla, en cuyo caso sale. Para ello testea si
; el contenido del registro Tecl_TeclaOrden es igual al número de teclas del teclado.
;
Tecl_TerminaColumnas
        movlw   Tecl_UltimaTecla
        subwf   Tecl_TeclaOrden,W       ; (W) = (Tecl_TeclaOrden)-Tecl_UltimaTecla.
        btfsc   STATUS,C                ; ¿C=0?, ¿(W) negativo?, ¿(Tecl_TeclaOrden)<15?
        goto    Tecl_NoPulsada                  ; No, se ha llegado al final del chequeo.
        bsf     STATUS,C                ; Sí. Va a chequear la siguiente fila.
        rlf     PORTB,W                         ; Apunta a la siguiente fila.
        goto    Tecl_ChequeaFila
Tecl_NoPulsada
        bcf     STATUS,C                ; Posiciona C=0, indicando que no ha pulsado
        goto    Tecl_FinTecladoLee      ; tecla alguna y sale.
Tecl_GuardaValor
        movf    Tecl_TeclaOrden,W       ; El orden de la tecla pulsada en (W) y sale.
        bsf     STATUS,C                ; Como hay tecla tecla pulsada, pone C=1.
Tecl_FinTecladoLee
        return
```

Ejemplo de ejercicio con teclado H y LCD

La siguiente librería configura las salidas y entradas para usar el teclado escanea las teclas y si es pulsada alguna retorna el valor de tecla en modo hexadecimal para ser mostrado directamente en un display LCD. Las variables y subrutinas utilizadas son:

KB_Port	Configura los puertos B del PIC para ser usados con el teclado.
KB_Scan	Escanea todas las teclas y devuelve la tecla pulsada en formato Hexadecimal.

Tecla Variable en donde se devuelve la tecla pulsada

```
KB_Port
                    ;Inicializa la puerta B para las entradas del
                    ;teclado Programa RB0-3 como salidas  RB4-7
                    ;entradas con las resistencias de polarización
                    ;habilitadas----------------------------
        BSF     STATUS,RP0      ;Selecciona Pagina 1 (Banco)
        MOVLW     0F0h          ;PB4-7 como entradas
        MOVWF     TRISB         ;y PB0-3 como salidas
        BSF     OPCION,RBPU     ;Habilita R de polarización en entradas
        BCF     STATUS,RP0      ;Vuelve a la pagina 0 (Banco)
        RETURN                  ;Retorna.
```

Ahora para saber que tecla esta siendo pulsada necesitamos escanear el teclado, y esto se consigue mediante la siguiente rutina que consiste en ir poniendo una a una las líneas RB0 a RB3 (conectadas a las filas del teclado) a nivel bajo ya que estas salidas tienen conectada la resistencia de polarización a Vcc y por lo tanto están siempre a nivel lógico alto. Cada vez que una fila se pone a nivel bajo se hacen 4 comprobaciones para ver si una de las cuatro columnas se a puesto a nivel bajo y así saber la tecla pulsada.

KB-SCAN

```
                        ;Escanea el teclado
        CLRF    Tecla           ;Borra Tecla y
        INCF    Tecla,f         ;prepara Tecla para primer código.
        MOVLW   0Eh             ;Saca 0 a la primera fila
        MOVWF   PORTB           ;de la Puerta B
        NOP                     ;Nada para estabilización de señal.
Cheq_Col    BTFSS   PORTB,4     ;Primera columna = 0
        GOTO    antirebotes     ;Sale si se ha pulsado tecla.
        INCF    Tecla,f         ;Si no tecla pulsada, incrementa tecla.
        BTFSS   PORTB,5         ;Segunda columna = 0
        GOTO    antirebotes     ;Sale si se ha pulsado tecla.
        INCF    Tecla,f         ;Si no tecla pulsada, incrementa tecla.
        BTFSS   PORTB,6         ;Tercera columna = 0
        GOTO    antirebotes     ;Sale si se ha pulsado tecla.
        INCF    Tecla,f         ;Si no tecla pulsada, incrementa tecla.
        BTFSS   PORTB,7         ;Cuarta columna = 0
        GOTO    antirebotes     ;Sale si se ha pulsado tecla.
        INCF    Tecla,f         ;Si no tecla pulsada, incrementa Tecla.
Ultima_Tecla                    ;comprueba si se a escaneado todo el teclado
        MOVLW   d'17'           ;Carga W con el n£mero de Teclas + 1.
        SUBWF   Tecla,w         ;y lo compara con el valor actual de Tecla.
        BTFSC   STATUS,Z        ;Si Tecla + 1 = valor actual.
        GOTO    NTeclas         ;No ha sido pulsada ninguna tecla.
        BSF     STATUS,C        ;Pone a 1 Bit C.
        RLF     PORTB,f         ;as¡ la Fila 1 pasa a 1 con la rotar a  izda
        GOTO    Cheq_Col

        NTeclas CLRF Tecla      ;Coloca variable Tecla a 0
        RETURN                  ;y regresa.

antirebotes         ;ahora se espera a que la tecla sea soltada para evitar rebotes
                    ;y reactivaciones de tecla
                    ;esta parte puede ser eliminada si para nuestro proyecto no es
                    ;necesaria o es un inconveniente.

Espera1 BTFSS   PORTB,4         ;Si no se suelta la tecla FILA 1
        GOTO    Espera1         ;vuelve a esperar.
Espera2 BTFSS   PORTB,5         ;Si no se suelta la tecla FILA 2
        GOTO    Espera2         ;vuelve a esperar.
```

```
Espera3   BTFSS   PORTB,6          ;Si no se suelta la tecla FILA 3
          GOTO    Espera3          ;vuelve a esperar.
Espera4   BTFSS   PORTB,7          ;Si no se suelta la tecla FILA 4
          GOTO    Espera4          ;vuelve a esperar.

          MOVF    Tecla,w     ;pone en w el numero contenido en la variable
          CALL    T_Conv      ;llama a la tabla de conversion y retorna con
          MOVWF Tecla         ;el valor en hexadecimal y lo pone en la variable.
          RETURN              ;vuelve al programa principal que hizo la llamada.
;-------------------------------------------------------------------------------

T_Conv       ADDWF   PCL,1
          RETLW   '0'               ;Tecla nº0 = 0
          RETLW   '1'               ;Tecla nº1 = 1
          RETLW   '4'               ;Tecla nº2 = 4
          RETLW   '7'               ;Tecla nº3 = 7
          RETLW   'A'               ;Tecla nº4 = A
          RETLW   '2'               ;Tecla nº5 = 2
          RETLW   '5'               ;Tecla nº6 = 5
          RETLW   '8'               ;Tecla nº7 = 8
          RETLW   '0'               ;Tecla nº8 = 0
          RETLW   '3'               ;Tecla nº9 = 3
          RETLW   '6'               ;Tecla nº10 = 6
          RETLW   '9'               ;Tecla nº11 = 9
          RETLW   'B'               ;Tecla nº12 = B
          RETLW   'F'               ;Tecla nº13 = F
          RETLW   'E'               ;Tecla nº14 = E
          RETLW   'D'               ;Tecla nº15 = D
          RETLW   'C'               ;Tecla nº16 = C
```

Parte de código puede ser eliminada sin ningún problema si no queremos que el pic tenga que esperar a que soltemos la tecla para continuar con su ejecución normal.

```
; Descripcion:
; Esta libreria realiza un escaneo en un teclado matricial de 4x4 teclas,
; el escaneo lo realiza poniendo un cero logico en las filas correspondientes
; y verificando las columnas para encontrar la interseccion provocada por
; el accionamiento de una tecla y asi saber la fila y columna de la tecla.
; ESTA SUBRUTINA CONVIERTE LA TECLA PULSADA EN SU EQUIVALENTE ASCII
;       para ser usada con el display LCD.
;
; Se tienen que inicializar las siguientes variables en el programa que
; use esta libreria:
;                   - ( Tecla )  ;Devuelve la tecla pulsada
;*******************************************************************************

KB_Port                   ;Inicializa la puerta B para las entradas del
                          ;teclado. Programa RB0-3 como salidas y RB4-7
                          ;entradas con las resistencias de polarizacion
                          ;habilitadas.
          BSF     STATUS,RP0      ;Selecciona P gina 1
          MOVLW   0F0h          ;PB4-7 como entradas
          MOVWF   TRISB          ;y PB0-3 como salidas
```

```
            BSF     OPCION,RBPU     ;Habilita R de polarizacion en entradas
            BCF     STATUS,RP0      ;Vuelve a la p gina 0.
            RETURN                  ;Retorna.

KB_Scan                             ;Escanea el teclado
            CLRF    Tecla           ;Borra Tecla y
            INCF    Tecla,f         ;prepara Tecla para primer c¢digo.
            MOVLW   0Eh             ;Saca 0 a la primera fila
            MOVWF   PORTB           ;de la Puerta B
            NOP                     ;Nada para estabilizaci¢n de señal.
Cheq_Col    BTFSS   PORTB,4         ;Primera columna = 0
            GOTO    antirebotes     ;Sale si se ha pulsado tecla.
            INCF    Tecla,f         ;Si no tecla pulsada, incrementa tecla.
            BTFSS   PORTB,5         ;Segunda columna = 0
            GOTO    antirebotes     ;Sale si se ha pulsado tecla.
            INCF    Tecla,f         ;Si no tecla pulsada, incrementa tecla.
            BTFSS   PORTB,6         ;Tercera columna = 0
            GOTO    antirebotes     ;Sale si se ha pulsado tecla.
            INCF    Tecla,f         ;Si no tecla pulsada, incrementa tecla.
            BTFSS   PORTB,7         ;Cuarta columna = 0
            GOTO    antirebotes     ;Sale si se ha pulsado tecla.
            INCF    Tecla,f         ;Si no tecla pulsada,incrementa Tecla.

Ultima_Tecla    MOVLW   d'17'       ;Carga W con el n£mero de Teclas + 1.
            SUBWF   Tecla,w         ;y lo compara con el valor actual de Tecla.
            BTFSC   STATUS,Z        ;Si Tecla + 1 = valor actual.
            GOTO    NTeclas         ;No ha sido pulsada ninguna tecla.
            BSF     STATUS,C        ;Pone a 1 Bit C.
            RLF     PORTB,f         ;as¡ la Fila 1 pasa a 1 con la rotaci¢n a izqda.
            GOTO    Cheq_Col

NTeclas     CLRF    Tecla           ;Coloca variable Tecla a 0
            RETURN                  ;y regresa.

antirebotes     ;ahora se espera a que la tecla sea soltada para evitar rebotes
                ;y reactivaciones de tecla
                ;esta parte puede ser eliminada si para nuestro proyecto no es
                ;necesaria es un inconveniente.

Espera1     BTFSS   PORTB,4         ;Si no se suelta la tecla FILA 1
            GOTO    Espera1         ;vuelve a esperar.
Espera2     BTFSS   PORTB,5         ;Si no se suelta la tecla FILA 2
            GOTO    Espera2         ;vuelve a esperar.
Espera3     BTFSS   PORTB,6         ;Si no se suelta la tecla FILA 3
            GOTO    Espera3         ;vuelve a esperar.
Espera4     BTFSS   PORTB,7         ;Si no se suelta la tecla FILA 4
            GOTO    Espera4         ;vuelve a esperar.

            MOVF    Tecla,w         ;pone en w el numero contenido en la variable
            CALL    T_Conv          ;llama a la tabla de conversion y retorna el
            MOVWF   Tecla           ;valor  hexadecimal y lo coloca en la variable.
            RETURN      ;vuelve al programa principal que hizo lallamada


T_Conv      ADDWF   PCL,1
            RETLW   '0'             ;Tecla nº0 = 0
            RETLW   '1'             ;Tecla nº1 = 1
            RETLW   '4'             ;Tecla nº2 = 4
            RETLW   '7'             ;Tecla nº3 = 7
            RETLW   'A'             ;Tecla nº4 = A
```

```
RETLW   '2'         ;Tecla nº5 = 2
RETLW   '5'         ;Tecla nº6 = 5
RETLW   '8'         ;Tecla nº7 = 8
RETLW   '0'         ;Tecla nº8 = 0
RETLW   '3'         ;Tecla nº9 = 3
RETLW   '6'         ;Tecla nº10 = 6
RETLW   '9'         ;Tecla nº11 = 9
RETLW   'B'         ;Tecla nº12 = B
RETLW   'F'         ;Tecla nº13 = F
RETLW   'E'         ;Tecla nº14 = E
RETLW   'D'         ;Tecla nº15 = D
RETLW   'C'         ;Tecla nº16 = C
```