Introducción a la programación del PIC 16F84

INTRODUCCIÓN A LA PROGRAMACIÓN
Microcontrolador PIC 16F84

mail : enric.serra

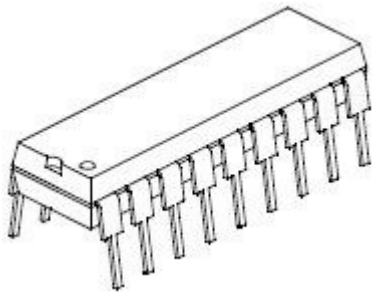
0 - INTRODUCCIÓN.

Este documento es una introducción a la programación del PIC 16f84.

Todos los comentarios y bugs serán bien recibidos.

1 – MICROCONTROLADORES.

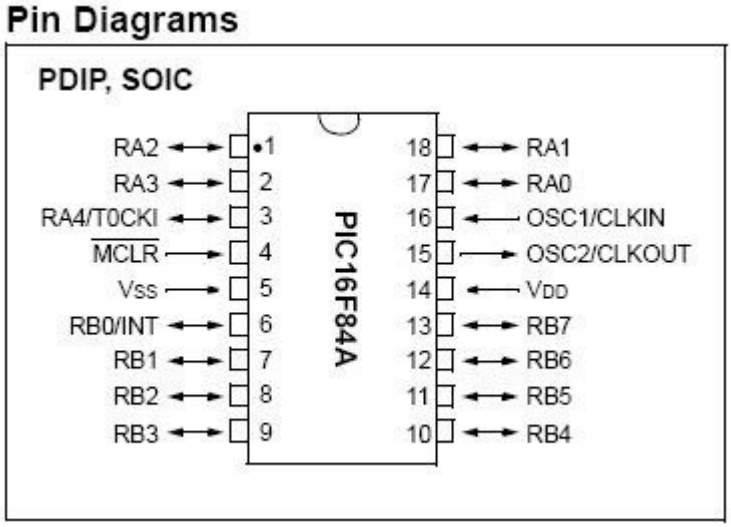
Un microcontrolador, es un circuito integrado programable que contiene los elementos necesarios para controlar un sistema.
PIC significa *Peripheral Interface Controler* es decir un controlador de periféricos.
Cuando hablamos de un circuito integrado programable que controla periféricos, estamos hablando de un sistema que contiene entre otras cosas una unidad arimético-lógica, unas memorias de datos y programas, unos puertos de entrada y salida, es decir estamos hablando de un pequeño ordenador diseñado para realizar unas funciones específicas.
Podemos encontrar microcontroladores en lavadoras, teclados, teléfonos móviles, ratones etc.
Hay multitud de microcontroladores con más memoria, entradas y salidas, frecuencia de trabajo, coste, subsistemas integrados y un largo etc dependiendo de cada tipo de microcontrolador. El presente documento esta basado en el popular microcontrolador PIC 16F84 del fabricante *Microchip Technology Inc* ya que es un sistema sencillo, barato y potente para muchas aplicaciones electrónicas.
Las características del controlador las podemos encontrar en la web del fabricante microchip o en el siguiente enlace pic16f84.pdf

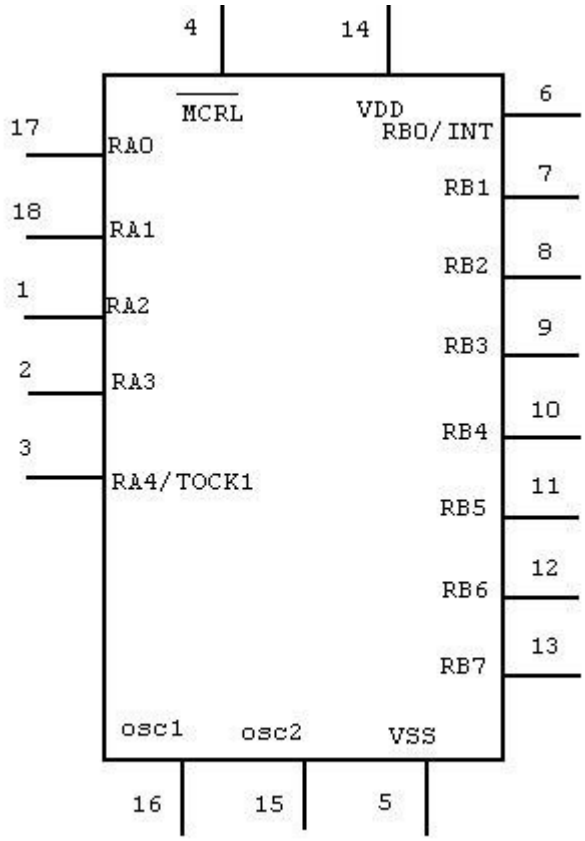


Pic 16f84

Las patillas del PIC 16F84, 18 en el modelo A son las siguientes:

Encapsulado y simbología.



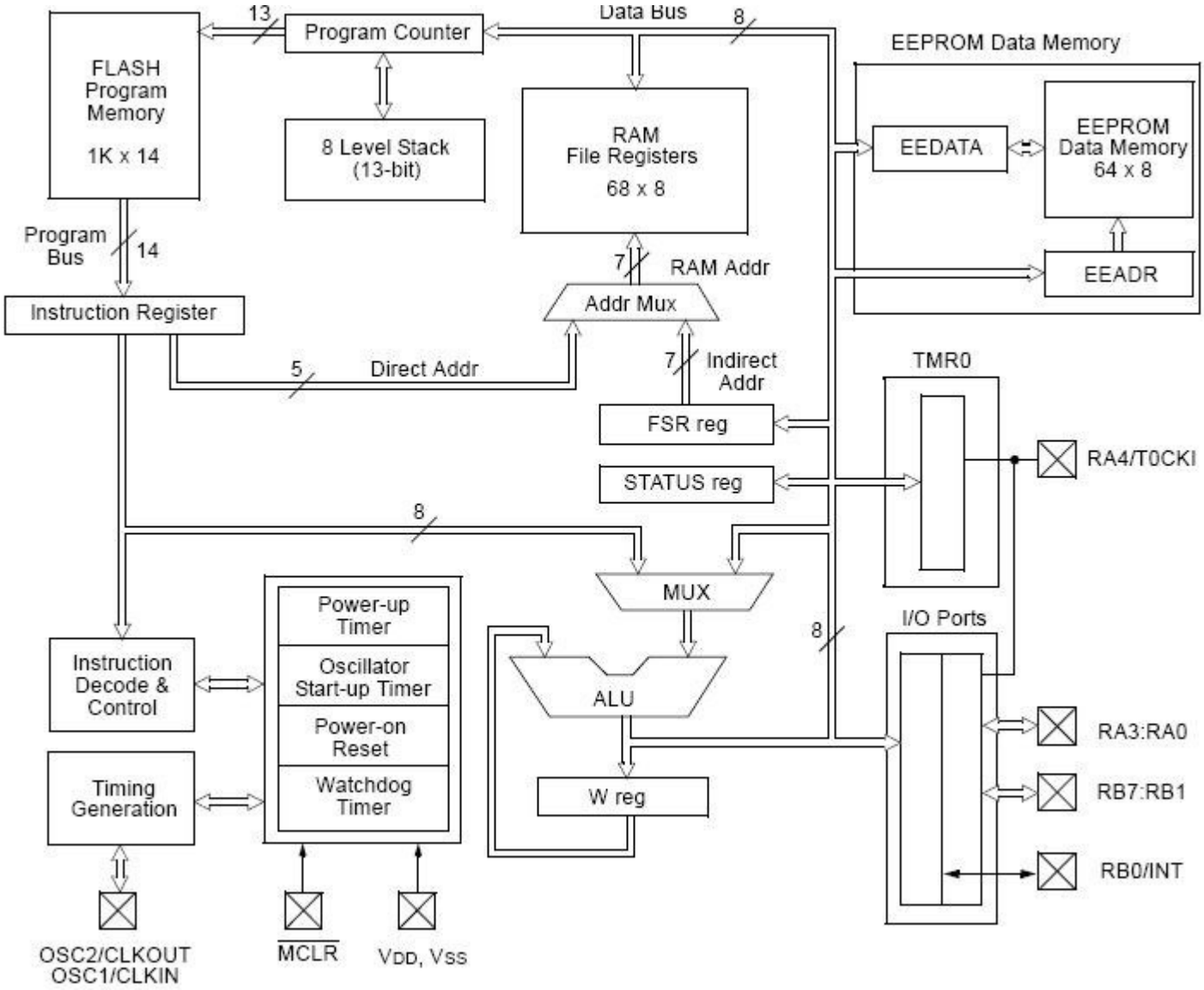


PIC 16F84

Web de microchip

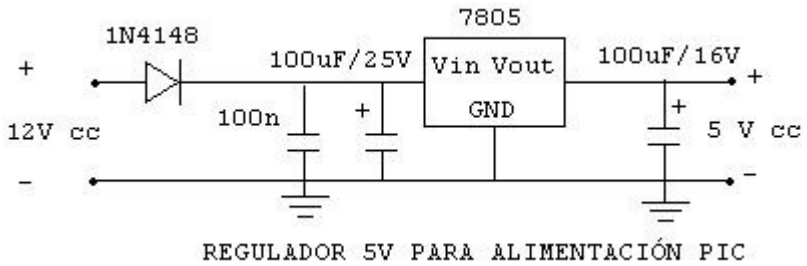
2.- CARACTERISTICAS

La estructura del microcontrolador



Alimentación :
El PIC se alimenta a 5 V entre los puntos Vdd (+) y Vss (-). El consumo del circuito depende de las cargas en los puertos y de la frecuencia de trabajo.
Práctica 1.1: Diseñar y montar el siguiente regulador de V

Regulador 5 V para alimentar un PIC 16F84



REGULADOR 5V PARA ALIMENTACIÓN PIC

Frecuencia de trabajo:
Los PIC's necesitan un reloj oscilador que marcará la frecuencia de trabajo.

Estos osciladores pueden ser del tipo :

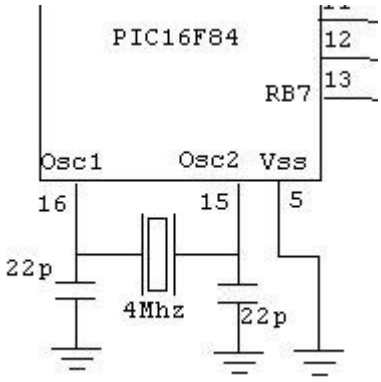
RC Formado por una resistencia y un condensador

HS seutiliza un cristal de cuarzo o resonador cerámico (Hasta 10 Mz)

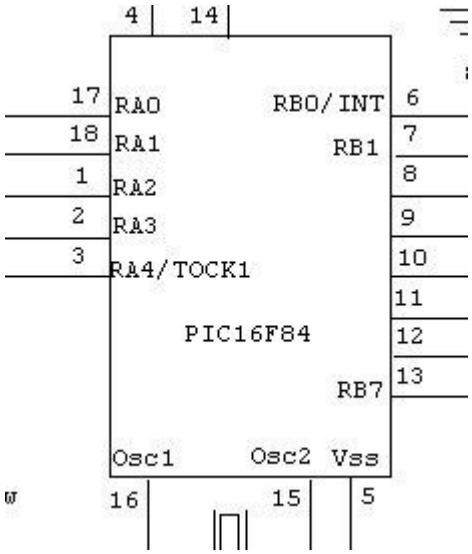
XT Cristal o resonador hasta 4 Mhz

LP Bajo consumo (hasta 200Khz)

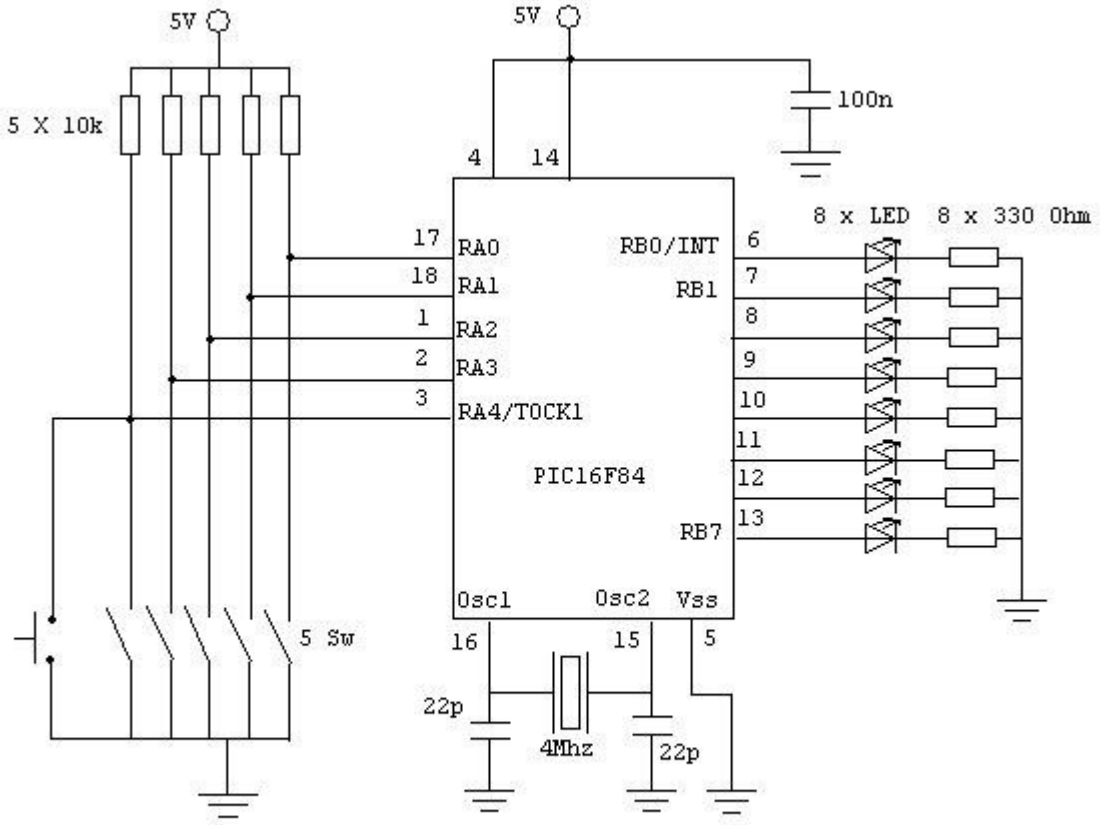
Los osciladores se colocan entre las patillas OSC1 y OSC2



Puertos de ENTRADA/SALIDA
Los puertos son entradas y salidas del microcontrolador al exterior, por ellas enviarmos o introducimos señales digitales TTL (5V) de forma que podemos comunicar el microcontrolador con el exterior.
En este caso tenemos 2 puertos de entrada y salida E/S. Sus nombres son RA y RB.
El puerto RA tiene 5 pins RA0-RA4, un caso particular es RA4/TOCK1 que puede actuar como pin de entrada o como entrada de impulsos para un contador denominado TMRO
El puerto B tien 8 líneas que van desde RB-RB7 .Cada línea del RA o del RB se puede configurar como entrada o salida mediante 2 registros llamados TRISA y TRISB.



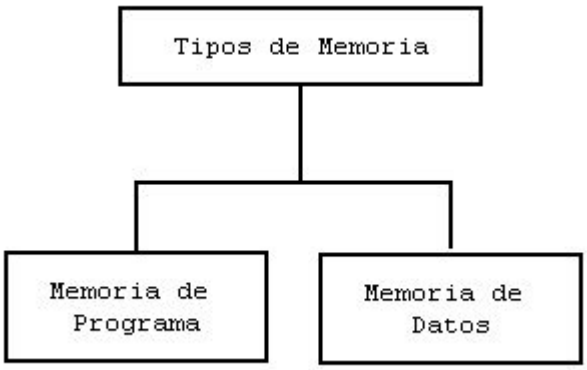
Con esta información podemos montar un sencillo entrenador para PIC's 16F84 con el puerto RA como entrada y el puerto RB como salida. Utilizaremos como entrada unos microinterruptores y como salida unos leds conectados al puerto RB0-RB7.
1.2 Práctica: Diseñar el PCB y montar el siguiente entrenador PIC



ENTRENADOR PIC 16F84

Memorias.

Todo dispositivo programable necesita de una memoria para poder almacenar el programa, poder manejar variables y almacenar datos.



MEMORIA DE PROGRAMA

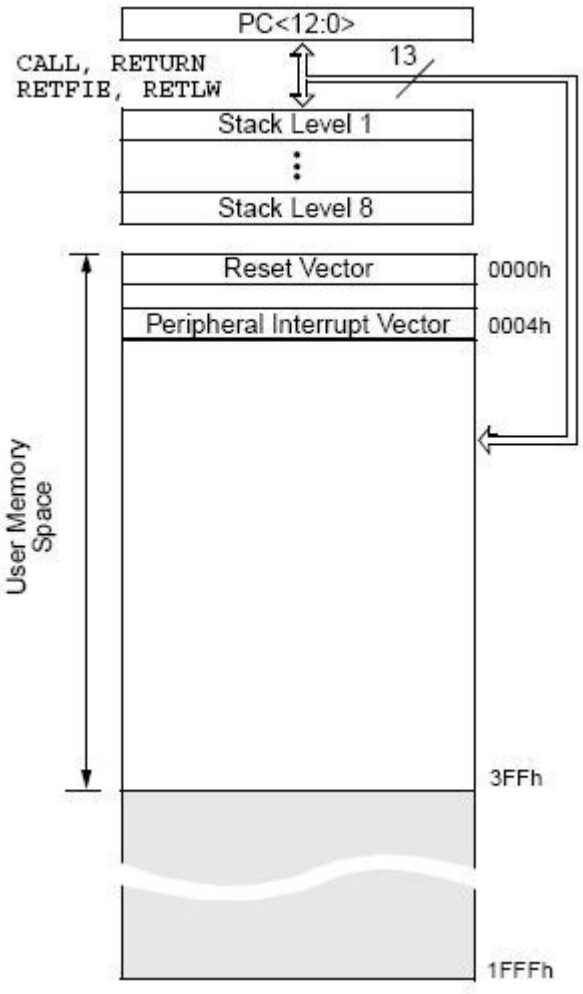
En el PIC 16F84 la memoria de programa o de instruciones es una memoria tipo FLASH de 1K

En esta memoria almacenaremos el programa que ejecutará el microcontrolador

Existe un registro especial llamado **contador de programa** PC cuya finalidad es avanzar por las instrucciones del programa de forma secuencial excepto cuando se encuentran instrucciones de salto.

Mapa de memoria de programa

MEMORIA DE DATOS



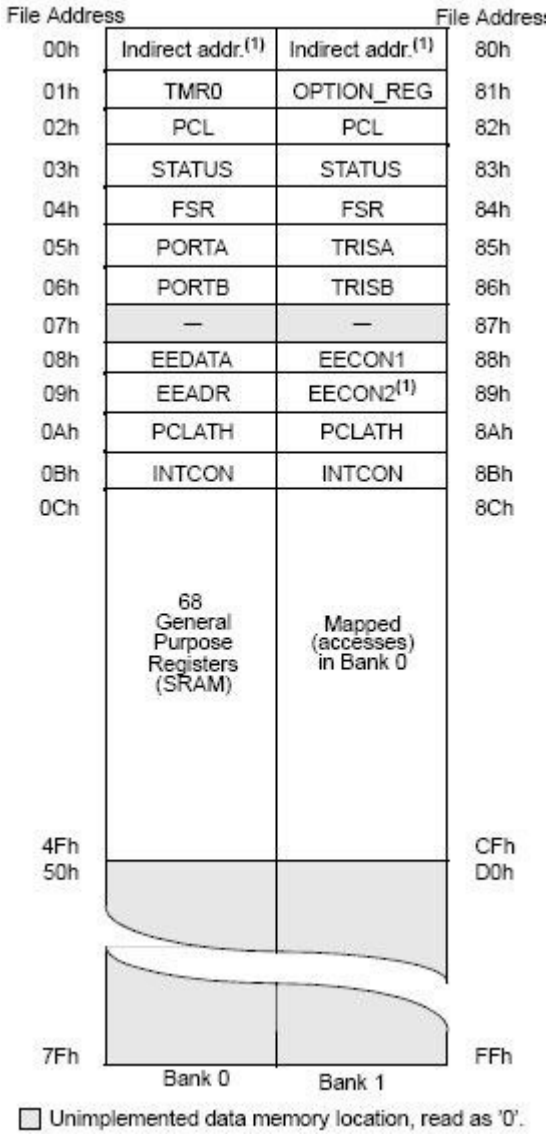
La memoria de datos sirve para almacenar variables, leer puertos de entrada o escribir en los puertos de salida, podemos tambien acceder al temporizador o al registron EEPROM

La memoria de **datos** en el PIC 16F84 está formada por dos zonas:

MEMORIA RAM de 68 registros

MEMORIA EEPROM de 68 registros cuya caracteristica principal es que no se perderán los datos cuando se desconecta la alimentación.

Mapa de memoria de datos



Esta memoria está formada por dos bancos Bank0 y Bank1

Mapa de memoria RAM detallada

TABLE 2-1: SPECIAL FUNCTION REGISTER FILE SUMMARY

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

Otras caracteristicas

Dentro del microcontrolador hay unos registros especiales que determinan algunas de las características notables del microcontrolador:

- Temporizador/Contador TMR0

- Perro guardián watch Dog (WD)

- Interrupciones.

- Reset .(Reinicio del sistema)

Estas características se explicarán mas adelante en el apartado correspondiente a la programación.

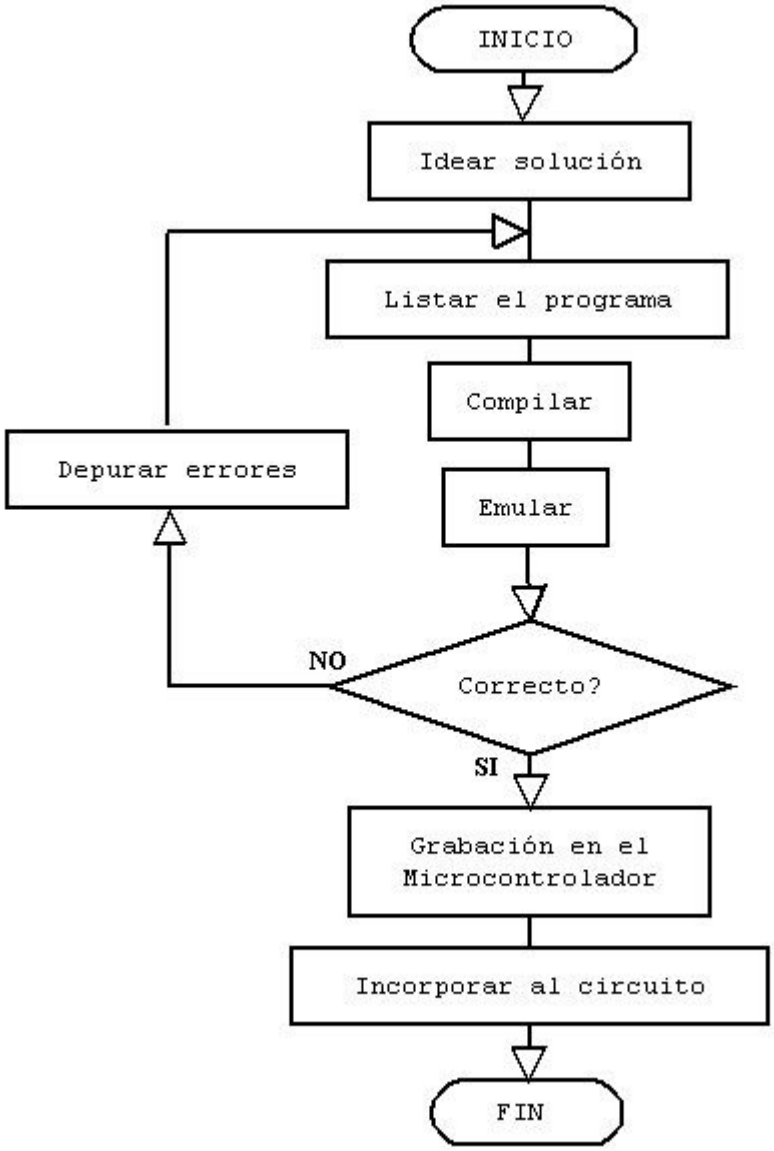
---

3 - DISEÑO DE PROYECTOS

Frente a un problema técnico, hay que buscar una solución de forma barata y sencilla, en este proceso de búsqueda de soluciones, los microcontroladores PIC pueden ayudarnos a realizar soluciones sencillas, rápidas y baratas.

Partiremos de un planteamiento teniendo presente todas las variables que afectan al sistema,desarrollaremos la idea y la implementaremos con las herramientas adecuadas.

Diagrama de flujo del desarrollo de proyectos con microcontrolador



Herramientas y ficheros generados en cada paso del desarrollo de un proyecto mediante PIC's

Pasos a seguir	Herramientas Windows	Herramientas Linux	Ficheros
Idear solución	Pensar la solución y diseñar circuito con programas PCB	PCB	
Listar	Listar programa con EDIT,MPLAB o con cualquier editor ASCII	Editores ASCII	Listado.asm
Compilar	MPASM	gpasm	Listado.hex

Emular	MPSIM , SimuPIC, Misim	gputils,gpsim,pikDEV,Misim	
Grabar	WinPICme-TR + circuito grabador, IC-Prog	pikDEV	

Enlaces

Microchip

Misim

Gpasm

---

4 PROGRAMACIÓN EN ENSAMBLADOR PIC 16F84

Para programar un PIC 16F84 necesitamos conocer las instrucciones para generar el código fuente para posteriormente compilarlo por ejemplo con MPASM, emular el programa y poder grabarlo para implementarlo en el circuito correspondiente.

El listado de instrucciones de microchip (el fabricante del microcontrolador) son las siguientes:

TABLE 7-2: PIC16CXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected
BYTE-ORIENTED FILE REGISTER OPERATIONS								
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00	1011	dfff	ffff	
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z
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	Z
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z
MOVWF	f	Move W to f	1	00	0000	lfff	ffff	
NOP	-	No Operation	1	00	0000	0xx0	0000	
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z
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	Z

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	$\overline{TO}$,$\overline{PD}$
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 val on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by a 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 secon executed as a NOP.

Un listado en código fuente es un fichero ASCII con extensión ASM que está formado por 4 columnas para identificar las diferentes funciones:
- Etiquetas: Dan nombre a determinadas partes del programa (hasta 32 caracteres)
- Instrucciones: Son las instrucciones que se pasan al microcontrolador o una DIRECTIVA al ensamblador.
- Datos: Datos u operandos para las instrucciones.
    Registros (f)
    Bits
    Etiquetas
    Número constante literal (L)
- Comentarios Siempre después de ; son descripciones para hacer más legible el listado.

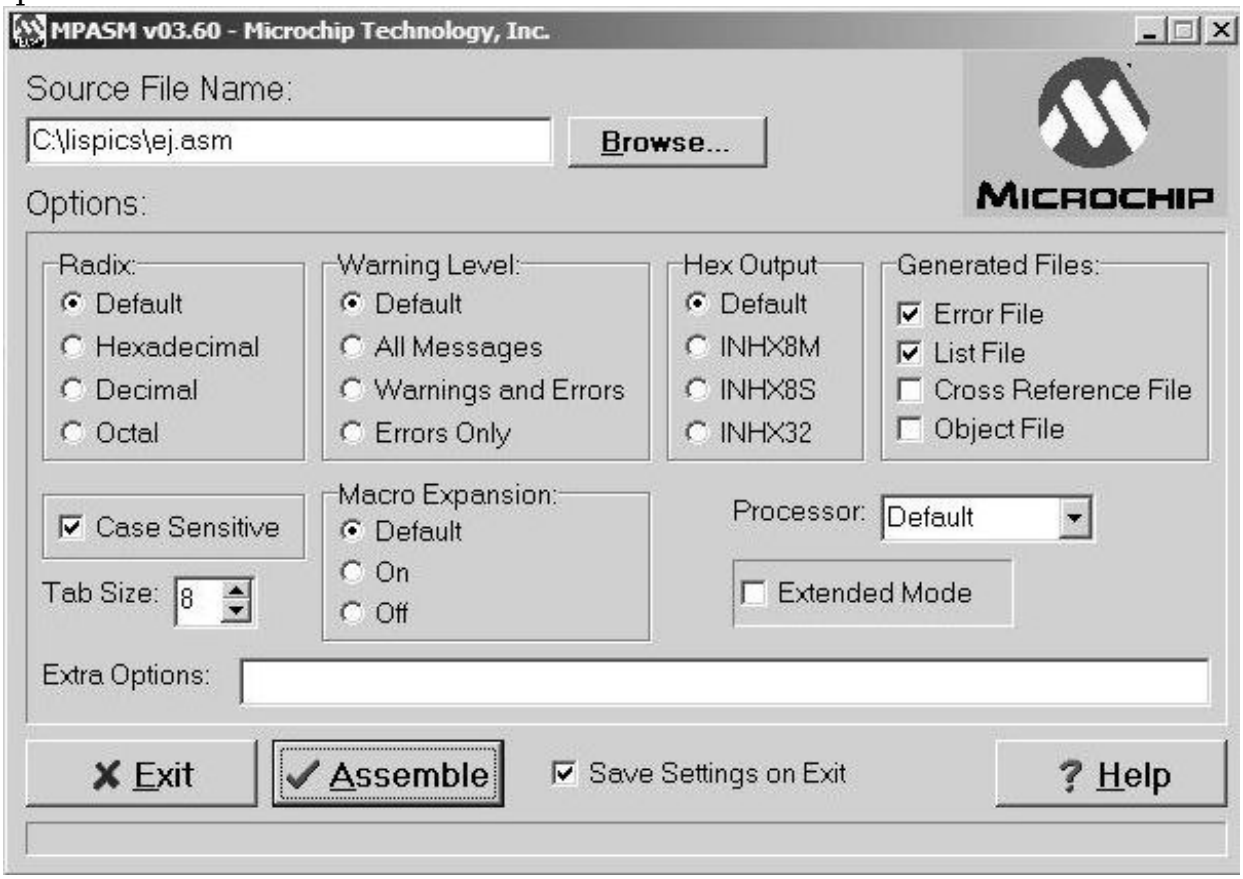
EJEMPLO DE PROGRAMA

Realizamos en un editor el siguiente listado de programa que llamaremos EJ1.ASM

```
; -------------------------------------------------------

; EL SIGUIENTE PROGRAMA SUMA 05 Y 02

; -------------------------------------------------------

LIST P=16F84 ; Modelo de PIC

SUMA1 EQU 0X0C ; DIRECCION HEXADECIMAL DEL OPERANDO 1
SUMA2 EQU 0X0D ; DIRECCION DEL SEGUNDO OPERANDO
```

```
RESUL EQU 0X0E ; DIRECCION DEL RESULTADO

; -------------------------------------------------
; Se han definido 3 variables SUMA,SUMA2 y RESUL
; en diferentes posiciones de la memoria de datos
; en el REGISTER FILE MAP
; -------------------------------------------------

ORG 0

; -------------------------------------------------
; Origen de las instrucciones en el PROGRAM MEMORY MAP
; Directiva de compilación
; -------------------------------------------------

movlw 0X05 ; PONEMOS 05 en hexadecimal EN W
movwf SUMA1 ; PASAMOS W A LA MEMORIA SUMA1
movlw 0X02 ; PONEMOS 02 Hex EN W
movwf SUMA2 ; PASAMOS W A LA MEMORIA SUMA2
movfw SUMA1 ; PONEMOS EN W SUMA1 = 05
addwf SUMA2 , 0 ; SUMAMOS W+SUMA2 Y EL RESULTADO
EN W
movwf RESUL ; PONEMOS VALOR ACUMULADOR EN RESUL

END ; FIN DE PROGRAMA

;-------------------------------------------------
```

Compilaremos con el MPASM



El compilador generará los siguientes ficheros EJ1.HEX,EJ1.COD,EJ1.ERR,EJ1.LST
Con el fichero EJ1.HEX podremos grabar el microcontrolador con el programa WIN-PICME o cualquier otro y un grabador de pics.

4.1 COMANDOS BÁSICOS (I) CARGA DE VALORES,SUMA Y BORRADO DE REGISTROS

INSTRUCCIÓN	EJEMPLO	NOTA
movwf f	movwf SUMA	mover acumulador w al registro f
movlw k	movlw 0x05	mover un literal l (0x05) al acumulador w
addwf f,d	addwf SUMA2 ,0	suma w y f (SUMA2) si d=0 el resulatado en W si d=1 el resultado en f (SUMA2)
addlw k	addlw 0x02	suma literal (0x02) y w
clrf f	clrf SUMA	pone a 0 el registro f (SUMA)
clrw	clrw	pone a 0 el acumulador w

EJEMPLO 2

Sumar en binario 0001 y 0010. Guardar el resultado en 0x0E

```
; --------------------------------------------------
; EJ2.ASM PROGRAMA PARA SUMAR EN BINARIO 0001 Y 0010
; RESULTADO GUARDADO EN 0X0E
; --------------------------------------------------
LIST P=16F84 ; Modelo de PIC
SUMA1 EQU 0X0C ; DIRECCION HEXADECIMAL DEL OPERANDO 1
RESUL EQU 0X0E ; DIRRECCION DEL RESULTADO

; --------------------------------------------------
; Se han definido 2 variables SUMA y RESUL
; en diferentes posiciones de la memoria de datos
; en el REGISTER FILE MAP
; --------------------------------------------------

ORG 0

; --------------------------------------------------
; Origen de las instrucciones en el PROGRAM MEMORY MAP
; Directiva de compilación
; --------------------------------------------------

movlw b'0001' ; PONEMOS 0001 EN W
movwf SUMA1 ; PASAMOS W A LA MEMORIA SUMA1
movlw b'0010' ; PONEMOS 0010 EN W
addwf SUMA1 , 0 ; SUMA W A LA MEMORIA SUMA1
RESULTADO EN W
```

```
movwf RESUL ; MOVER W A RESUL

END ; FIN DE PROGRAMA
```

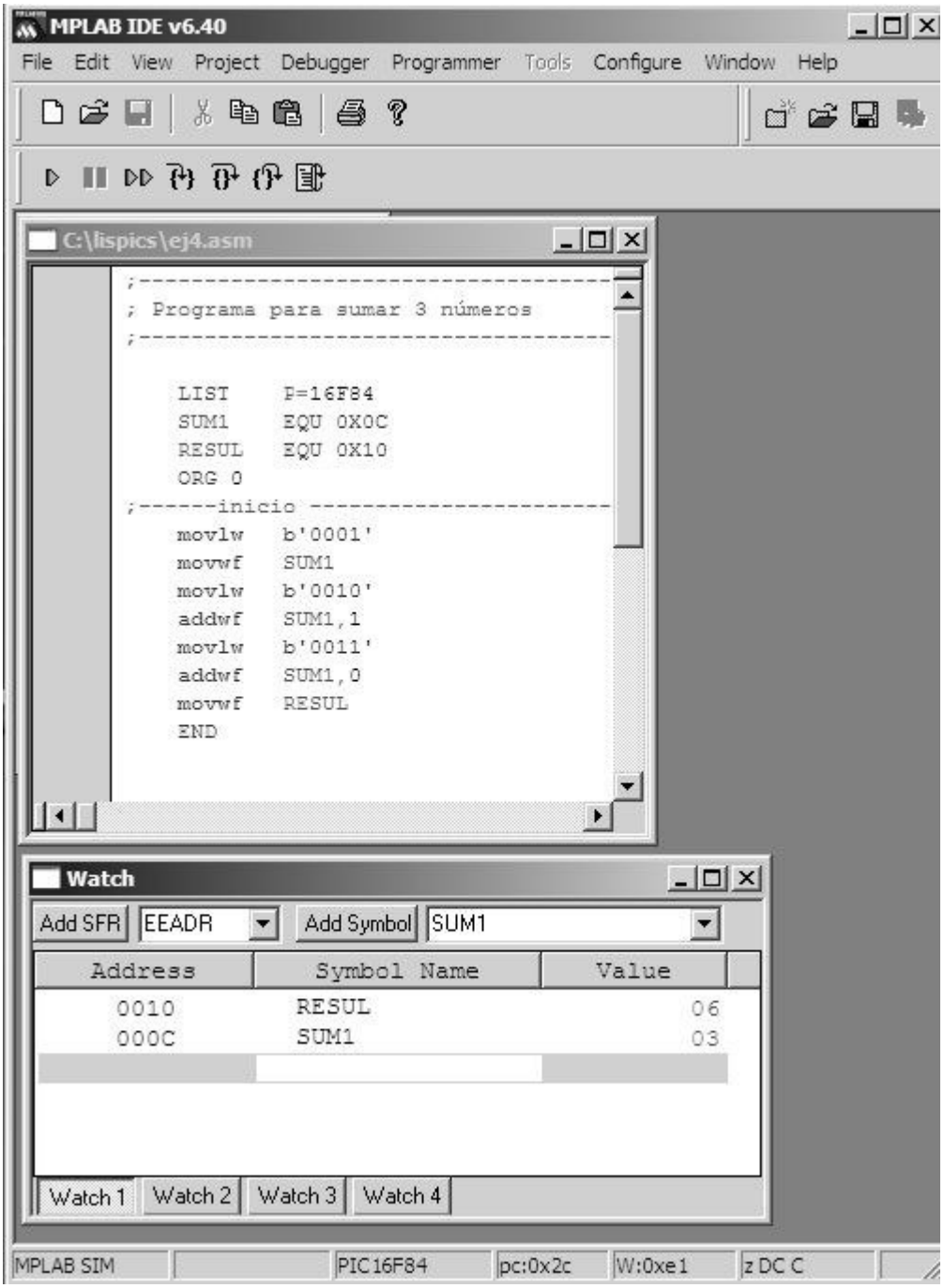
EJEMPLO 3

Sumar en binario 2 y 3, guardar el resultado en 0x0C

```
; --------------------------------------------------
; EJ3.ASM PROGRAMA PARA SUMAR EN BINARIO 2 Y 3
; RESULTADO GUARDADO EN 0X0D
; --------------------------------------------------
LIST P=16F84 ; Modelo de PIC

SUMA1 EQU 0X0C ; DIRECCION HEXADECIMAL DEL
OPERANDO 1
RESUL EQU 0X0D ; DIRRECCION DEL RESULTADO

; --------------------------------------------------
; Se han definido 2 variables SUMA1 y RESUL
; en diferentes posiciones de la memoria de datos
; en el REGISTER FILE MAP
; --------------------------------------------------

ORG 0

; --------------------------------------------------
; Origen de las instrucciones en el PROGRAM MEMORY MAP
; Directiva de compilación
; --------------------------------------------------

movlw b'0010' ; PONEMOS UN 2 EN W
movwf SUMA1 ; PASAMOS W A LA MEMORIA SUMA1
movlw b'0011' ; PONEMOS 3 EN W
addwf SUMA1 , 0 ; SUMA W A LA MEMORIA SUMA1
RESULTADO EN W
movwf RESUL ; MOVER W A RESUL

END ; FIN DE PROGRAMA

;--------------------------------------------------
```

EJEMPLO 3

Sumar tres números (1, 2 y 3)



Listado y funcionamiento en el entorno de simulación MPLAB IDE de microchip.

En la ventana Watch podemos ver el valor de SUM1 y RESUL después de arrancar el programa.

EJEMPLO 4

Suma de los números 3 y 5 en decimal y almacenamos el resultado en 0x12.

El compilador acepta sistemas de numeración hexadecimal, decimal, octal, binario y ASCII.

TIPO	SINTAXIS
Decimal	d'cantidad'

	.cantidad
Hexadecimal	h'cantidad' 0xcantidad cantidad h
Octal	o'cantidad'
Binario	b'cantidad'
ASCII	a'caracter' 'caracter

Listado

```
; ----------------------------------------
; SUMA EN DECIMAL 3 Y 5 RESULTADO EN 0X12
; ----------------------------------------

      LIST P=16f84
      SUM EQU 0x10 ; Memoria intermedia
      RESUL EQU 0x12 ; Memoria de resultado

      ORG 0
      ;-----------****** INICIO *****----------------
      movlw d'3' ; Cargamos en acumulador 3
      movwf SUM ; Almacenamos el 3 en SUM 0x10
      movlw d'5' ; Cargamos 5 en el acumulador
      addwf SUM,0 ; Sumamos y guardamos en w
      movwf RESUL ; Guardamos el resultado en 0x12
      END
      ;-----------------------------------------------
```

Ejercicios propuestos

4.1.1- Sumar cuatro números en binario (0,1,2,3)

4.1.2- Sumar tres números y almacenar los números y el resultado en las direcciones 0x0c 0x0d 0x0e 0x0f

4.1.3 - Sumar 2 números, almacenar el resultado en 0x0d, borrar el registro SUM y borrar el acumulador W

4.2 COMANDOS (II) INCREMENTO Y DECREMENTO .RESTAS

INSTRUCCIÓN	EJEMPLO	NOTAS
incf f,d	incf	Incrementa el registro f, si d=0 se

	SUMA,0	**almacena en w, si d=1 se almacena en el registro**
decf f,d	**decf SUMA,1**	**Decrementa el registro SUMA y el resultado de guarda en SUMA**
sublw k	**sublw b'0110'**	**Resta 0110 - w y resultado almacenado en w**
subwf f,d	**subwf RESTA,0**	**Resta f-w y resultado en w**

EJEMPLO 5 Cargar el número 5 en 1Ah (VALOR1) y incrementar dos veces (Almacenar en 1ch)

```
;-----------------------------------
; Programa que incrementa el valor 5
; -----------------------------------
LIST P=16F84
VALOR EQU 1aH; Almacenamos valor en 1aH
ORG 0
;-----------------------------------
; INICIO
;-----------------------------------
INICIO

        movlw h'5'; Cargamos 5 en w
        movwf VALOR; movemos 5 a VALOR
        incf VALOR,1; Incrementamos VALOR
        incf VALOR,1


END
```

EJEMPLO 6 DECREMENTAR UN VALOR

```
; Decrementador de registro
; Cargamos el valor 5 en 0x0a
; Decrementamos 2 veces hasta 3

        LIST p=16F84
        v equ 0x0a
        org 0

inici

        movlw 0x05
        movwf v
        decf v,1
        decf v,1
        movfw v
```

```
end
```

Ejercicios propuestos

4.2.1 Restar dos números binarios y almacenar el resultado en la dirección 0x10

4.2.2 Decrementar el valor 10 cinco veces y almacenarlo en la dirección 0x2b

---

4.3 INSTRUCCIONES LÓGICAS

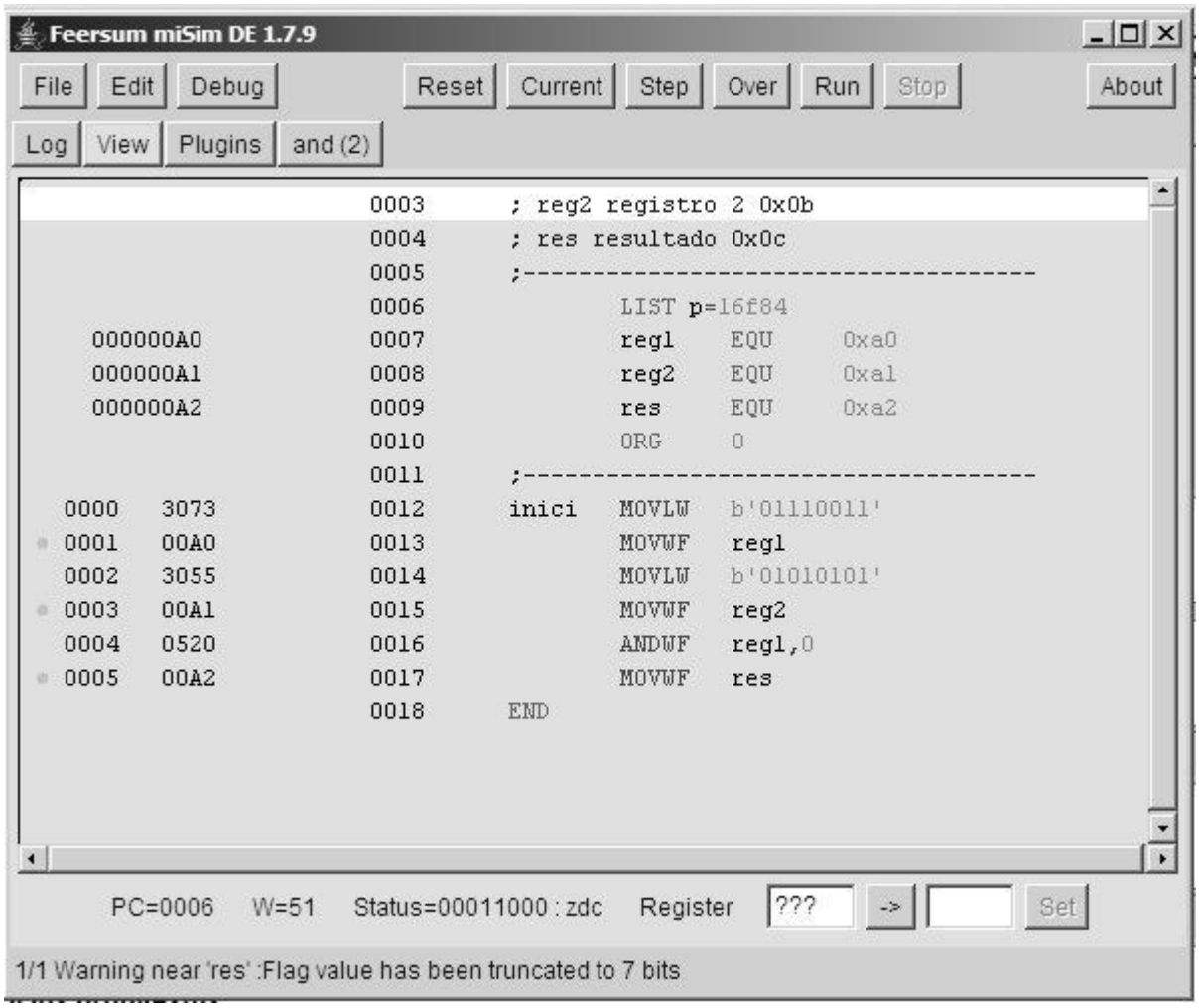
INSTRUCCIÓN	EJEMPLO	NOTAS
andlw k	**andlw b'ooooo111'**	**AND del literal b'00000111'con w**
andwf f,d	**andwf reg,1**	**AND del w y el registro reg**
comf f,d	**comf reg,0**	**COMPLEMENTO del registro reg (cambiar 0 por 1)**
iorlw k	**iorlw valor**	**OR de w con valor**
iorwf f,d	**iorwf reg,1**	**OR de w con reg**
rlf f,d	**rlf reg,0**	**ROTATE LEFT (rota izquierda registro reg) resultado en w**
rrf f,d	**rrf reg,1**	**ROTATE RIGHT(rota derecha registro reg) resultado en reg**
swapf f,d	**swapf reg,0**	**intercambiar 4 bits de + peso con los 4 de - peso del registro reg**
xorlw k	**xorlw b'00001110'**	**OR EXCLUSIVA del literal b'00001110' con w**
xorwf f,d	**xorwf reg,0**	**OR EXCLUSIVA del registro reg con w resultado en w**

EJEMPLO7 Realiza una operación AND entre dos registros

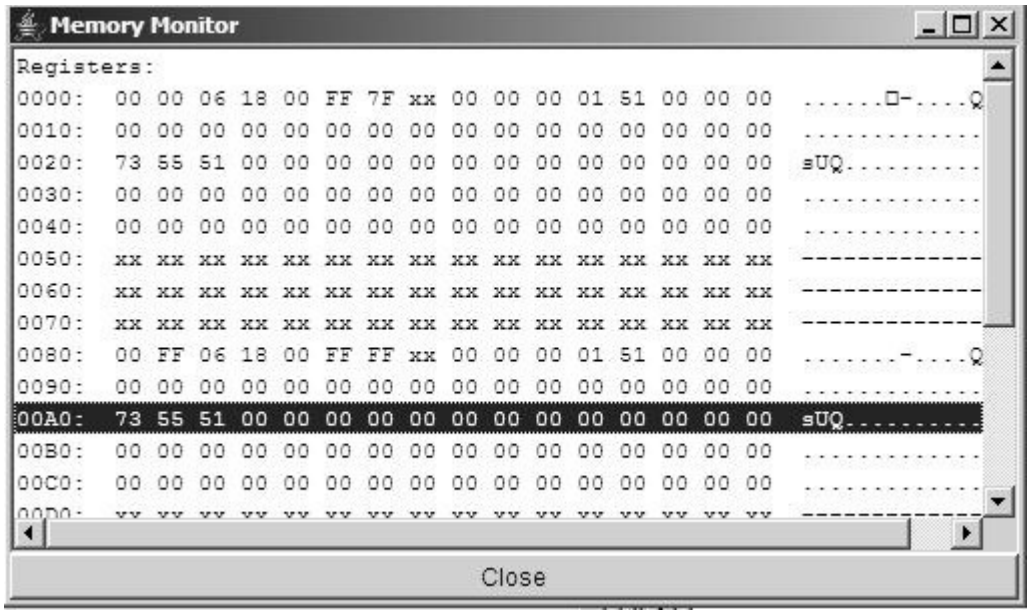
```
; realizar and entre dos registros
; reg1 registro 1 0xa0
; reg2 registro 2 0xa1
; res resultado 0xa2
```

```
;-----------------------------------
list p=16f84
reg1 equ 0xa0
reg2 equ 0xa1
res equ 0xa2
org 0
;-----------------------------------
inici

        movlw b'01110011' ; cargamos 73H en w 73H=0111
        0011 (7 3)
        movwf reg1 ;enviamos a reg1 reg1->01110011
        movlw b'01010101' ;cargamos en w 55H en w 55H=
        0101 0101 (5 5)
        movwf reg2 ;movemos 01010101 a reg 2
        andwf reg1,0 ;realizamos AND entre w y reg1 y lo
        almacenamos en w
        movwf res ; movemos el resultado al registro de
        resultados res
        end

;-------------------------------------------------------------------------------

; El resultado AND de 0111 0011 y 0101 0101 es 0101 0001
(51H)

;-------------------------------------------------------------------------------
```

Podemos ver la simulación y el resultado en el entorno misim (java)



Pantalla de misim después de ensamblar



Memory monitor del misim que muestra los resultados 73,55 y 51

EJEMPLO 8 Realizar operación OR entre un registro y un literal

```
;---------------------------------
; OR ENTRE UN REGISTRO Y UN LITERAL
;---------------------------------
```

```
; REG1 0XAO REGISTRO1
; RES 0XA1 RESULTADO
;--------------------------------
list p=16f84
reg1 equ 0xa0
res equ 0xa1
org 0
;----- inicio program --------------
inici

      movlw b'00001100'
      movwf reg1
      iorlw b'00110000'
      movwf res


end
```

EJEMPLO 9 Rotar dos veces hacia la izquierda un registro

```
; ---------------------------------------------
; rotar el valor 00111111 2 veces a la izquierda
;---------------------------------------------

list p=16f84
reg equ 0x0b
org 0
;---------------------------------------------
inici

      movlw b'00000110'
      movwf reg
      rlf reg,1
      rlf reg,1
      end
```

Ejercicios propuestos

4.3.1 Realizar una rotación 4 veces hacia la derecha de un registro de valor '10110000'

4.3.2 Realizar una OR Exclusiva de val1='00110101' y val2='10100110'

4.3.3 Intercambiar los 4 bits menos significativos por los cuatro más significativos del valor b'00111010', almacenarlos en un registro

---

4.4 INSTRUCCIONES DE BIT

INSTRUCCIÓN	EJEMPLO	NOTAS

bcf f,b	**bcf registro,1**	**Pone a 0 el bit 1 del registro**
bsf f,b	**bsf registro,3**	**Pone a 1 el bit 3 del registro**

4.5 INSTRUCCIÓN GOTO

Es un salto incondicional utilizado para realizar bucles, la istrucción es **GOTO K** donde K es la etiqueta donde queremos saltar.

Ejemplo:

```
INICIO
    movwl b'10001100'
    goto INICIO
END
```

El programa anterior realizará un bucle cargando indefinidamente el valor 10001100 en el registro de trabajo W

4.6 SUBRUTINAS

Podemos realizar bloques de instrucciones que podemos llamar dentro del programa siempre que queramos, estos bloques se denominan subrutinas y la forma de llamarlas es como sigue:

Programa con subrutinas	Equivalente sin subrutinas
INST CALL SUB1 ; Llamada a subrutina SUB1 INST ;Lugar de retorno INST CALL SUB1 INST ;Lugar de retorno SUB1 INST1 ; Bloque se subrutina INST2 INST3 RETURN ;Retorno de la llamada CALL	INST INST1;Subrutina INST2;Subrutina INST3;Subrutina INST INST INST1;Subrutina INST2;Subrutina INST3;Subrutina INST END

END	

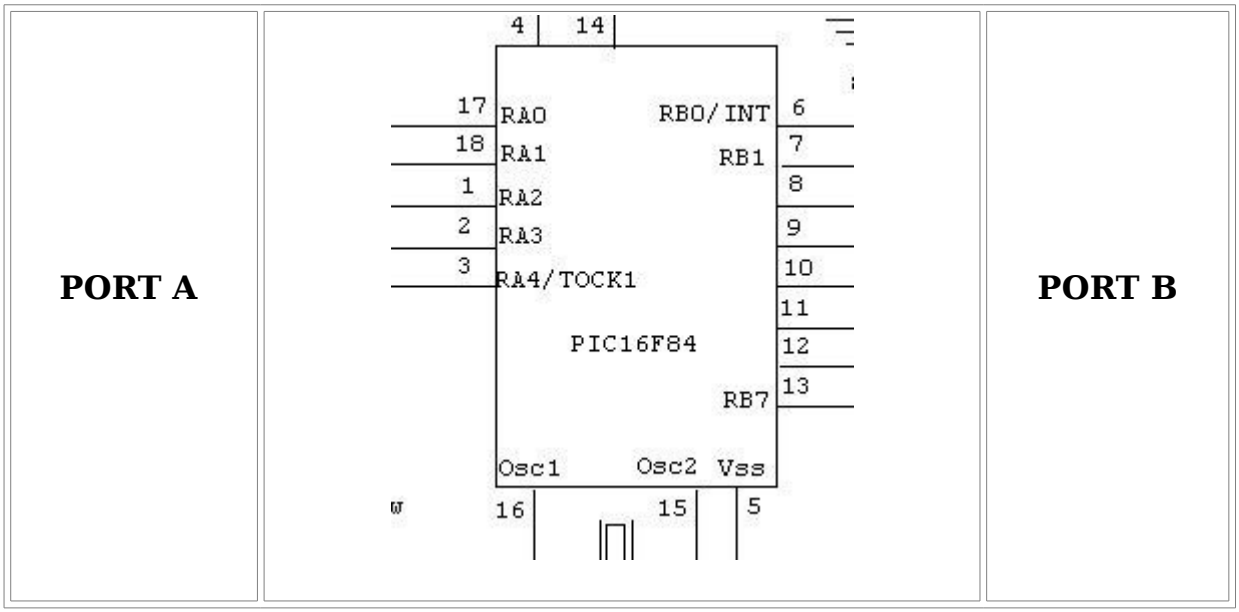
Podemos crear las subrutinas que necesitemos, de forma que evitaremos repetir instrucciones dentro del mismo programa y éste será más legible.

---

5 ENTRADAS Y SALIDAS EN EL MICROCONTROLADOR.

CONFIGURANDO LOS PUERTOS DE ENTRADA Y SALIDA.

El microcontrolador tiene 2 puertos PORTA (05H) y el PORTB (06H)



Cada línea de los puertos que corresponden a las patillas RA0-RA4 y RBO-RB7 pueden ser configurados como entradas o salidas.

Para configurar los puertos existen dos registros especiales TRISA y TRISB.

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

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

PORTA (05H) BANK0 BITS RA4-RA3-RA2-RA1-RA0 --> TRISA BANK1

PORTB (06H) BANK0 BITS RB7-RB6-RB5-RB4-RB3-RB2-RB1-RB0 --> TRISB BANK1

Vemos como existen 2 bancos, BANK0 y BANK1 para pasar de un banco a otro utilizamos el registro STATUS (03H) mediante el bit 5 de ese registro, de forma que si el bit 5 es 0 estamos en el banco 0 y si el bit 5 es 1 pasaremos al banco 1.

Dirección	bank0	bank1	Notas
03h	STATUS	STATUS	Bit 5=0 Bank0. Si bit 5=1 Bank1
05h	PORTA	TRISA	Bits=1 Entradas. Bits=0 Salidas
06h	PORTB	TRISB	Bits=1 Entradas. Bits=0 Salidas

Cuando se resetea el microcontrolador se accede al banco 0, para acceder al banco 1 hay que poner a 1 el bit 5 del registro STATUS

Nota sobre el registro STATUS

El registro STATUS es un registro especial que está en la dirección 03h.

Los bits del registro STATUS se les denominan banderas (flags), cuando arranca el microcontrolador las banderas están de la siguiente forma STATUS='00011XXX'

Estos 8 bits tiene el siguiente significado:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
IRP	RP1	RPO	/ T0	/ PD	Z	DC	C

Significado:

BIT	Significado
C Carry	Acarreo en las operaciones, se produce al desbordar la operación.
DC Digit carry	Activo cuando hay acarreo en el 4 bit de menor peso
Z Zero	Es 1 cuando el resultado de una operación es 0
/ PD Power down	0 si está en sleep, 1 cuando se activa clrwdt

/ T0 Timer out	Fin de temporización
RPO Register bank select	Selección de bancos RP0=0 bank0 RP0=1 Bank1
RP1	No usado
IRP	No usado

El proceso para configurar los puertos es el siguiente:

1. Al arrancar el micro controlador el banco por defecto es BANK0
2. Accedemos al banco1 BANK1 poniendo a 1 el bit5 del registro STATUS (03h)
3. Cargar en TRISA los valores adecuados (0 salidas 1 entradas)
4. Cargar en TRISB los valores adecuados (0 salidas 1 entradas)
5. Acceder al banco 0 BANK0 poniendo a 0 el bit 5 del registro STATUS
6. Realizar el programa para activar las entradas y salidas

Ejemplos de configuración del microcontrolador

Configurar todo salidas:

TRISA (05H BANK1) 00000

TRISB (06H BANK1) 00000000

Configurar todo entradas:

TRISA (05H BANK1) 11111

TRISB (06H BANK1) 11111111

Configuración de PORTA como entradas y PORTB como salidas:

TRISA (05H BANK1) 11111

TRISB (06H BANK1) 00000000

PORTA como entradas y PORTB la mitad entradas y la mitad salidas

TRISA (05H BANK1) 00000

TRISB (06H BANK1) 00001111

SECUENCIA PARA CONFIGURACIÓN DE LOS PUERTOS

1 Poner el bit 5 del registro STATUS a 1 para acceder al banco1

bsf 0x03,5 ;bit set file 0x03 (Status) bit 5 = 1

2 Configuración de TRISA y TRISB

movlw b'11111111' ;(1=entradas 0=salidas)

movwf 0x05 ;0x05= TRISA 8 entradas 0 salidas

movlw b'00000000' ;(1=entradas 0=salidas)

movwf 0x06 ; (TRISB PUERTOB salidas)

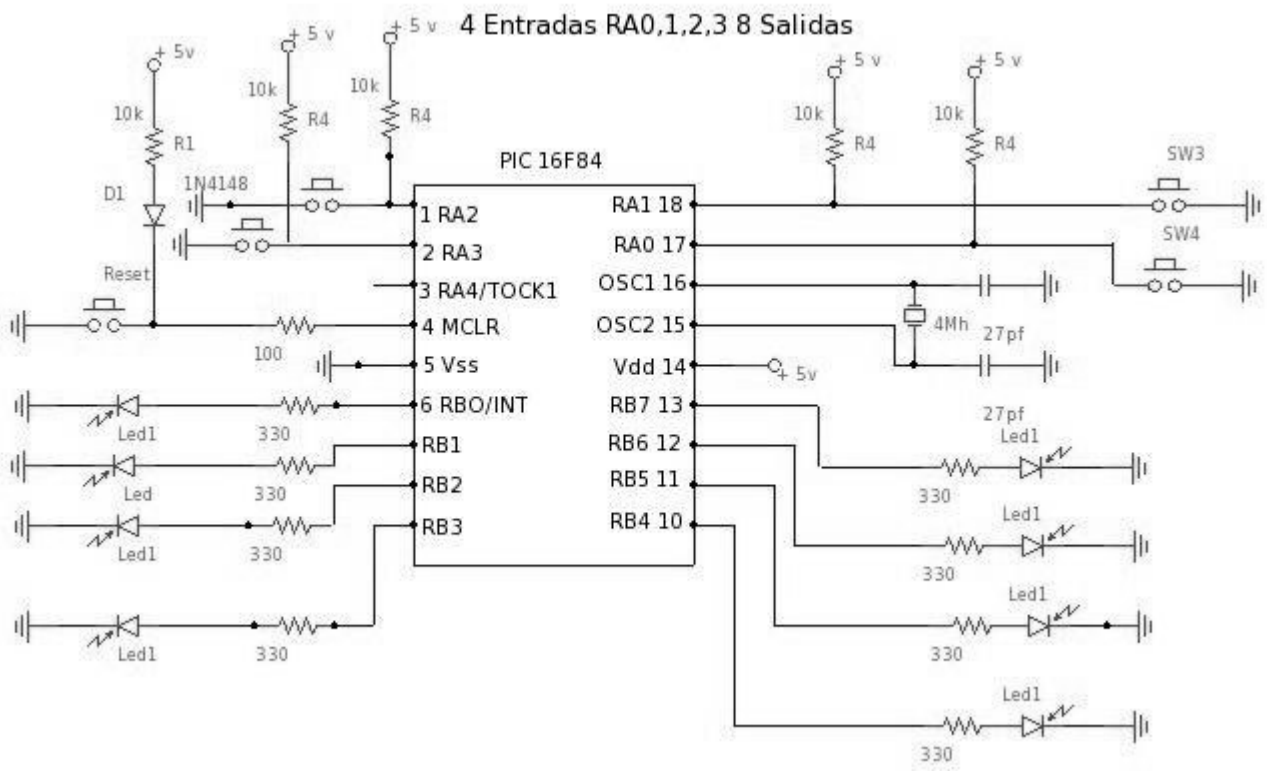
3 Poner a 0 el bit 5 del registro STATUS para volver al banco 0

bcf 0x03,5 ; Bit clear file (Ponemos a 0 para acceder al banco 0)

4 Iniciciamos el programa

EJEMPLO 10 Realizar un programa que lea interruptores en el port A y saque el valor por los leds en port B

Utilizaremos el siguiente esquema:

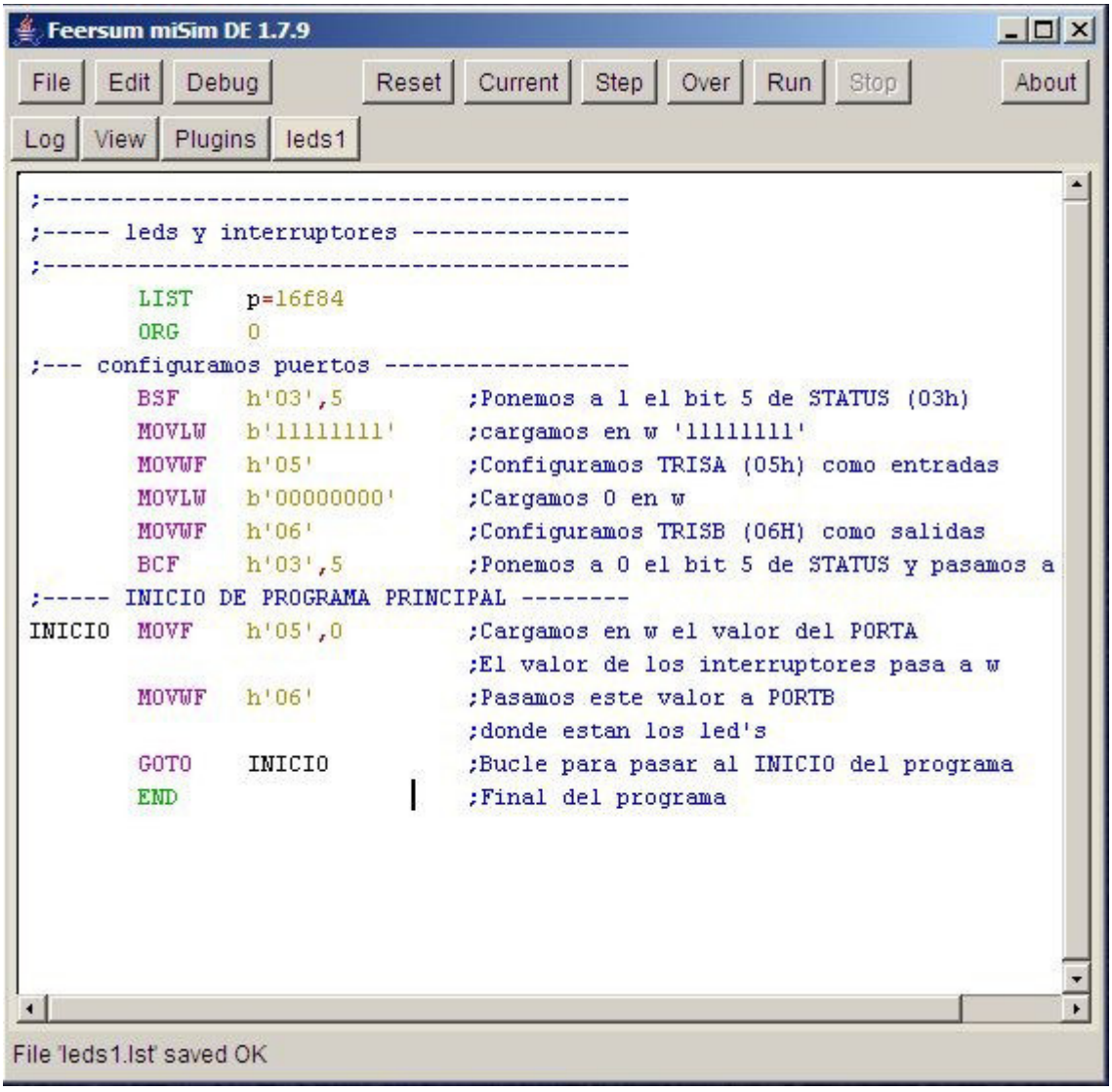


Tenemos 4 entradas en RA0 RA1 RA2 y RA3 y 8 salidas RB0 RB1 RB2 RB3 RB4 RB5 RB6 RB7

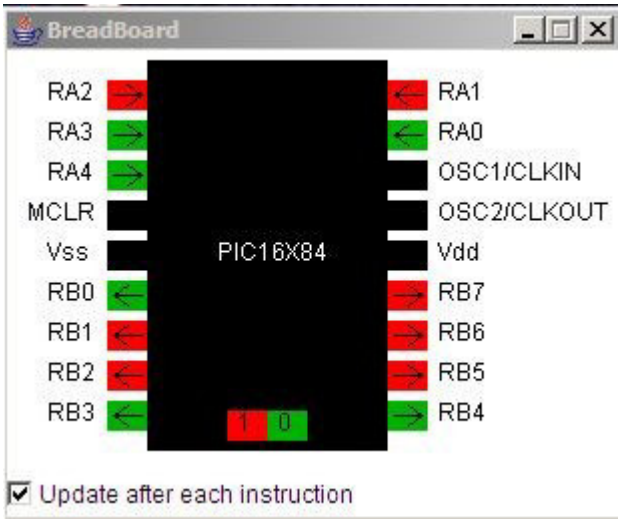
```
;----------------------------------------
;----- leds y interruptores ----------
;----------------------------------------

        list p=16f84
        org 0
        ;--- configuramos puertos ----------------
```

```
        bsf h'03',5 ;Ponemos a 1 el bit 5 de STATUS (03h)
        MOVLW b'11111111' ;cargamos en w '11111111'
        movwf h'05' ;Configuramos TRISA (05h) como entradas
        movlw b'00000000' ;Cargamos 0 en w
        movwf h'06' ;Configuramos TRISB (06H) como salidas
        bcf h'03',5 ;Ponemos a 0 el bit 5 de STATUS y pasamos
        a bank0

;----- INICIO DE PROGRAMA PRINCIPAL --------
INICIO

        MOVF h'05',0 ;Cargamos en w el valor del PORTA
        movwf h'06' ;Pasamos este valor a PORTB
        goto INICIO ;Bucle para pasar al INICIO del programa
        end ;Final del programa
```

Nuestro programa en el simulador FEERSUM MISIM DE 1.7.9



Podemos ver el estado del microcontrolador



Tambien el estado de los dispositivos de entrada (Interruptores) y los LEDS de salida

Vamos a ver el mismo programa utilizando variables

```
; ----- LEDS E INTERRUPTORES
            LIST P=16F84
            STATUS EQU H'03'
            PORTA EQU H'05'
            PORTB EQU H'06'
            ORG 0
;--- CONFIGURACIÓN DE PUERTOS
            BSF STATUS,5
            MOVLW H'FF'
            MOVWF PORTA
            MOVLW H'00'
            MOVWF PORTB
            BCF STATUS,5
; -- PROGRAMA PRINCIPAL
```

```
INICIO

        MOVF PORTA,0

        MOVWF PORTB

        GOTO INICIO

END ;
```

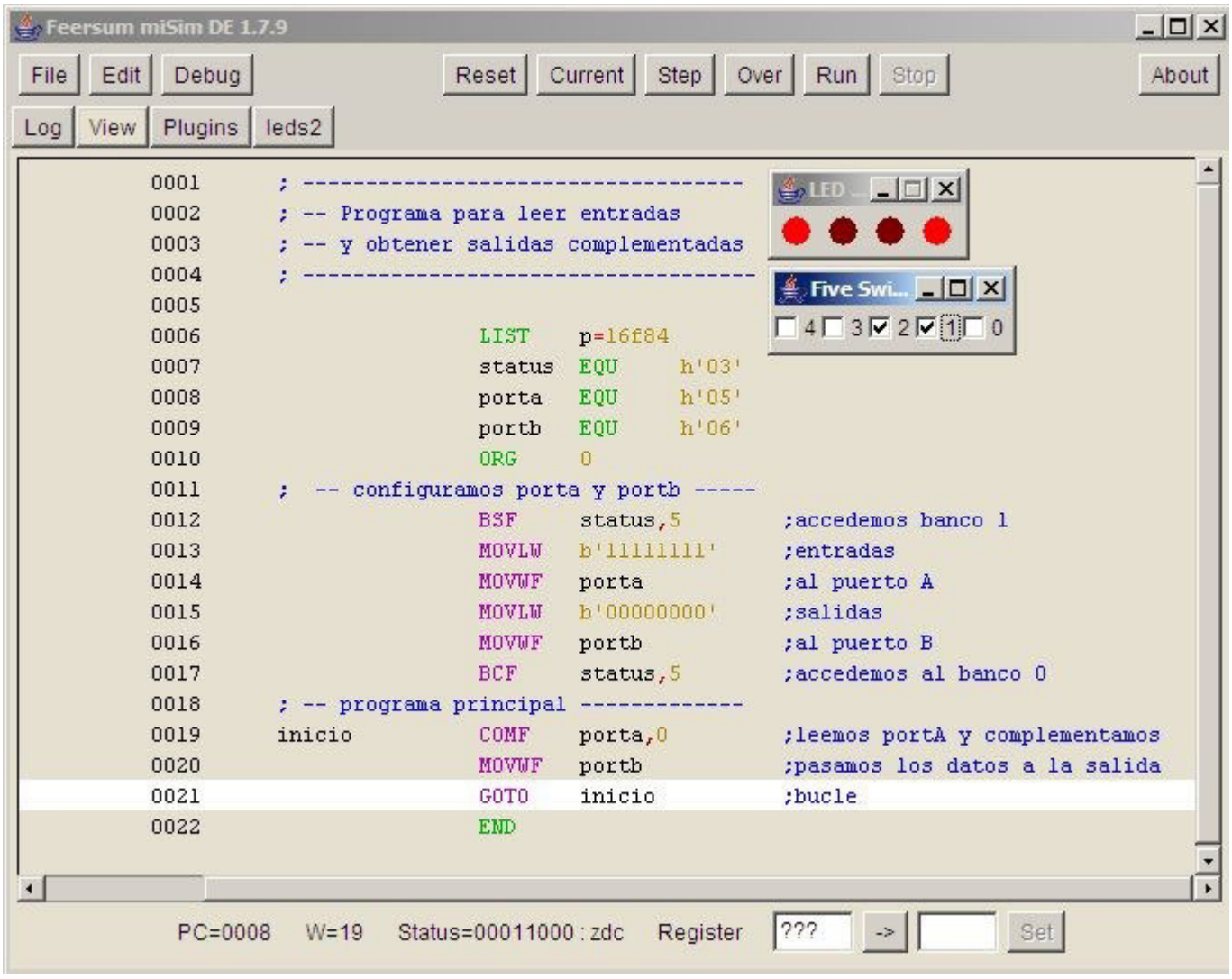
Ej 5.1 Poner comentarios al programa anterior.

EJEMPLO 11 Realizar un programa para leer entradas y obtener las salidas complementadas

Utilizaremos el mismo esquema de 4 entradas y 8 salidas.

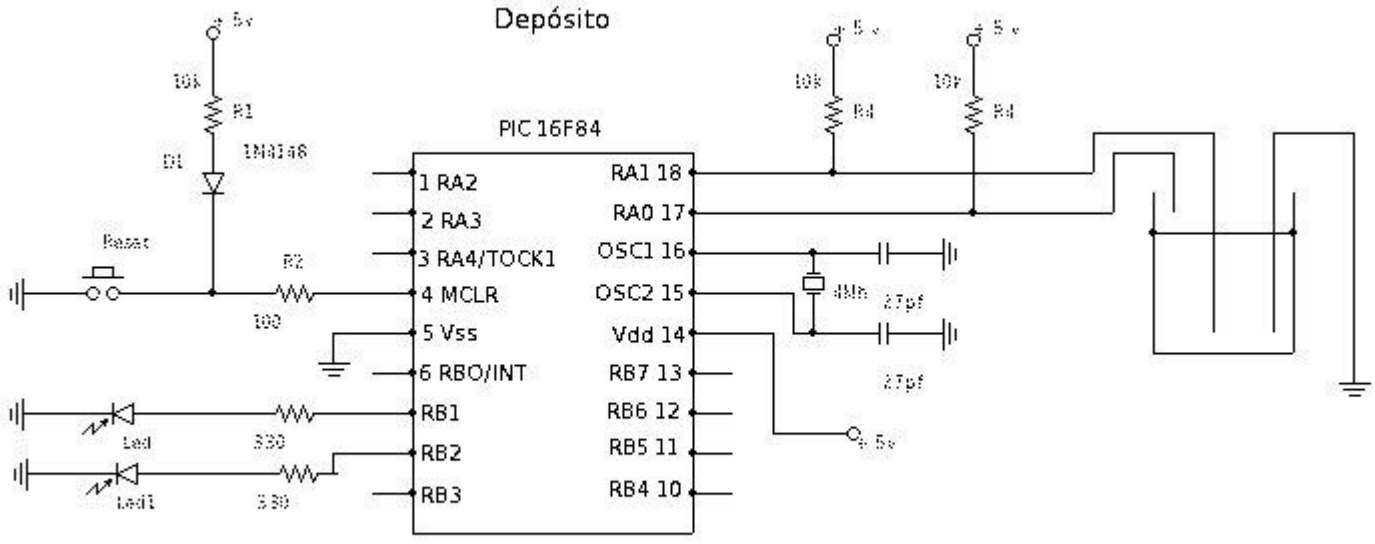
```
; ----------------------------------
; -- Programa para leer entradas
; -- y obtener salidas complementadas
; ----------------------------------

    list p=16f84
    status equ h'03'
    porta equ h'05'
    portb equ h'06'
    org 0
    ; -- configuramos porta y portb -----
    bsf status,5 ;accedemos banco 1
    movlw b'11111111' ;entradas
    movwf porta ;al puerto A
    movlw b'00000000' ;salidas
    movwf portb ;al puerto B
    bcf status,5 ;accedemos al banco 0
    ; -- programa principal ------------
    inicio comf porta,0 ;leemos portA y complementamos
    movwf portb ;pasamos los datos a la salida
    goto inicio ;bucle
    end
```

Nuestro programa simulado mediante misim



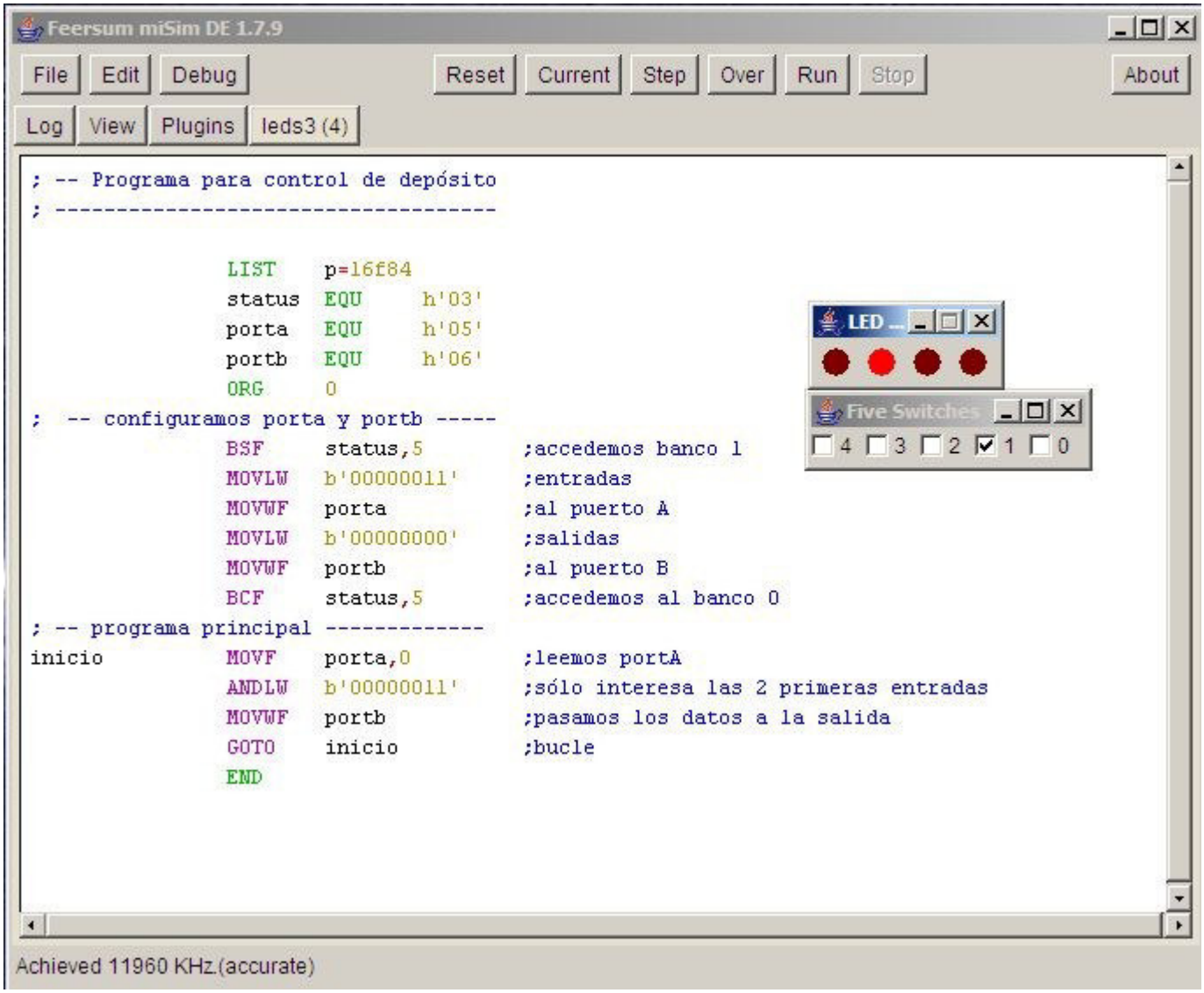
EJEMPLO 12 Diseñar un sencillo medidor de nivel

Utilizaremos el siguiente esquema



En RA1 Y RA0 tenemos 2 sondas de detección de nivel de agua y en RB1 y RB2 tenemos las salidas

Evidentemente este programa es de por sí poco útil, pero nos permite practicar con la programación de los puertos.

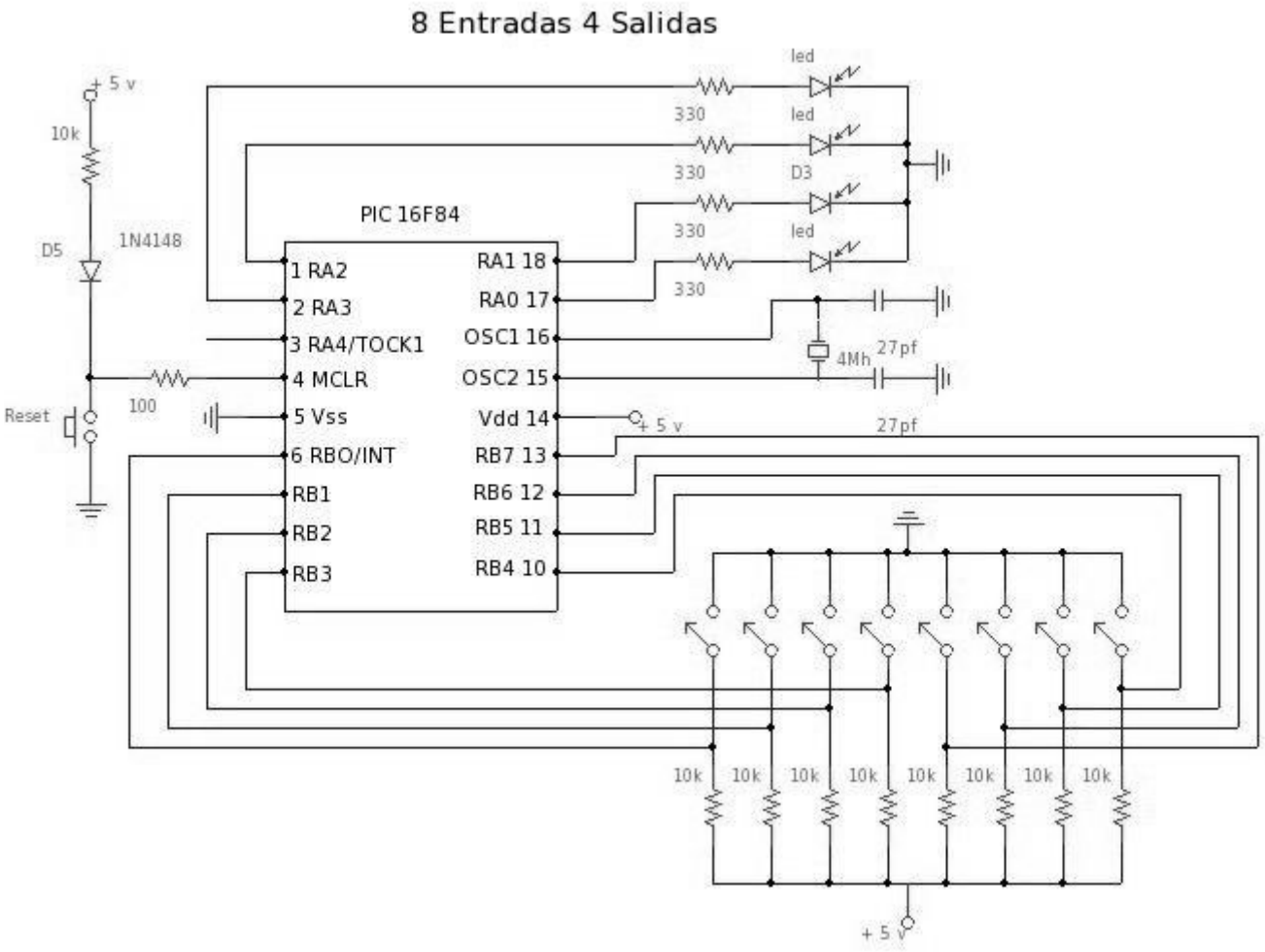


Nuestro programa simulado en misim

Ej 5.2 Modificar el programa para tener 4 niveles en el depósito.

Ej 5.3 Diseñar y realizar un programa con 8 entradas y 4 salidas. El programa sólo utilizará las 4 primeras entradas.

El programa leerá las entradas y sacará las salidas complementadas



Esquema para realizar el proyecto.

Ej 5.4 Utilizando 8 entradas, realizar una puerta AND de 4 bits (utiliza RB0-RB3 como una entrada y RB4-RB7 como la segunda entrada)

---

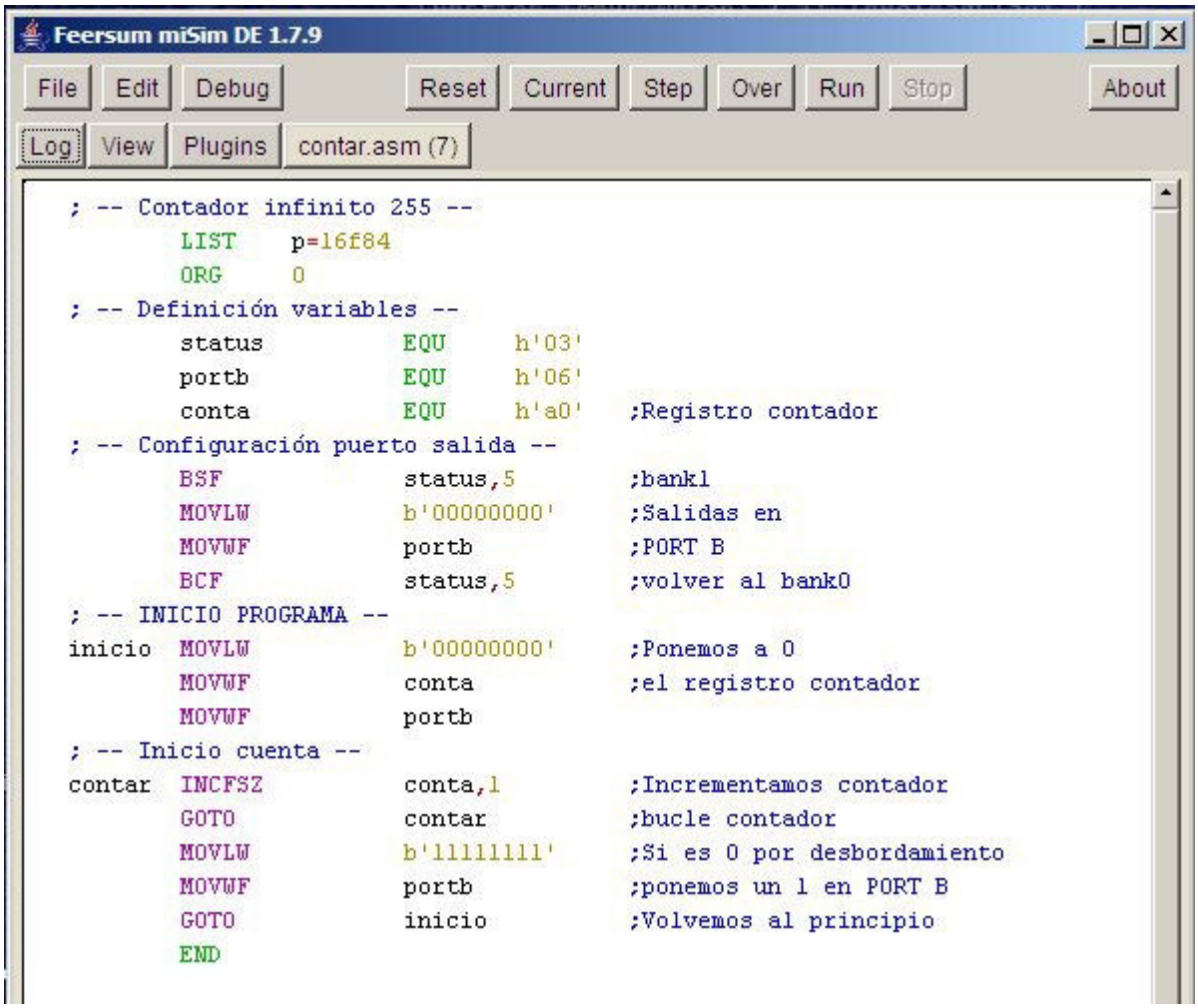
6 SALTOS EN FUNCIÓN DE BITS Y DE REGISTROS

INSTRUCCIÓN	EJEMPLOS	NOTAS
btfsc f,b	**btfsc registro,1**	**bit test file skip clear Si el bit b del registro es 1 la instrucción siguiente se ejecuta normalmente, si b=0 la instrucción siguiente se ignora y se salta**
btfss f,b	**btfss f,b**	**bit test file skip set. Si el bit b del registro es 0 la siguiente instrucción se ejecuta normalmente, si b=1la siguiente instrucción se salta**
decfsz f,d	**decfsz contador,1**	**decrement f, skip if 0. Decrementa el registro si el resultado NO es 0 la instrucción siguiente se ejecuta normalmente, si el resultado es cero la instrucción siguiente se ignora y se**

		salta
incfsz f,d	**incfsz contador,0**	**increment file skip if 0. Incrementa el contenido del registro, si el resultado NO es 0 la siguiente instrucción se ejecuta normalmente, pero si es 0 (por desbordamiento de 11111111) la instrucción siguiente se ignora y se salta**

Son funciones para tomar decisiones dependiendo del valor del bit o del registro, se utilizan para contar/descontar registros o para determinar si se ha producido un evento.

EJEMPLO 13 Programa contador hasta 255 y luego vuelve a 0.El resultado se visualiza con leds cuando llega a 255



Nuestro programa en el sinulador

Ej 6.1 Realizar un decrementador de 255 a 0

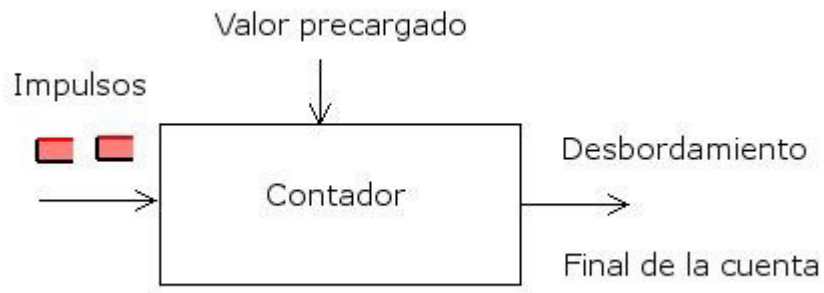
---

7 TEMPORIZACIÓN EN EL PIC 16F84

Cuando se necesita un control del tiempo en el microcontrolador, se utiliza el timer TMR0, que puede configurarse como CONTADOR y como TEMPORIZADOR.

El timer TMR0 es un contador que determina el tiempo que tarda en desbordarse el contadordependiendo del valor precargado.

Su funcionamiento es:



El contador el PIC es de 8 bits que va desde b'00000000' hasta b'11111111', cuando se produce el desbordamiento vuelve al valor inicial b'00000000'.

Los impulsos de entrada del contador pueden ser externos mediante el pin RA4/TOCK1 o bien de una señal del reloj interno, de forma que puede funcionar de dos formas distintas:

1- Como contador de impulsos que entran por RA4/TOCK1

2-Como temporizador de tiempos.

Para configurar la forma de actuación hay que acceder al registro OPTION y actuar sobre el bit TOCS

Si TOCS=1 el timer actua como contador.

Si TOCS=0 el timer actua como temporizador.

Bank 0								
00h	INDF	Uses contents of FSR to address data memory (not a physical register)						
01h	TMR0	8-bit real-time clock/counter						
02h	PCL	Low order 8 bits of the Program Counter (PC)						
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC
04h	FSR	Indirect data memory address pointer 0						
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1
07h		Unimplemented location, read as '0'						
08h	EEDATA	EEPROM data register						
09h	EEADR	EEPROM address register						
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)			
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF
Bank 1								
80h	INDF	Uses contents of FSR to address data memory (not a physical register)						
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1

Registros y bits importantes para control del temporizador

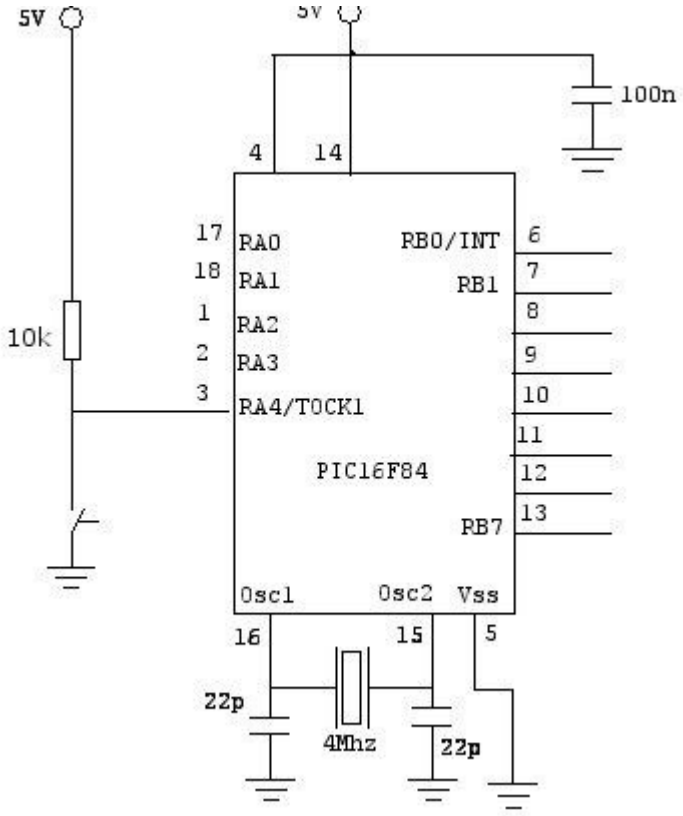
TMRO COMO CONTADOR

Se introducen los impulsos a contar por el pin RA4/TOCK1

Se programa el bit TOSE del registro OPTION_REG de la siguiente forma:

TOSE=1 Actua por flanco actico descendente

TOSE=0 Actua por flanco activo ascendente



Circuito utilizado para la entrada de un contador.

Cuando se desborda el contador pasa de '11111111' a '00000000' y se activa el bit TOIF del registro INTCON

TMRO COMO TEMPORIZADOR

Al trabajar como temporizador se cuentan los impulsosde la frecuencia de oscilación o frecuencia de trabajo, teniendo en cuenta que un ciclo de máquina ocupa 4 ciclos del cristal oscilador. Así tendremos que la frecuencia de conteo será

Fc=Frecuencia oscilación/4

Para nuestro caso, que el reloj es de 4Mh, tendremos Fc=4Mhz/4 = 1 microsegundo

Como 1 us es un periodo de tiempo muy pequeño, el microcontrolador tiene la posibilidad de utilizar un divisor de frecuencia llamado prescaler, el prescaler se configura con los bits PS2,PS1,PS0 del registro REG_OPTION.

BITS DE CONFIGURACIÓN Y LECTURA DEL TEMPORIZADOR TMR0.

Los bits de lectura y configuración son los siguientes:

Dirección	**Registro**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
01h bank0	TMRO	X	X	X	X	X	X	X	X
0bh bank0	INTCON	X	X	X	X	X	TOIF	X	X

01h bank 1	OPTION_REG	X	X	TOCS	TOSE	PSA	PS2	PS1	PS0

Registro **TMRO**: Valor del contador o del temporizador.

Registro **INTCON**:

Bit 2 - TOIF: TMRO Overflow Interrupt Flag , Bandera de interrupción del contador o temporizador TMRO, cuando el registro TMRO pasa de b'11111111' a b'00000000' el valor de TOIF pasa a 1

Registro **OPTION_REG:**

Bit 5 - TOCS (TMRO Clock Source Select Bit)

TOCS = 0 Pulsos del reloj interno F oscilación / 4; el TMRO se programa como temporizador.

TOCS = 1 Pulsos externos introducidos por el pin RA4/TOCK1, TMRO actua como contador.

Bit 4 - TOSE (TMRO Source Edge select Bit)

TOSE = 0 El contador del pin RA4/TOCK1 actua con flanco ascendente

TOSE = 1 El contador actua con flanco descendente

Bit 3 PSA (Prescaler Asignment Bit)

PSA = 0 El divisor de frecuencia se asigna a TMRO (Contador o temporizador)

PSA = 1 Divisor asignado al Watchdog (no explicado todavia)

Bit 2 Bit 1 Bit 0 (Prescaler Rate Select Bits) Bits de selección del divisor de frecuencia.

PS2 PS1 PS0	Divisor del Prescaler TMRO
000	1:2
001	1:4
010	1:8
011	1:16
100	1:32
101	1:64

110	1:128
111	1:256

EJEMPLOS DEL TMRO COMO CONTADOR

Para programar un contador con el pic necesitamos:

1- Pasar al banco 1 BSF STATUS,5

2- Programar los bits 4 TOSE y el bit 5 TOCS del registro REG_OPTION (01H) de la siguiente forma:

TOCS=1 Contador

TOSE=0 activo por flanco descendente

TOSE=1 activo por flanco ascendente

PSA=1 prescaler al wachdog ya que no necesitamos prescaler al contador

movwl b'00111000' ;Contador y activado por flanco descenente

movwf REG_OPTION

3- Pasar al banco 0 BCF STATUS,5

4- Iniciar el contador a 0 clrf TMRO

5 - Leer TMRO movf TMRO,0

EJEMPLO 14. UTILIZAR EL CIRCUITO DE CONTADOR PARA CONTAR IMPULSOS Y VISUALIZARLOS EN LEDS DE SALIDA

```
; ------------------------
; -- programa contador ----
; -- visualizar por leds --
; -- la cuenta en PORTB ---
; ------------------------

LIST p=16f84

TMRO EQU H'01' ;Registro de configuración y contador
STATUS EQU H'03' ;Registro de STATUS
PORTB EQU H'06' ;Registro Puerto B de salida

org 0
; -- CONFIGURACION DEL CONTADOR Y EL PUERTO SALIDA

BSF STATUS,5 ;Acceso al banco 1
```

```
MOVLW b'00111000' ;Configuramos TMRO como contador
MOVWF TMRO ;por flanco ascendente
MOVLW b'00000000' ;configuramos como salida
MOVWF PORTB ;el puerto B
BCF STATUS,5 ;Pasamos al banco 0
CLRF TMRO ;iniciamos el contador a 0

;-- Programa principal ---

inicio MOVF TMRO,0 ;Cargamos el contador en w
MOVWF PORTB ;Pasamos el valor cuenta al PORTB
GOTO inicio ;Bucle
END ;Final de programa
```

EJEMPLOS DEL TMRO COMO TEMPORIZADOR

Para programar el temporizador necesitamos calcular el tiempo de temporización, aplicaremos la siguiente fórmula:

Tiempo = 4 * Tiempo Frecuencia oscilador * Prescaler * (256-Carga en el temporizador)

Tiempo frecuencia oscilador (Para 4 Mhz) = 1 u segundo

Prescaler desde 1/2 hasta 1/256

Para programar un temporizador con el pic necesitamos:

1- Pasar al banco 1 BSF STATUS,5

2- Programar los bits 4 TOSE, el bit 5 TOCS y los bits 3,2,1,0 del registro REG_OPTION (01H) de la siguiente forma:

TOCS=0 Temporizador

TOSE=0 activo por flanco descendente

TOSE=1 activo por flanco ascendente

PSA (bit 3)=0 prescaler al temporizador

PS2,PS1,PS0 (bits 2,1,0)Según la tabla de selección del prescaler

movwl b'11010101' ;Temporizador flanco ascendente PSA=0 Prescaler al temporizador y prescaler 101 (1/64)

movwf REG_OPTION

3- Pasar al banco 0 BCF STATUS,5

4- Iniciar el temporizador a 0 clrf TMRO

5 - Leer TMRO movf TMRO,0

```
; -------------------------------------------------
; -- Programa de temporización del parpadeo de un led
; -------------------------------------------------
list p=16f84
portb equ h'06'
timer equ h'01'
estatus equ h'03'
org 0
; --------- iniciamos puertos y temporizador --
bsf estatus,5 ; Pasamos al banco 1
movlw b'11010101' ; Cargamos el valor en el
movwf timer ; Registro Option del timer
movlw b'00000000' ; Cargamos 00 en el registro
movwf portb ; Puerta B (salidas)
bcf estatus,5 ; Pasamos al banco 0
clrf portb ; Ponemos la salida PORTB a 0
; -- Programa principal --
inicio bsf portb,0 ; Ponemos un 1 en la salida 0 del port B
call temp
bcf portb,0 ; Ponemos a 0 el bit 0 del port B
call temp ; Llamamos a la subrutina del temporizador
goto inicio ; Volvemos al inicio
; -- Subrutina de temporización --
temp clrf timer ; Ponemos el temporizador a 0
bucle btfss timer,7 ; Comprobamos si el temporizador llega a b'10000000'
goto bucle ; Volvemos al bucle
return ; Si el valor es b'10000000' del temporizador volvemos de la
; subrutina
; -- Final de subrutina --
end
```

Ejercicios propuestos:

Ejercicio 7.1 Realizar un contador de decenas y obtener el resultado en un display.

Ejercicio 7.2 realizar un programa para encender los leds del portb de la siguiente forma:

Encendemos led 0

Temporizamos (0.5 segundos)

Apagamos led0 y encendemos led1

Temporizamos

Apagamos led1 y encendemos led2

....

Cuando llegamos al led7 apagamos led7 temporizamos y volvemos a encender el led 1

Ejercicio 7.3 Realizar un semáforo temporizado cuyo bucle sea:

Encendemos verde y temporizamos

Encendemos led amarillo parpadeante

Encendemos led rojo y temporizamos

Volvemos al inicio.
