

# PROGRAMACIÓN PIC (II)

## Microcontrolador PIC 16F84

mail : [enric.serra](mailto:enric.serra)

### 0 - INTRODUCCIÓN.

Este documento es una continuación al documento de programación del PIC 16f84.

Este documento se puede copiar y utilizar siempre que se indique la procedencia (Escola Professional Salesians Joan XXIII) y se indique su autor Enric Serra.

Todos los comentarios y bugs serán bien recibidos.

Listado de instrucciones:

**TABLE 7-2: PIC16CXXX INSTRUCTION SET**

Mnemonic, Operands	Description	Cycles	14-Bit Opcode				Sta Affe	
			MSb			LSb		
<b>BYTE-ORIENTED FILE REGISTER OPERATIONS</b>								
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C.D
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	
CLRF	f	Clear f	1	00	0001	1fff	ffff	
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00	1011	dfff	ffff	
INCF	f, d	Increment f	1	00	1010	dfff	ffff	
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00	1111	dfff	ffff	
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	
MOVF	f, d	Move f	1	00	1000	dfff	ffff	
MOVWF	f	Move W to f	1	00	0000	1fff	ffff	
NOP	-	No Operation	1	00	0000	0xxx0	0000	
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C.D
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff	
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	

BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		
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	TO,PD	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk	Z	
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	TO,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 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 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.

## 1 - SUBROUTINAS DE RETARDO.

Un retardo, es una forma de control de tiempo en la programación del PIC, Las instrucciones 'normales' utilizan un ciclo de máquina para ejecutarse, un ciclo máquina es la unidad básica de tiempo de ejecución de un programa en un PIC y depende de la velocidad del oscilador.

Hay instrucciones llamadas de salto como goto,return, call, btfsf etc que necesitan 2 ciclos máquina para ejecutarse.

Si contamos los ciclos máquina de una determinada parte de instrucciones de un programa, podremos controlar los tiempos de retardo.

Como sabemos que  $F=1/T$ , siendo  $F$ =frecuencia y  $T$ =tiempo, podemos determinar cuánto tiempo consumirá una instrucción en el microcontrolador, sabiendo que para ejecutar una instrucción el microcontrolador utiliza 4 pulsos de reloj.

Ej : Si un microcontrolador funciona a 4MHz, ¿qué tardará en ejecutar una instrucción?

$$F=1/4T \rightarrow T=1*4/F$$

$$\text{Si } F=4\text{MHz}$$

$$T=1/F = 1*4/4000000 = 1\text{useg}$$

Es decir que para un reloj de 4 MHz, cada instrucción simple tardará 1 useg, y para las instrucciones de salto tardará 2useg.

```

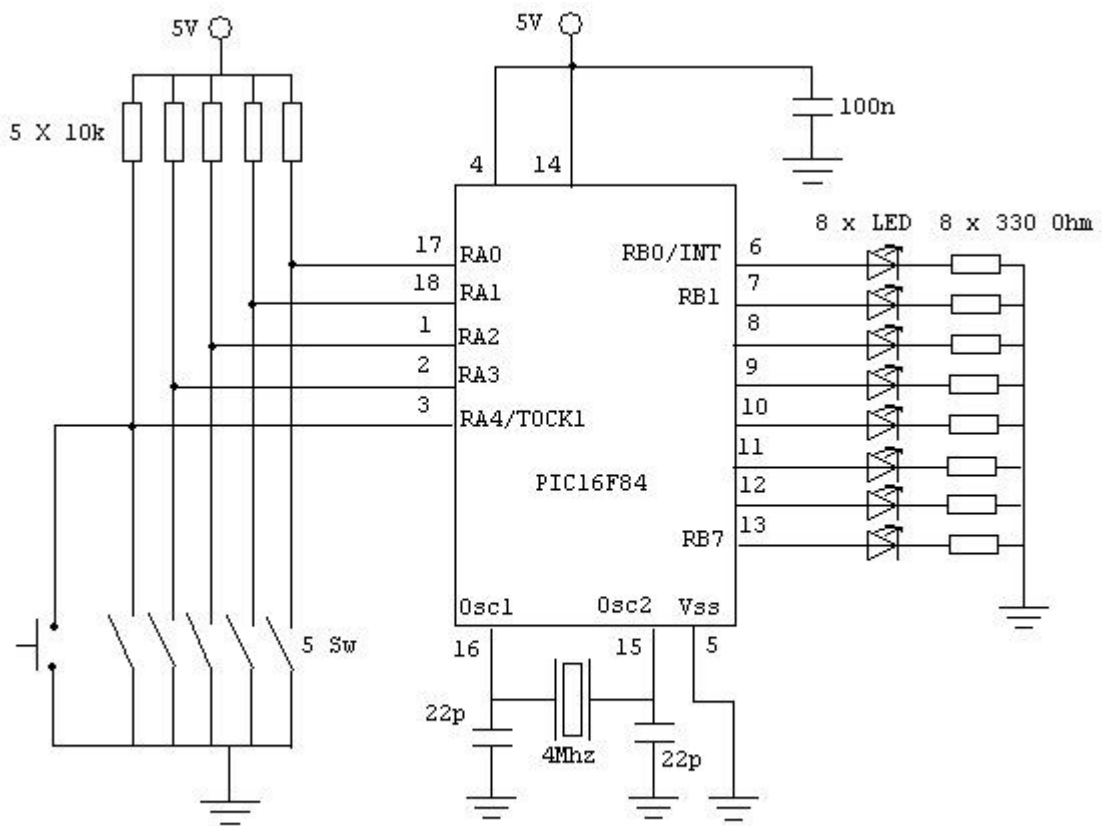
Feersum miSim DE 1.7.9
File Edit Debug Reset Current Step Over Run Stop About
Log View Plugins retardo0.asm (2)
;----- Programa bucle simple -----
LIST p=16f84
contador EQU h'10' ;Contador del bucle

ORG 0
;-----
BSF h'03',5 ;Bank 1
CLRF h'06' ;PortB Salidas
BCF h'03',5 ;Bank 0
;-----
inici MOVLW h'FF' ;Cargamos 11111111 en w
MOVWF h'06' ;enviamos todo '1' al port B
CALL retardo ;Subrutina retardo
CLRF h'06' ;Borramos el port B
CALL retardo ;Llamamos a la subrutina retardo
GOTO inici ;Repetimos el programa
;-----
retardo MOVLW d'249' ;Cargamos contador.La llamada al retardo son 2 cm
MOVWF contador ;con el valor 249 (1 cm)
;----- Inicio bucle retardo -----
bucle NOP ;NOP consume 1 ciclo máquina (CM)
DECFSZ contador,1 ; 1 cm si no salto y 2 cm al saltar
GOTO bucle ;consume 2 cm
RETURN ;2 cm
;-----
END
;-----
;-- El retardo se calcula de la siguiente forma:
;-- 2 + 1 + 1 + (249)*1+(249-1)*1 +2+ (249-1)*2 +2
;-- 1001 useg = 1 mseg

```

File 'retardo0.asm' saved OK

1.1 Práctica: Montar el siguiente circuito y grabar el pic con el ejercicio anterior.



ENTRENADOR PIC 16F84

Podemos realizar mas retardos con bucles anidados, es decir uno dentro del otro para conseguir retardos de milisegundos y de segundos dependiendo del valor que podamos cargar a los contadores.

Veamos un ejemplo.

```

;-----
;--      Subrutinas retardo mediante bucles
;--      leemos port A y realizamos un menu
;--      con los valores leidos de la siguiente forma:
;--      Si porta=0 retardo 200ms
;--      si porta=1 retardo 1 seg
;--      si porta=2 retardo 5 seg
;--      si porta=3 retardo 10 seg
;-----

list          p=16f84
contador1    equ   h'10'  ;Variable contador
contador2    equ   h'11'
contador3    equ   h'12'
portb        equ   h'06'
porta        equ   h'05'
status       equ   h'03'
pcl          equ   h'02'  ;Contador programa
org          0

;-----Configuracion ports -----
bsf          status,5      ;bank1
clrf         portb         ;portb salidas
movlw       b'00011111'
movwf       porta         ;porta entradas
bcf         status,5      ;bank0

```

```

;-----
;--          Programa principal          --
;-----
prog movf  porta,0
    andlw  b'00000011'    ;Solo 3 ultimos bits
    addwf  pcl,1
    goto   prog0          ;Si porta=0 ->prog0
    goto   prog1          ;Si porta=1 ->prog1
    goto   prog2          ;Si porta=2 ->prog2
    goto   prog3          ;Si porta=3 ->prog3
    goto   prog           ;Bucle principal
;----- Subprogramas retardos -----
prog0 ;-- Subprograma 0 (200ms)
    movlw  b'00000001'    ;enviamos un 1
    movwf  portb          ;al puerto b
    movlw  d'200'         ;cargamos contador1 con 200
    call   miliseg        ;llamamos subrutina milisegundos (tiempo
en on)
    clrf   portb          ;enviamos un 0 al portB
    movlw  d'200'         ;cargamos contador1 con 200
    call   miliseg        ;volvemos a llamar subrutina (tiempo en off)
    goto   prog
prog1 ;-- Subprograma 1 1 seg
    movlw  b'00000001'
    movwf  portb
    movlw  d'10'          ;cargamos contador3 con 10
    call   segundo        ;llamada subrutina segundo (tiempo en on)
    clrf   portb
    movlw  d'10'          ;cargamos contador3 con 10
    call   segundo        ;llamada subrutina segundo (tiempo en off)
    goto   prog
prog2 ;-- Subprograma 2 (5 seg)
    movlw  b'00000001'
    movwf  portb
    movlw  d'50'
    call   segundo
    clrf   portb
    movlw  d'50'
    call   segundo
    goto   prog
prog3 ;-- Subprograma 3 (10 seg)
    movlw  b'00000001'
    movwf  portb
    movlw  d'100'
    call   segundo
    clrf   portb
    movlw  d'100'
    call   segundo
    goto   prog
;----- Subrutinas de retardo con bucles anidados -----
;----- Retardo milisegundos con 2 bucles anidados-----
;-----
miliseg ;-- Subrutina Retardo milisegundos
    movwf  contador2      ;cargamos contador2 con 200
bucle1 movlw d'249'        ;y contador 1 con 249

```

```

    movwf contador1      ;Aprox 200 ms
bucle2 nop              ;No Operation (no hace nada,pasa un ciclo de
maquina)
    decfsz contador1,1   ;Decrementamos contador 1
    goto bucle2          ;Si contador1 distinto 0 repite operacion
    decfsz contador2,1   ;Si contador1=0 decrementados contador2
    goto bucle1          ;Si contador2 no es cero -> vamos a bucle1
    return               ;Si contador2=0 finaliza subrutina
;----- Retardo en segundos con 3 bucles anidados -----
;-----
segundo ;-- Subrutina Retardo segundos
    movwf contador3     ;cargamos contador3 con valor de
prog1,prog2,prog3
bucle3 movlw d'100'     ;y contador 2 con 100
    movwf contador2
bucle4 movlw d'249'    ;cargamos contador1 con 249
    movwf contador1
bucle5 nop              ;NOP (un ciclo de maquina 1u segundo)
    decfsz contador1,1
    goto bucle5
    decfsz contador2,1
    goto bucle4
    decfsz contador3,1
    goto bucle3
    return
;----- Final de subrutinas de retardo -----
end

```

1.2 Práctica: Con el circuito del ejercicio 1.1 grabar el pic con el ejercicio anterior y comprobar su funcionamiento.

1.3 Práctica: Modifica el programa para obtener 1 seg, 2 seg y 5 seg.

1.4 Modifica el programa para obtener 60 segundos (1 minuto)

## 2. SALTOS INDEXADOS

Los saltos indexados permiten realizar tablas de verdad o realizar menus a subprogramas dependiendo de las entradas.

Para ello, se actua sobre el registro h'02' es decir el CONTADOR DE PROGRAMA' o PCL.

La técnica consiste en añadir un valor de desplazamiento al contador de programa (este valor de desplazamiento se denomina 'offset') mediante un comando de suma.

EJEMPLO	NOTA
addwf PCL,1	Suma al registro PCL (Contador de programa) el valor de W

### EJEMPLO

El programa genera un salto indexado utilizando el comando addwf h'02',1

de forma que mediante dos interruptores accedemos a 4 diferentes subprogramas denominados prg00,prg01,prg02 y prg03. En cada subprograma sacamos por el display el valor leído en el puerto de entrada portA.

```

;-----
;--          Programa de salts indexats          ---
;-----
;--  El programa tria una eixida depenent      ---
;--  de les entrades de la seguent forma      ---
;--  utilizen't el registre pcl (contador PRG ---
;--  i la ordre addwf pcl,1                    ---
;-----
;--  La tabla es la seguent:                    ---
;--  00 -> prg00 -> Eixida en el display 0    ---
;--  01 -> prg01 -> Eixida 1                  ---
;--  10 -> prg02 -> Eixida 2                  ---
;--  11 -> prg03 -> Eixida 3                  ---
;-----

list    p=16f84

pcl     equ    h'02' ;Contador de programa
org     0

bsf     h'03',5      ;Bank 1
clrf    h'06'        ;Port B eixides
movlw   b'00011111'
movwf   h'05'        ;Port A entrades
bcf     h'03',5      ;Bank 0

;-----
inici   movf    h'05',0      ; Llegim entrades
        andlw   b'00000011' ; Pillem dos ultims bits
        addwf   pcl,1       ; Salt indexat amb PCL h'02'
salt    goto    prg00       ; Inici salt indexat
        goto    prg01       ; Activacio menu
        goto    prg02
        goto    prg03

;-----
prg00   movlw   b'10111111'  ;Eixida 0
        movwf   h'06'        ;Enviar al portB
        goto    inici

prg01   movlw   b'00000110'  ;Eixida 1
        movwf   h'06'
        goto    inici

prg02   movlw   b'11011011'  ;Eixida 2
        movwf   h'06'
        goto    inici

prg03   movlw   b'01001111'  ;Eixida 3
        movwf   h'06'
        goto    inici

```

end

**; FIN DE PROGRAMA**

El siguiente programa utiliza la entrada de un interruptor de forma que podemos desplazar los leds o un display a derechas o izquierdas, dependiendo del valor de entrada.

```
-----  
;--          Programa de salts indexats          ---  
-----  
;--  El programa crea una tabla de dependents ---  
;--  de la entrada A0 de la següent forma ---  
;--  utilitzan't el registre pcl (contador PRG ---  
;--  i la ordre addwf pcl,1          ---  
-----  
;--  La seqüència es:          ---  
;--  0 -> prg00 -> Display gira a dretes ---  
;--  1 -> prg01 -> Display gira esquerres ---  
-----  
  
list  p=16f84  
  
pcl      equ    h'02'  ;Contador de programa  
conta1  equ    h'10'  ;Contadors retard led  
conta2  equ    h'11'  ;--- // ---  
porta   equ    h'05'  
portb   equ    h'06'  
org     0  
  
bsf     h'03',5      ;Bank 1  
clrf   portb        ;Port B eixides  
movlw  b'00011111'  
movwf  porta        ;Port A entrades  
bcf    h'03',5      ;Bank 0  
  
-----  
  
movlw  h'00'        ;Iniciatitzar els contadors  
movwf  conta1      ;per al retard dels leds  
movlw  h'00'        ;--- //  
movwf  conta2      ;--- //  
  
;----- Programa principal -----  
inici  
movf   porta,0     ; Llegim entrades  
andlw  b'00000001' ; Pilleu ultim bit  
addwf  pcl,1       ; Salt indexat amb PCL h'02'  
salt   goto  prg00  ; Inici salt indexat pcl+0  
goto   prg01       ; Activacio menu   pcl+1  
  
;----- Programa 00 -----  
prg00  ;Si el interruptor no esta polsat  
movlw  b'00000001' ;activem desplaçament dreta
```

```

movwf portb
call retard ;Cridem retard leds
movlw b'00000010'
movwf portb
call retard
movlw b'00000100'
movwf portb
call retard
movlw b'00001000'
movwf portb
call retard
movlw b'00010000'
movwf portb
call retard
movlw b'00100000'
movwf portb
call retard
goto inici ;-- Fi del programa 00 -----
;----- Programa 01 -----
prg01 ;Si interruptor activat
movlw b'00100000' ;activem desplaçament esquerra
movwf portb
call retard ;Cridem retards leds
movlw b'00010000'
movwf portb
call retard
movlw b'00001000'
movwf portb
call retard
movlw b'00000100'
movwf portb
call retard
movlw b'00000010'
movwf portb
call retard
movlw b'00000001'
movwf portb
call retard
goto inici ;-- Fi del programa 01 -----
;----- Subrutina retard -----
;----- Retard dels leds -----
retard incfsz conta1,1 ; Retard per als leds
goto retard ; Utilitza 2 contadors
incfsz conta2,1 ; Amb un bucle
goto retard
return
;----- Fi de PROGRAMA -----
end

```

2.1 Práctica: Con el circuito del ejercicio 1.1 grabar el pic con el ejercicio anterior y comprobar su funcionamiento.

2.2 Práctica: Realizar un programa que tenga la siguiente tabla de verdad, grabar en el microcontrolador y comprobar funcionamiento.

Entradas port A (RA0 RA1 RA2)	Salidas port B (RB0 RB1 RB2 RB3 RB4 RB5 RB6 RB7)
000	10101010
001	01010101
010	00001111
011	11110000
100	11001100
101	00110011
110	00011100
111	11100001

2.3 Realiza un programa que tenga una tabla con la siguiente función: RA0 and RA1 and RA2 or RA3

2.4 Realiza el ejercicio 2.1 pero con instrucciones rlf y rrf

### 3 Tablas

Una tabla de datos es una información grabada en el PIC que podemos cargar en el registro W mediante la instrucción retlw k.

Las tablas se usan para guardar datos o funciones digitales.

Existen dos métodos para utilizar las tablas o bien utilizamos el comando retlw k o la directiva DT.

COMANDO	NOTA
RETLW K	Carga en W el valor K y vuelve al programa despues de la llamada
DT valor,valor,...	Directiva define tabla (equivale a retlw k)

### Ejemplo:

```

;-----
;-- Se desea implementar la siguiente tabla de verdad
;
;      I2  I1 I0 / O3 O2 O1 O0
;      -----
;      0  0  0  0  0  1  0
;      0  0  1  1  1  0  1
;      0  1  0  0  1  1  0
;      0  1  1  0  0  0  1
;      1  0  0  1  1  1  1
;      1  0  1  1  1  0  0
;      1  1  0  0  0  1  1
;      1  1  1  0  0  0  0
;
;      Entradas I2 I1 I0 al puerto A
;      Salidas O3 O2 O1 O0 al puerto B
;-- Utilizamos el comando retlw
;-----

```

```

list p=16f84
org 0
;----- Configuración puertos -----
bsf h'03',5
clrf h'06'
movlw b'00011111'
movwf h'05'
bcf h'03',5
;----- Inicio programa -----
inici movf h'05',0
andlw b'00000111' ;Sólo 3 últimos bits
call tabla ;llamamos subrutina tabla
movwf h'06' ;enviamos w al port B
goto inici ;volvemos a empezar
;----- Tabla -----
tabla addwf H'02',1 ;Sumamos w y PCL (contador programa)
RETLW B'00000010' ;si porta = 000 W=00000010 y volvemos
con return
RETLW B'00001101' ;si porta = 001 W=00001101 y volvemos
RETLW B'00000110'
RETLW B'00000001'
RETLW B'00001111'
RETLW B'00001100'
RETLW B'00000011'
RETLW B'00000000' ;si porta = 1111 W=00000000
;-----
END

```

En el siguiente ejemplo se utiliza la directiva de compilación DT (Define tabla) para leer un puerto de entrada portA y sacar por el portB mediante un display el valor leído en binario.

```

;-----
;-- Utilización tablas mediante directiva DT --
;-- La sintaxis es DT valor,valor,valor, .....
;-- cada valor equivale a un literal cargado
;-- en w, es equivalente a la instrucción
;-- retlw valor
;-- El siguiente programa lee las entradas y
;-- saca el valor leído en binario mediante un
;-- display 7 segmentos
;-----
list p=16f84
org 0
;-----Configuración puertos -----
bsf h'03',5 ;bank 1
clrf h'06' ;portb=00000000
movlw b'00011111' ;configurar port A
movwf h'05' ;como entradas
bcf h'03',5 ;bank 0
;-----
inici movf h'05',0 ;Leer port A

```

```

andlw b'00000111' ;tomamos sólo los 3 últimos bits
call tabla ;Llamada subrutina tabla
movwf h'06' ;Enviar w a port B
goto inici ;Bucle

```

```

;-----
;-- Subrutina tabla
;-- Los datos corresponden a los valores hex para el
;-- display 7 segmentos
;-- los datos se cargan en w
;-----

```

```

tabla addwf h'02',1 ;Sumamos el valor del portA al PCL
dt h'3f',h'06',h'5b',h'4f',h'66',h'6d',h'7d',h'07',h'7f'
return

```

```

;-----
;-- Según el valor sumado del port A saltará a un valor definido en
;-- la tabla DT,de forma si es 0->3f si es 1->06 si es 2->5b etc
;-- y devuelve el valor en w
;-----

```

```

end

```

Ej 3.1 Realizar mediante tablas la función OR de 4 entradas.

Ej 3.2 Crear una tabla con el abecedario disponible en un display y que se pueda elegir la letra según la entrada por el port A.

Ej 3.3 Realizar un programa que muestre por el display 'alcoi' letra a letra, utiliza un contador y un retardo de 1 segundo

## DIRECTIVAS

Una directiva del ensamblador no es una instrucción por sí misma, pero muchas veces facilita la labor de programación.

A continuación se señalan algunas directivas y funciones específicas a la hora de programar el microcontrolador.

Para más información se puede consultar (en inglés) [la guía de usuario del ensamblador MPASM](#)

Directivas	Significado
DT	Define Tabla Ej DT b'00001111',b'00110011',b'00001110'...
END	Fin de programa
EQU	Definición de constante Ej porta EQU h'05'

INCLUDE	Permite incluir ficheros con definiciones o subrutinas Ej INCLUDE <P16F84A.INC>
ORG	Origen de los datos del programa
#DEFINE	Define una etiqueta que sustituye a un texto Ej #DEFINE entrada porta,1
LIST	Listado de opciones del ensamblador Ej LIST p=16f84

WATCHDOG (WDT)(Literalmente Perro Guardián) se encarga de vigilar el correcto funcionamiento del sistema, de forma que revisa cada x tiempo que todo funcione normalmente, si no es así provoca un reset. es una de las opciones de la directiva \_CONFIG

Otra opción interesante a la hora de grabar el microcontrolador es habilitar PWRTE de forma que impedimos la lectura de los datos en el microcontrolador y protegemos datos como software.

---



---