PIC

16F87X

TRABAJO

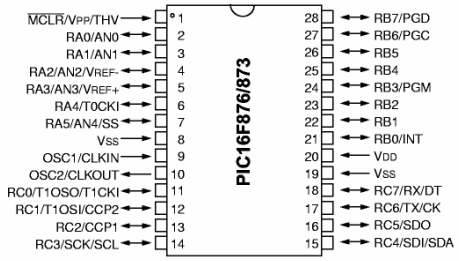
EXPLICACIÓN

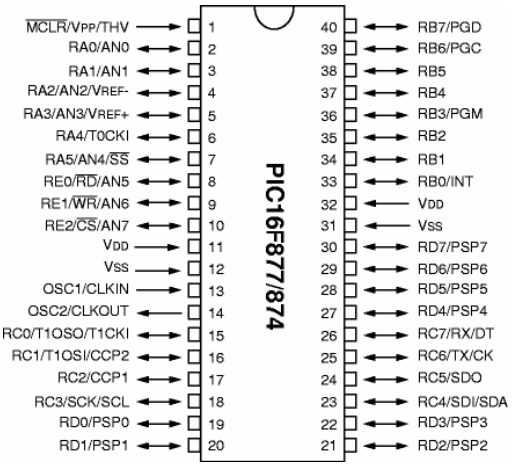
Sebastián Martín García

2º DPE

INDICE

DIFERENCIAS ENTRE PIC16F84 Y 16F87X





MICROCHIP

PIC 16F87X

Diferencias entre 16F84 y 16F87X

El PIC 16F84 ha sido precedido por el 16C84, prácticamente igual, con excepción de la memoria de programa que era de tipo EEPROM en lugar de FLASH. El FLASH soporta 1.000 operaciones de Escritura/Borrado y el EEPROM 100.000.

El PIC 16F84 tiene una memoria FLASH de 1K palabras, solo un Timer y 13 líneas de E/S digitales y el modelo normal soporta una frecuencia de 10 MHz. Aunque el "A", llega a 20 MHz. Es un microcontrolador categorizado como gama baja por su bajo coste y sencillez, pero que ha dado mucho que hablar.

La memoria RAM de datos de los PIC 16F87X posee una capacidad de 192 bytes en dos de los modelos y de 368 bytes en los otros dos. Aunque superan ampliamente los 68 bytes del 16F84 mantienen la misma estructura básica de 4 bancos de 128 bytes cada uno, seleccionables por los bits RP0 y RP1 del registro de estado (STATUS bits 5 y 6 respectivamente).

La memoria de datos no volátil de 64 bytes tipo EEPROM que tenía el 16F84, en los nuevos 16F87X de 28 patas sube a 128 bytes, y en los de 40 patas hasta 256 bytes.

Los 16F87X manejan hasta 14 posibles fuentes de interrupción y 3 Timer, frente a las 4 fuentes y 1 Timer del 16F84. El numero de puertas también se ha aumentado considerablemente, con 3 puertas los de 28 patas y hasta 5 puertas los de 40.

Además lo nuevos PIC´s, incorporan los siguientes módulos, inexistentes en el antiguo 16F84:

- *Dos módulos CCP:*

  Capaces de comparar y capturar impulsos. La captura se efectúa con una precisión de 12,5 ns y una resolución de 16 bits, mientras que la comparación con igual resolución alcanza una precisión de 200 ns. Además, la sección PWM varía la anchura de los impulsos, técnica muy empleada en los motores.

- *Comunicación Serie:*

  La típica USART, orientada a la comunicación entre subsistemas o máquinas (RS-232) y la MSSP destinada a la comunicación entre diversos circuitos integrados y que admite el protocolo I2C y SPI.

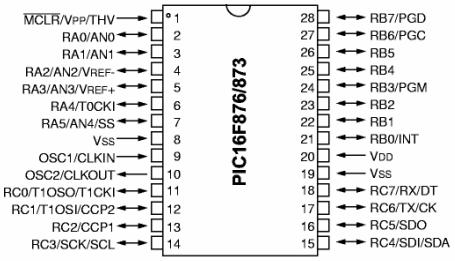
- *Comunicación en Paralelo:*

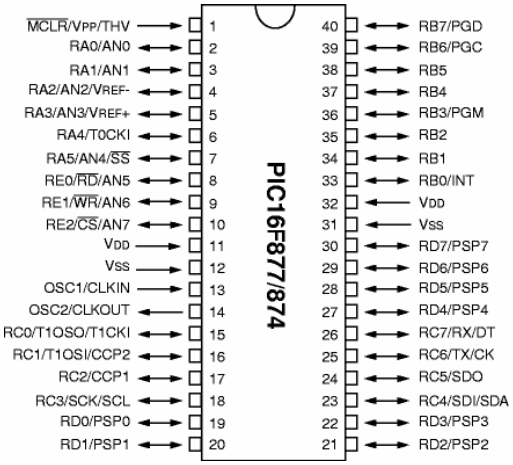
  Los PIC 16F874/7 de 40 patas está disponible el protocolo PSP, más rápido que la comunicación serie pero hipoteca muchas minas de E/S, 8 de la puerta D y 3 de control de la Puerta E.

- *Conversor A/D:*

  En todos los PIC 16F87X existe un conversor A/D de 10 bits, con 5 canales de entrada en los de 28 patas y 8 en los de 40.

SENSORES ANALOGICOS MAS UTILIZADOS





Sensores Analógicos más utilizados

Aunque el PIC 16F84 sirve para un sinfín de aplicaciones, hay varias para las que este microcontrolador no sirve. Un de ellas es la que vamos a tratar en este apartado, se trata de aplicaciones en las que sea necesario un conversor A/D para su tratamiento. Vamos a tratar un par de sensores analógicos, una LDR y un sensor de temperatura NTC.

Sensor de Luminosidad LDR

Un sensor de luminosidad LDR es un elemento cuya resistencia entre bornes varía en función de luz que incide sobre su superficie. Cuando no hay luz tiene una resistencia infinita y según va aumentando la luz, va disminuyendo hasta 0.

Es un elemento sin polaridad y se puede encontrar con diferentes diámetros según el rango de valores de luminosidad que sea capaz de diferenciar.

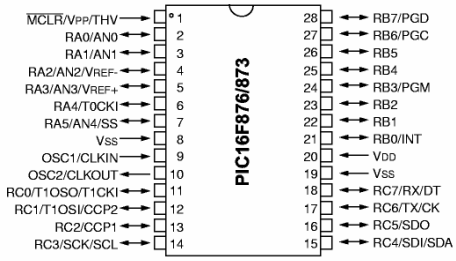
Sensor de Temperatura LM35

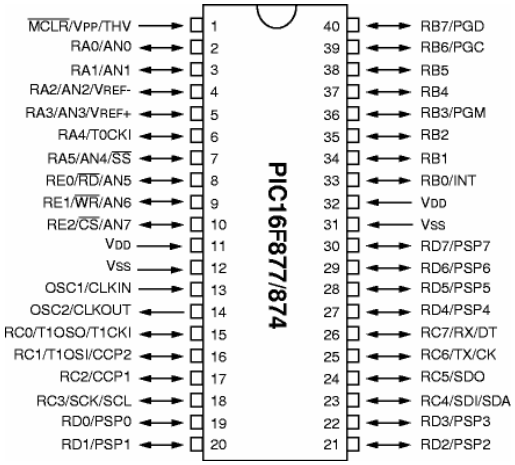
Otro sensor analógico ampliamente utilizado es el de temperatura. Existen muchos modelos de sensores de temperatura y su elección depende de varios parámetros, rango de temperaturas, precisión, el coste, resistencia, etc.,...

Uno de los sensores más utilizados es el LM35 también llamado estándar. Sus características son:

1. Su tensión de salida Vout es proporcional a la temperatura en una proporción de 10mV/ºC
2. Su rango de funcionamiento esta comprendido entre 0º y 100º C.
3. Su tensión de funcionamiento Vs está entre +4 VDC y +30 VDC.
4. Su precisión es de ± 0.9 ºC.

ORGANIZACIÓN DE LA MEMORIA





MICROCHIP

Organización de la Memoria

Memoria de Programa

La memoria FLASH en la que se graba el programa de aplicación en los PIC 16F87X, puede tener una capacidad de 4K u 8K palabras de 14 bits cada una. Dicha memoria está dividida en páginas de 2K palabras y está direccionada con el PC, que tiene un tamaño de 13 bits. La pila que tiene 8 niveles de profundidad, es transparente para el usuario, es decir, funciona automáticamente y no dispone de instrucciones para guardar o sacar de ella información. Con la instrucción CALL y con las interrupciones, el valor se salva en el nivel superior. Con las instrucciones RETURN, RETFIE Y RETLW, en valor contenido en el nivel superior de la pila, se carga en el PC. Al poseer la pila solo 8 niveles, le corresponde al programador preocuparse por los anidamientos en las subrutinas para sobrepasar dicho valor. El vector de reset ocupa la dirección 0000h y el vector de interrupción la 0004h, igual que el PIC 16F84.

Memoria de Datos RAM

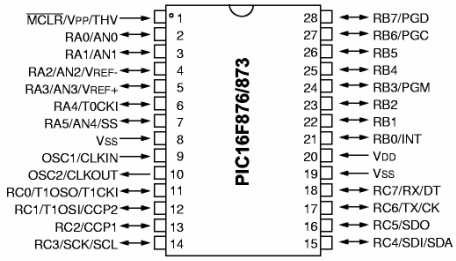
La memoria de datos tiene posiciones implementadas en RAM y otra en EEPROM. En la sección RAM, se alojan los registros operativos fundamentales, en el funcionamiento del procesador y en el manejo de sus periféricos, además de registros que el programador puede usar para información de trabajo propia de la aplicación. La memoria EEPROM es para guardar datos de forma no volátil y se considera un dispositivo especial.

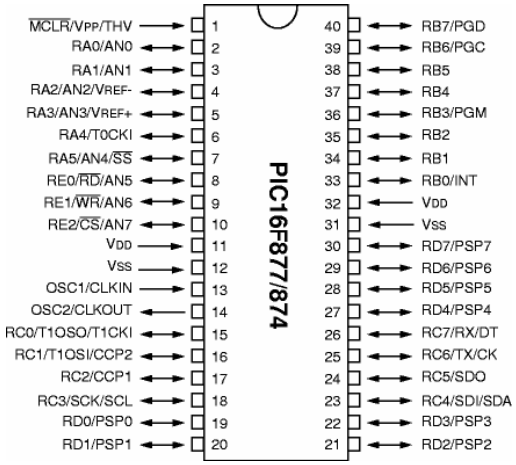
La RAM estática consta de 4 bancos con 128 bytes cada uno. En las posiciones iniciales de banco se ubican los registros específicos que gobiernan el procesador y sus recursos. Dos modelos de 16F87X tienen 192 bytes de RAM y otros dos de 368 bytes.

Para seleccionar el banco al que se desea acceder en la RAM se emplean los bits 6 y 5 del *Registro de Estado* (STATUS) RP1 y RP0 respectivamente, según el código siguiente:

BANCO	RP1	RP0
0	0	0
1	0	1
2	1	0
3	1	1

INSTRUCCIONES



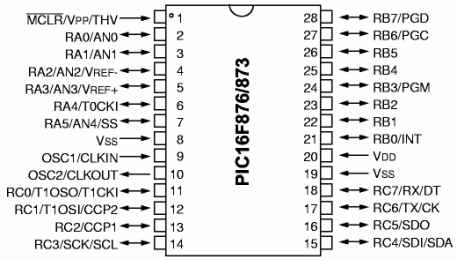


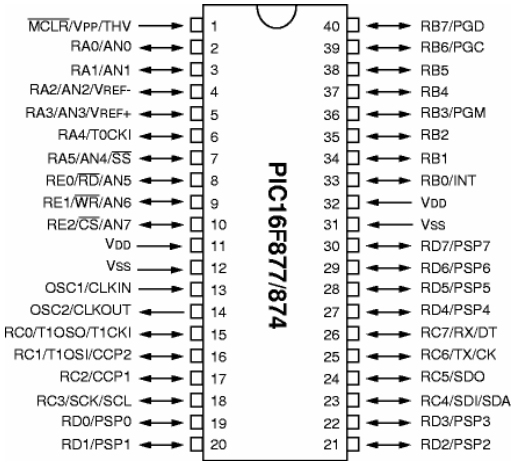
Instrucciones

Los mismos formatos, iguales modos de direccionamiento y las mismas 35 instrucciones que tenía el PIC 16F84 sirven para todos los modelos PIC 16F87X. No obstante, en los nuevos PIC, al contener más recursos, existen nuevos registros específicos de control cuyos bits se deberán escribir o leer para su gobierno.

INSTRUCCIONES QUE MANEJAN REGISTROS	
ADDWF	Suma W y F
ANDWF	AND W con F
CLRF	Borra F
CLRW	Borra W
COMF	Complementa F
DECF	Decrementa F
DECFSZ	Decrementa F, si es 0 salta
INCF	Incrementa F
INCFSZ	Incrementa F, si es 0 salta
IORWF	OR entre W y f
MOVF	Mueve f
MOVWF	Mueve W a f
NOP	No opera
RLF	Rota f a la izquierda, a través del acarreo
RRF	Rota f a la derecha, a través del acarreo
SUBWF	Resta a f el registro W
SWAPF	Intercambia f
XORWF	XOR de W con f
INSTRUCCIONES QUE MANIPULAN BITS	
BCF	Borra bit de f
BSF	Pone a 1 el bit de f
BTFSC	Testea un bit de f y salta si vale 0
BTFSS	Testea un bit de f y salta si vale 1
INSTRUCCIONES DE CONTROL Y DE OPERANDOS INMEDIATOS	
ADDLW	Suma inmediata a W
ANDLW	AND inmediato con W
CALL	Llamada a subrutina
CLRWDT	Borra el Perro guardián
GOTO	Salto incondicional
IORLW	OR inmediato con W
MOVLW	Mueve a W un valor inmediato
RETFIE	Retorno desde interrupción
RETLW	Retorno y carga de W
RETURN	Retorno de subrutina
SLEEP	Pasa a estado de reposo
SUBLW	Resta W de un inmediato
XORLW	OR Exclusiva a W

REGISTROS ESPECÍFICOS





Registros

➢ *REGISTRO DE ESTADO (STATUS)*

Este es el registro más usado de todos pues sus bits están destinado a controlar las funciones vitales del procesador. Por ese motivo está duplicado en las cuartas posiciones de cada banco (03h, 83h, 103h, 183h)

IRP	RP1	RP0	TO#	PD#	Z	DC	C

Los tres bits de menos peso son los señalizadores de ciertas condiciones en las operaciones lógico-aritméticas:

- Z: Señalizador de cero. Se pone 1 cuando el resultado es 0.
- C: Acarreo-llevada del 8º bit. Se pone a uno automáticamente cuando existe acarreo en el bit de más peso en las instrucciones de suma. También actúa como señalizador de llevada en las instrucciones de resta, pero en este caso la correspondencia es inversa, si vale 0 es llevada.
- DC: Acarreo-llevada en el cuarto bit. Funciona igual que el señalizador C, pero para el 4º bit. Es muy útil para las operaciones en BCD.

Los señalizadores PD# y TO#, son activos por nivel bajo (#) y sirven para indicar la causa que ha provocado la reinicialización del procesador.

- PD#: Se activa a 0 al ejecutarse la instrucción SLEEP. Se pone a uno automáticamente tras la conexión de alimentación o bien al ejecutarse la instrucción CLRWDT
- TO#: Se activa a nivel bajo al desbordarse el perro guardián. Toma el valor 1 tras la conexión de alimentación o al ejecutarse las instrucciones CLRWDT o SLEEP.

Los PIC se resetean al conectar la alimentación (POR – Power on Reset). También se resetean cuando la tensión de alimentación baja de 4V (BOR – Brown on Reset), aunque esta función es factible desactivarla poniendo a 0 el bit BODEM, presente en la palabra de configuración, tanto en el Reset POR como en el BOR los bits PD# y TO# toman el valor 1, mientras que en los demás casos dependen de la causa que ha provocado el Reset.
Finalmente los tres bits de más peso del registro de estado se emplean para seleccionar el banco de la RAM al que se desea acceder

RP1	RP0	BANCO SELECCIONADO
0	0	Banco 0 (00h – 7Fh)
0	1	Banco 1 (80h – FFh)
1	0	Banco 2 (100h – 17Fh)
1	1	Banco 3 (180h – 1FFh)

El bit IRP se usa conectado con el bit de más peso del registro FSR para elegir el banco de RAM en el direccionamiento indirecto.

➢ *REGISTRO DE OPCIONES (OPTION)*

Tiene las mismas funciones que tenía en el PIC 16F84:

1ª: Asigna el divisor de frecuencias al TIMER0 o al perro guardián.

2ª: Elige el rango en el que trabaja el divisor de frecuencia.

3ª: Selecciona el tipo de reloj del TIMER0, que puede ser interno o externo a través de la pastilla TOCKI. También selecciona el flanco activo.

4ª: Selecciona el flanco activo para la interrupción externa por RB0/INT

5ª: Activa o desactiva las resistencias de pull-up de la Puerta B

El registro OPTION toma el valor b'11111111' (FF) en cualquier tipo de reinicialización que se produzca.

RBPU#	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0

PS2	PS1	PS0	División del TMR0	División del WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

PSA: Asignación de divisor de frecuencias.
1 = El divisor de frecuencias se le asigna al WDT.
0 = El divisor de frecuencias se le asigna al TIMER0.

TOSE: Tipo de flanco en TOCKI.
1 = Incremento del TIMER0 cada flanco descendente.
0 = Incremento del TIMER0 cada flanco ascendente.

TOCS: Tipo de reloj para el TIMER0
1 = Pulsos introducidos a través del TOCKI (contador)
0 = Pulsos de reloj internos FOSC/4 (Temporizador)

INTEDG: Flanco activo de la interrupción externa.
1 = Flanco ascendente
0 = Flanco descendente

RBPU#: Resistencias de pull-up de la Puerta B
1 = Desactivadas
0 = Activadas

➢ *REGISTRO PARA CONTROLAR LAS INTERRUPCIONES*

Los PIC 16F87X tienen muchas causas que pueden originar una interrupción, 13 posibles causas los de 28 patas y 14 los de 40. Al aceptarse una interrupción se salva el valor del PC en la Pila y se carga aquel con el valor 0004h, que es el vector de interrupciones.
El PIC 16F84 tenía 4 causas que generaban interrupción: desbordamiento del TMR0, activación de la pata de interrupción RB0/INT, cambio de estado de una de las cuatro patas de más peso de la Puerta B y finalización de la escritura de un byte en la EEPROM. Los nuevos PIC, además de las causas que producen interrupción en el 16F84, tienen las siguientes:

1ª Desbordamiento del Timer 1
2ª Desbordamiento del Timer 2
3ª Captura o comparación del módulo CCP1.
4ª Captura o comparación del módulo CCP2.
5ª Transferencia en la Puerta serie síncrona.
6ª Colisión de bus en la Puerta serie síncrona.
7ª Fin de transmisión en el USART.
8ª Fin de recepción en el USART.
9ª Fin de la conversión en el conversor A/D
10ª Transferencia en la puerta paralela esclava (solo en los de 40 patas)

1. REGISTRO DE CONTROL DE INTERRUPCIONES (INTCON)

Se trata de un registro leíble y escribible, para facilitar su acceso se ha duplicado en los cuatro bancos. Tiene la misión de controlar las interrupciones provocadas por TMR0, cambio de estado en las cuatro líneas de más peso de la Puerta B y activación en la patilla RB0/INT. Es muy parecido al registro que con el mismo nombre existí en el 16F84, solo cambia el bit 6 en los nuevos PIC que es el PIE (permiso de interrupción de lo periféricos) en lugar del EEIE que tenía de 16F84 para permitir la interrupción cuando finalice la escritura de un byte en la EEPROM. El bit PEIE actúa como una segunda llave parcial de permiso o prohibición de las causas de interrupción que no está complementadas en INTCON y que las provocan los restantes periféricos del microcontrolador. GIE es el bit de permiso global de todas las interrupciones.

GIE	PEIE	TOIE	INT	RBIE	TOIF	INTF	RBIF

GIE: Bit de permiso global de interrupciones.
1 = Permitido
0 = Prohibido

PEIE: Bit de permiso de los periféricos que no se controlan con INTCON

TOIE: Bit de permiso de interrupción del TMR0.

INTE: Bit de permiso de la interrupción externa por RB0/INT

RBIE: Bit de permiso de la interrupción por cambio en RB4-RB7

TOIFF: Señalizador de desbordamiento en TMR0.

INTF: Señalizador de activación de la patilla RB0/INT

RBIF: Señalizador de cambio en RB4-RB7

2. REGISTRO DE PERMISO DE INTERRUPCIONES 1 (PIE1)

Contiene los bits que permiten o prohiben las interrupciones provocadas por los periféricos internos del microcontrolador y que no estaban contempladas en INTCON.
Ocupa la dirección 8Ch y para que cumplan su función, los bits de PIE1, es necesario que el PIE sea igual a 1 en INTCON, 6. El bit PSPIE solo es válido solo es válido en los modelos de 40 patas, manteniéndose a 0 en los de 28 patas.

PSPIE	ADIE	RCIE	TXIE	SPIE	CCP1IE	TMR2IE	TMR1IE

PSPIE: Permiso de interrupción para la puerta paralela esclava al realizar una operación de lectura/escritura. En modelos de 40 patas.

ADIE: Permiso de interrupción para el conversor A/D al finalizar la conversión.

RCIE: Permiso de la interrupción para el receptor de USART cuando el buffer se llena.

TXIE: Permiso de la interrupción para el transmisor de USART cuando el buffer se vacía.

SSPIE: Permiso de interrupción para la puerta serie síncrona.

CCP1IE: Permiso de interrupción para el módulo CCP1 cuando se produce una captura o comparación.

TMR2IE: Permiso de interrupción para el TMR2 con su desbordamiento.

TMR1IE: permiso de interrupción para el TMR1 con su desbordamiento.

3. REGISTRO DE PERMISO DE INTERRUPCIONES 2 (PIE2)

Contiene los bits de permiso de interrupción de las tres causas que no figuraban en el PIE1. La de fin de escritura de la EEPROM, colisión de bus en el modo de SSP y producción de una captura o comparación en el módulo CCP2. El bit 6 es un bit reservado y su valor es siempre 0. Cuando se leen los bit que no tienen asignada función, se obtiene 0.

-	0	-	EEIE	BCLIE	-	-	CCP2IE

EEIE: Permiso de interrupción por fin de escritura en la EEPROM de datos.

BCLIE: Permiso de interrupción por colisión de bus en el SSP cuando dos o más maestros tratan de transferir al mismo tiempo.

CCP2IE: Permiso de interrupción el módulo CCP2.

- ***REGISTRO DE LOS SEÑALIZADORES DE INTERRUPCION 1 Y 2 (PIR1 y PIR2)***

  En correspondencia con los bits de permiso/prohibición de las causas de interrupción recogidas en el registro PIE1 y PIE2, existen otros dos registros, el PIR1 y PIR2, cuyos bits actúan de señalizadores del momento en el que se origina la causa que provoca la interrupción, independientemente de si está permitida o prohibida. Ocupan las direcciones 0Ch y 0Dh.

REGISTRO PIR1

PSPIF	ADIF	RCIF	TXIF	SSPIF	CCPIF	TMR2IF	TMR1IF

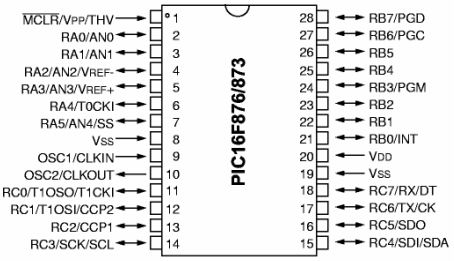
REGISTRO PIR2

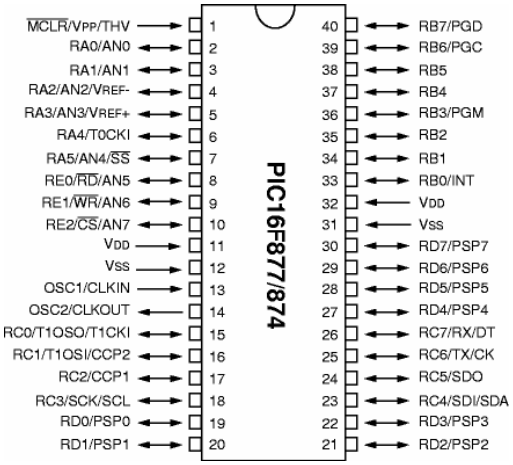
-	0	-	EEIF	BCLIF	-	-	CCP2IF

➢ DISTRIBUCION MEMORIA RAM

Banco 0		Banco 1		Banco 2		Banco 3	
INDF	00h	INDF	80h	INDF	100h	INDF	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD	08h	TRISD	88h		108h		188h
PORTE	09h	TRISE	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reservado	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reservado	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	Propósito	117h	Propósito	197h
RCSTA	18h	TXSTA	98h	General	118h	General	198h
TXREG	19h	SPBRG	99h	16 Bytes	119h	16 Bytes	199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
Registros Propósito General 96 Bytes	20h	Registros Propósito General 80 Bytes	A0h	Registros Propósito General 80 Bytes	120h	Registros Propósito General 80 Bytes	1A0h
	7Fh	accesses 70h-7Fh	EFh F0h FFh	accesses 70h-7Fh	16Fh 170h 17Fh	accesses 70h - 7Fh	1EFh 1F0h 1FFh

LECTURA Y ESCRITURA EEPROM Y FLASH





➢ *LECTURA Y ESCRITURA DE LAS MEMORIAS EEPROM Y FLASH*

En el PIC 16F84 se podía leer y escribir la memoria de datos EEPROM. En los PIC 16F87X también se puede leer y escribir la memoria de código FLASH. Esto significa que un programa dinámicamente puede generar información que se puede grabar en la FLASH directamente sin necesidad de grabador externo.

Para manejar la memoria EEPROM de 64 bytes del PIC 16F84, bastaban 2 registros, para proporcionar la dirección de la memoria a consultar y para grabar el dato de 8 bits, sin embargo, en los PIC 16F87X, no basta con un solo registro para proporcionar la dirección de memoria, ya que esta alcanza los 13 bits, y lo mismo sucede con el dato, que a su vez alcanza los 14bits. Para cubrir esta necesidad el registro EEADR se concatena con el EEADRH, que contiene los 5 bits de más peso de la dirección. Por otra parte, el registro EEDATAH se concatena con EEDATA y tiene los 6 bits de más peso de la palabra leída o a escribir en la FLASH.

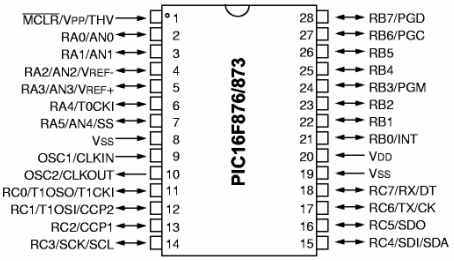
Para controlar la operación lectura/escritura de las memorias EEPROM y FLASH hay dos registros denominados EECON1 Y EECON2. El EECON2, no esta implementado físicamente y sólo se utiliza en la delicada operación de escritura, que tiene la elevada duración de 2 milisegundos. Antes de iniciar la escritura de una palabra se escribe en EECON2 primero el dato 55h y luego el Aah.

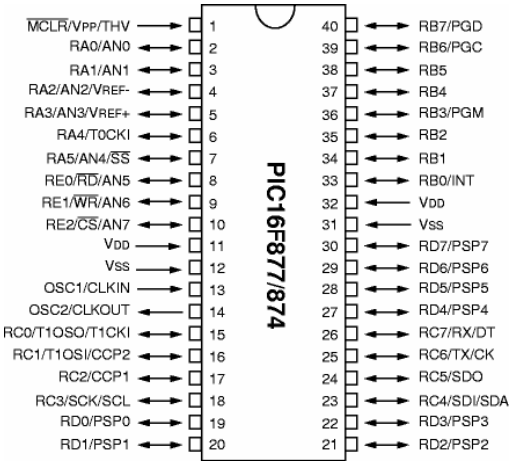
Para evitar escrituras indeseadas en la EEPROM, se controla el bit WREN, prohibiendo cualquier operación de escritura mientras duran los 72 milisegundos que temporiza el Timer de Power-Up. Para realizar la misma protección en la memoria FLASH, se debe poner a 0 el bit WRT de la Palabra de Configuración, que solo puede escribirse desde un grabador externo.

Dependiendo del valor y los bits de protección de código CP1 y CP0, ubicados en la palabra de configuración, se consiguen varias alternativas de protección contra lectura y escritura de la FLASH. A continuación:

CONFIGURACION DE BIT			POSICIONES DE FLASH	LECTURA INTERNA	ESCRIT. INTERNA	LECTURA ICSP	ESCRIT. ICSP
CP1	CP2	WRT					
0	0	X	Memoria de prog.	Sí	No	No	No
0	1	0	Áreas no protegidas	Sí	No	Sí	No
0	1	0	Áreas protegidas	Sí	No	No	No
0	1	1	Áreas no protegidas	Sí	Sí	Sí	No
0	1	1	Áreas protegidas	Sí	No	No	No
1	0	0	Áreas no protegidas	Sí	No	Sí	No
1	0	0	Áreas protegidas	Sí	No	No	No
1	0	1	Áreas no protegidas	Sí	Sí	Sí	No
1	0	1	Áreas protegidas	Sí	No	No	No
1	1	0	Memoria de prog.	Sí	No	Sí	Sí
1	1	1	Memoria de prog.	Sí	Sí	Sí	Sí

PUERTAS E/S


PIC16F876/873
MCLR/VPP/THV 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
VSS 8
OSC1/CLKIN 9
OSC2/CLKOUT 10
RC0/T1OSO/T1CKI 11
RC1/T1OSI/CCP2 12
RC2/CCP1 13
RC3/SCK/SCL 14
28 RB7/PGD
27 RB6/PGC
26 RB5
25 RB4
24 RB3/PGM
23 RB2
22 RB1
21 RB0/INT
20 VDD
19 VSS
18 RC7/RX/DT
17 RC6/TX/CK
16 RC5/SDO
15 RC4/SDI/SDA


PIC16F877/874
MCLR/VPP/THV 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7/PGD
39 RB6/PGC
38 RB5
37 RB4
36 RB3/PGM
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2

- ***Puertas de E/S***

Los microcontroladores PIC 16F87X encapsulados con 28 patas, disponen de 3 puertas de E/S (A,B,C) mientras que los de 40 patas, alcanzan 5 (A,B,C,D,E).

1. PUERTA A

Solo dispone de 6 líneas (RA0-RA5), son bidireccionales y se configuran a través del registro TRISA, situado en el Banco 1. En cada bit del registro TRISA de la Puerta se configura la correspondiente línea. Si el bit es 0, la línea está configura como salida, a su vez, si se pone a 1, la línea se configura como entrada.

El registro PORTA, es el de la Puerta A, que recoge el estado de cada línea de la Puerta, independientemente de cómo estén configuradas.

Las líneas R0/AN0, R1/AN1 y R2/AN2, además de líneas de E/S digitales, también pueden actuar como los canales 0, 1 y 2 por los que se puede aplicar una señal analógica al conversor A/D. La pata RA3/AN3/Vref+, también puede actuar como entrada de Tensión de Referencia para los periféricos que la precisan. La pata RA4/TOCKI actúa además de E/S digital, como entrada de señal de reloj para el Timer 0. La pata RA5/AN5/SS# tiene multiplexadas tres funciones: E/S digital, canal 4 para el conversor A/D y selección del modo esclavo cuando se trabaja con la comunicación serie síncrona.

Para seleccionar si las líneas de la Puerta A van a trabajar como E/S digitales o como canales de entrada para el conversor A/D, hay que escribir el valor adecuado sobre el registro ADCON1. Si se carga en dicho registro el valor 011x en sus 4 bits de menos peso, todas las líneas de puertas funcionan como E/S digitales.

DIREC-CION	*NOMBRE*	*BIT 7*	*BIT 6*	*BIT 5*	*BIT 4*	*BIT 3*	*BIT 2*	*BIT 1*	*BIT 0*	*VALOR EN POR BOR*	*VALOR EN EL RESTO DE RESETS*
05h	PORTA	-	-	RA5	RA4	RA3	RA2	RA1	RA0	-0x 000	-0u 000
85h	TRISA	-	-	Registro de configuración de la Puerta A						-11 1111	-11 1111
9Fh	ADCON1	ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0	-0- 0000	-0- 0000

x significa desconocido
u significa que no cambia
- significa que no está implementado y se lee como 0

2. PUERTA B

Dispone de 8 líneas bidireccionales cuya función se elige mediante la programación del TRISB. Todas las patas de la puerta B disponen de una resistencia interna de pull-up al positivo de la alimentación. Esta va conectada cuando el bit RBPU# (es bit 7 del registro OPTION), tiene valor 0. La resistencia de pull-up se conecta automáticamente siempre que la línea esté configurada como salida. Cuando se produce un Reset por conexión de la alimentación (POR) se desconectan todas las resistencias de pull-up.

Las líneas RB<7-4> pueden programarse para generar una interrupción cuando una de ellas cambia de estado. Se deben configurar como entradas y el valor que se introduce por ellas se compara con el anterior para si no coinciden generar una interrupción, siempre que lo autorice el bit de permiso situado en el INTCON.

La pata RB0/INT también puede programarse como petición de interrupción externa, el bit de permiso también está ubicado en el INTCON.

DIREC-CION	*NOMBRE*	*BIT 7*	*BIT 6*	*BIT 5*	*BIT 4*	*BIT 3*	*BIT 2*	*BIT 1*	*BIT 0*	*VALOR EN POR BOR*	*VALOR EN EL RESTO DE RESETS*
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h	TRISB	Registro de configuración de la Puerta B								1111 1111	1111 1111
81h	OPTION	RBPU#	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

x significa desconocido
u significa que no cambia
- significa que no está implementado y se lee como 0

3. PUERTA C

Consta de 8 líneas bidireccionales cuyo sentido se configura mediante el registro TRISC. Todas las patas de esta puerta tienen multiplexadas diferentes funciones:

- *RC0/T1OSO/T1CKI:* Esta línea puede actuar como E/S digital, como salida del Timer 1 o como entrada de impulsos para el Timer 1.
- *RC1/T1OSI/CCP2:* E/S digital, entrada al oscilador del Timer 1, entrada del modulo de Captura 2, Salida del comparador 2, salida del PWM2.
- *RC2/CCP1:* E/S digital, entrada Captura 1, Salida de comparador uno, salida del PWM1.
- *RC3/SCK/SCL:* E/S digital, Señal de reloj modo SPI, Señal de reloj modo I2C.
- *RC4/SDI/SDA:* E/S digital, Señal de datos modo SPI, Señal de datos modo I2C.
- *RC5/SDO:* E/S digital, Salida de datos en modo SPI.
- *RC6/TX/CK:* E/S digital, Línea de transmisión en USART, Señal de reloj síncrona en transmisión serie.
- *RC7/RX/DT:* E/S digital, Línea de recepción USART, Línea de datos en transmisión serie síncrona.

4. PUERTA D

Consta de 8 líneas bidireccionales. Solo la tienen los PIC encapsulados con 40 patas (16F877). Se configura mediante el registro TRISD. Todas las patas disponen en su entrada de un Trigger Schmitt.

Además de usarse como líneas de E/S digitales normales, implementan una puerta paralela esclava de 8 líneas (PSP), que sirve para permitir la comunicación en paralelo con otros elementos del sistema.

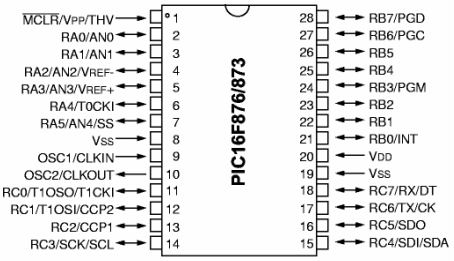
Las patas se denominan RD0/PSP0-RD7/PSP7, y para que funcionen como puerto de comunicación esclava en paralelo, es preciso poner el bit PSP MODE=1.

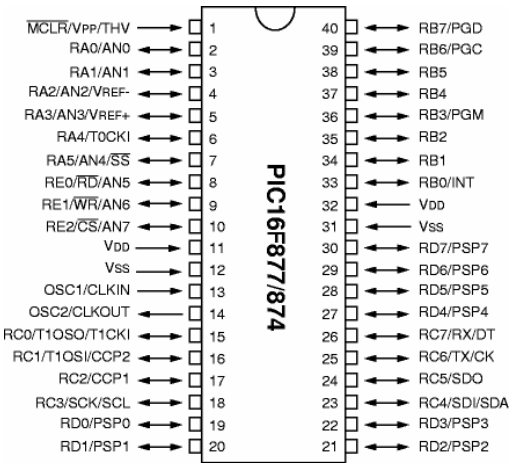
5. PUERTA E

Solo la tienen los PIC de 40 patas. Dispone de tres patas multifunción, que se configuran como entrada o salida, según el valor de los tres bits de menos peso del registro TRISE

- *RE0/RD#/AN5:* E/S digital, Señal de lectura en modo puerta paralela esclava, Canal 5 de conversor A/D.
- *RE1/WR#/AN6:* E/S digital, Señal de escritura en modo PSP, Canal 6 de conversor A/D.
- *RE2/CS#/AN7:* E/S digital, Selección de chip en modo PSP, Canal 7 de conversor A/D.

RECURSOS ESPECIALES





Recursos especiales

- ***PALABRA DE CONFIGURACIÓN***

  Es una posición reservada de la memoria de programa FLASH que ocupa la dirección 2007h y que solo es accesible durante la programación del PIC

CP1	CP0	DEBUG	-	WRT	CPD	LVP	BODEN	CP1	CP0	PWRTE#	WDTE	FOSC1	FOSC0

CP1:CP0 Código de protección de la memoria del programa. Están repetidos en los bits 13:12 y 5:4. Si los bits de código de protección no se programan, las protecciones de la memoria de código pueden ser leídas para verificación

CP1	CP0	Protección desde...	...hasta	Modelo PIC
0	0	0000h	0FFFh	16F873/4
0	0	0000h	1FFFh	16F876/7
0	1	0800h	0FFFh	16F873/4
0	1	1000h	1FFFh	16F876/7
1	0	0F00h	0FFFh	16F873/4
1	0	1F00h	1FFFh	16F876/7
1	1	No hay código protegido en la memoria FLASH		

DEBUG Modo Depurador en Circuito
1 = Desactivado. RB7:RB6 actúan como líneas de E/S.
0 = Activado. RB7:RB6 actúan en modo depurado. La depuración se puede hacer desde el MPLAB.

WRT Permiso de escritura en la memoria FLASH
1 = Se puede escribir en la parte no protegida de la FLASH
0 = Prohibición de escritura

CPD Código de Protección de la Memoria EEPROM de Datos
1 = No hay protección en la EEPROM
0 = Protección del código en la EEPROM

LVP Bit de Permiso para Programación de Bajo Voltaje
1 = RB3/PGM tiene permitida la grabación en bajo voltaje
0 = RB3/PGM funciona como E/S digital. La programación se realiza en alto voltaje.

BODEN Bit de Permiso para el Reset por Caída de Tensión
1 = BOR activada
0 = BOR desactivada

PWRTE# Bit de permiso para el Timer de Conexión de Alimentación
1 = PWRT desactivado
0 = PWRT activado

WDTE Bit de Permiso del Timer de perro guardián
1 = WDT activado
0 = WDT desactivado

FOSC1:0 Tipo de oscilador

FOSC1	FOSC0	Tipo
0	0	LP (Baja Potencia. De 35 a 200 kHz)
0	1	XT (Estándar. De 100 kHz a 4 MHz)
1	0	HS (Alta velocidad. Más de 4 MHz)
1	1	RC (Resistencia-Condensador)

➢ *PALABRA DE IDENTIFICACIÓN*

Se trata de cuatro palabras de la memoria de programa que se hallan comprendidas entre la dirección 2000h y la 2003h y están reservadas para que el usuario las pueda emplear en funciones de comprobaciones o "checksums", códigos de identificación, números de serie, fecha, modelo, lote, números secuenciales o aleatorios, etc. Estas cuatro posiciones solo son accesibles en la lectura y escritura durante la operación de programación/ verificación. Solo se deben emplear los cuatro bits de menos peso de las palabras de identificación.

➢ *REINICIALIZACIÓN O RESET*

Los PIC 16F87X disponen de diversa maneras de reinicializarse.

1º Reset por conexión de alimentación. (POR: Power on Reset) El valor de tensión de alimentación Vdd sube entre 1,2 a 1,7 V.
2º Activación de la pata MCLR#. (nivel bajo en dicha pata durante una operación normal)
3º Activación de la pata MCLR# estando el PIC en reposo (SLEEP)
4º Reset provocado por desbordamiento del Perro Guardián en una operación normal.
5º Reset provocado por el desbordamiento del Perro Guardia durante el estado de reposo (SLEEP)
6º Reset provocado por una caída de voltaje (BOR: Brown out Reset) Vdd baja entre 3,8 y 4,2 V.

Los bits TO# y PD# del registro de Estado toman un valor determinado en cada tipo de reset. También los bits 0 y 1 del registro PCON, llamados BOR# y POR# respectivamente, sirven para especificar las causas de un reset.

POR#	BOR#	TO#	PD#	TIPO DE RESET
0	x	1	1	Conexión de alimentación POR
0	x	0	x	Ilegal
0	x	x	0	Ilegal
1	0	1	1	Por caída de tensión BOR
1	1	0	1	Por WDT (operación normal)
1	1	0	0	Por WDT (modo SLEEP)
1	1	u	u	Activación normal MCLR#
1	1	1	0	MCLR# en SLEEP o interrupción para despertar de SLEEP

Tras un Reset, el contador de programa queda cargado con el valor 000h en todos los casos, menos cuando se produce el desbordamiento del WDT o cuando se despierta del modo SLEEP por una interrupción, en cuyos casos, PC se carga con el valor PC+1.

➢ *PERRO GUARDIAN (WDT: WATCHDOG TIMER)*

El WDT de los PIC16F87X es similar al del 16F84. Se trata de un contador que funciona con los impulsos de su propio oscilador y que provoca un Reset cuando se desborda en funcionamiento normal. Si el desbordamiento se produce cuando el microcontrolador se halla en estado de Reposo, se despierta y sigue su comportamiento normal.

Las instrucciones CLRWDT y SLEEP borran o ponen a cero el valor del WDT y el del Postdivisor. Si se ejecuta la instrucción CLRWDT y el Predivisor de Frecuencia está asignado al perro guardián, se borra, pero no cambia su configuración.

➢ *MODO DE REPOSO O BAJO CONSUMO*

Este funcionamiento se ejecuta con la instrucción SLEEP, igual que con el 16F84. Esta manera de trabajo se caracteriza por su bajo consumo, las líneas de E/S que se utilizaban mantienen su estado, las que no se empleaban reducen al mínimo su consumo, se detienen los temporizadores y tampoco opera el conversor A/D.

Al entrar en modo de reposo, si estaba funcionando, el WDT se borra, pero sigue trabajando. Existen varias formas de despertar del modo SLEEP, y seguir ejecutando la instrucción PC+1:

1º- Activación externa de la pata MCLR#.
2º- Desbordamiento del WDT, que sigue trabajando en reposo.
3º- Generación de interrupción por activación de la pata RB0/INT, o por cambio de estado en las cuatro patas de menos peso de la Puerta B.
4º- Interrupción originada por alguno de los nuevos periféricos de los PIC 16F87X tales como:
- *a) Lectura o escritura en la puerta paralela PSP.*
- *b) Interrupción del Timer 1.*
- *c) Interrupción del módulo CCP en modo captura.*
- *d) Disparo especial de Timer 1 funcionando en modo asíncrono con reloj externo.*
- *e) Interrupción en el módulo de comunicación SSP (Start/Stop).*
- *f) Transmisión o recepción del MSSP modo esclavo (SPI/I2C).*
- *g) Transmisión o recepción del USART.*
- *h) Fin de la conversión en el conversor A/D.*
- *i) Fin de operación.*
- *j) Fin de escritura sobre EEPROM.*

➢ *PROGRAMACIÓN DE LOS PIC 16F87X*

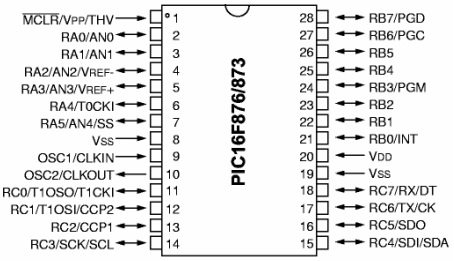
La posibilidad de programar a esta subfamilia de PIC en serie, permite grabar en la memoria de código el programa de trabajo, estando colocado el Pic sobre el circuito o producto de aplicación final. Esta característica permite a los fabricantes construir y montar completamente la tarjeta de circuito impreso y dejar pendiente la grabación del programa.

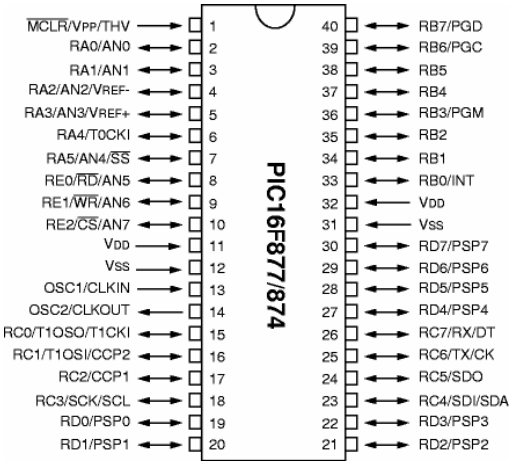
La programación en serie típica, que se realiza con un Voltaje Alto de 12 a 14 V aplicado por la pata MCLR#/Vpp requiere el uso de 5 patas del PIC:

a) VDD = 5V
b) GND o Tierra
c) Vpp = 12 a 14 V que se introducen por la pata MCLR#/Vpp
d) RB6 : Recibe los impulsos de reloj.
e) RB7 : Línea de datos con los bits en serie.

Una gran aportación esta gama de PIC es la programación con Voltaje Bajo (LVP : Low Voltage Programming), que no requiere la tensión de 12 a 14V. Para grabar en este modo hay que poner el bit LVP = 1, que reside en la Palabra de Configuración y la pata RB3/PGM se debe conectar a nivel alto. Entonces por la pata MCLR#/Vpp se aplica la tensión VDD de 5V mientras dura la operación de grabado. Cuando no se opera en este modo de programación se puede usar la pata RB3 como una línea de E/S digital.

TEMPORIZADORES





➢ *TIPOS Y CARACTERÍSTICAS GENERALES*

En la familia de los PIC 16F87X, disponen de tres temporizadores. El TMR0, el TMR1 y el TMR2.

En TMR0 es idéntico al del 16F84, sus funciones más representativas son:

1º TMR0 es un Contador/Temporizador de 8 bits.
2º Leíble y escribible.
3º Reloj interno o externo.
4º Selección de flanco en el reloj externo.
5º Predivisor de la frecuencia del reloj programable.
6º Generación de interrupción opcional en el desbordamiento.

El TMR1 se caracteriza por:

1º TMR1 es un Contador/Temporizador de 16 bits.
2º Leíble y escribible.
3º Selección de reloj interno o externo.
4º Interrupción opcional por desbordamiento de FFFFh a 0000h.
5º Posible reinicialización desde los módulos CCP.

El TMR2 tiene las siguientes características fundamentales:

1º TMR2 es un Temporizador de 8 bits.
2º Dispone de un Registro de Período de 8 bits (PR2)
3º Leíble y escribible.
4º Predivisor de Frecuencia programable.
5º Postdivisor de frecuencia programable.
6º Interrupción opcional al coincidir TMR2 y PR2.
7º Posibilidad de generar impulsos al modo SSP.

➢ *ESTRUCTURA INTERNA Y FUNCIONAMIENTO DEL TMR1*

El TMR1 es el único Temporizador/Contador ascendente con un tamaño de 16bits, lo que requiere el uso de 2 registros concatenados de 8 bits: TMR1h : TMR1L, que son los encargados de guardar el valor del montaje en cada momento, cuando el valor llega hasta FFFFh se activa el señalizador TMR1IF y se regresa al valor inicial 0000h. También si se desea se puede provocar una petición de interrupción.

El valor contenido en TMR1H: TMR1L puede ser leído o escrito y los impulsos que originan el contaje pueden provenir del exterior o de la frecuencia de funcionamiento del microcontrolador:

El TMR1 es capaz de funcionar de 3 formas:

1º Como temporizador
2º Como contador síncrono
3º Como contador asíncrono

➢ *REGISTRO DE CONTROL DEL TMR1 (T1CON)*

-	-	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC#	TMR1CS	TMR1ON

El funcionamiento del TMR1 esta gobernado por el valor con el que se programan los bits del registro T1CON que ocupa la dirección 10h de la memoria RAM.

El bit TMR1ON gobierna el permiso o la prohibición de funcionamiento del TMR1. Este bit es activo a nivel alto.

El bit TMR1C selecciona la fuente de los impulsos de contage. Si vale 0 elige el reloj interno y si vale 1 elige el reloj externo que se aplica por las patas RC0 y RC1.

Cuando los impulsos dependen de un reloj externo es preciso que el bit T1OSCEN tenga el valor 1, en cuyo caso las patas RC0 y RC1 actúan como entradas del Oscilador externo. Si T1OSCEN vale = 0 los impulsos vendrán a través de RC0.En ambos casos, el TIMER1 funciona como contados de elementos externos.

El predivisor de frecuencia (preescaler) es un simple divisor de la frecuencia de los impulsos que se aplican al TMR1 por 1, 2, 4 u 8. El rango de división lo eligen los bits T1CKPS1 y T1CKS0:

T2CKPS1	T2CKPS0	RANGO DEL PREDIVISOR
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

El bit T1SYNC# determina la posible sincronización o no de los impulsos del reloj externo con los del reloj interno, según valga 0 o 1 respectivamente.

El TMR1 Puede generar una petición de interrupción cuando se produce el sobrepasamiento del contaje. En esta situación se pone automáticamente a 1 el flag TMR1F, que es el bit 0 del registro especifico PIR1. El permiso de la prohibición de interrupción del TMR1 está controlada por el bit TMR1IE del PIE1.

Cuando el modulo de CCP está configurado como comparador para generar un "disparo especial", dicha señal resetea el TMR1. Para aprovechar esta característica el TMR1 debe estar configurado en modo temporizador o contador síncrono. En otro caso no se produce el Reset.

➢ *FUNCIONAMIENTO Y PROGRAMACION DEL TMR2*

Se trata de un temporizador ascendente de 8 bits, que leer y escribir, y que también puede realizar funciones especiales para la puerta serie síncrona (SPP) y para los módulos de captura y comparación (CCP)

La señal de TMR2 es interna con valor de Fosc/4, y antes de ser aplicada pasa por un predivisor de frecuencia con rangos de 1:1, 1:4, 1:16. La salida pasa por un postdivisor con rangos de 1:1 a 1:16.

Al entrar el microcontrolador en SLEEP, el TMR2 deja de funcionar porque no existe Fosc al pararse el oscilador.

Para controlar el funcionamiento de TMR2 se utiliza el registro T2CON.

-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2SCKPSI	T2SCKPSO

Los bits 1 y 0 del T2CON sirven para seleccionar el rango de división del predivisor de impulsos de la siguiente forma:

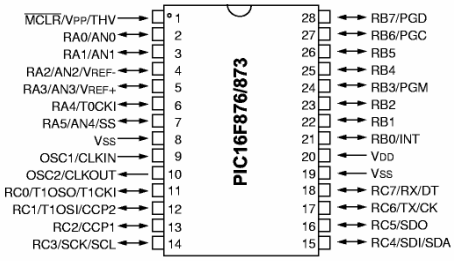
T2CKPS1	T2CKPS0	RANGO DEL PREDIVISOR
0	0	1:1
0	1	1:4
1	x	1:16

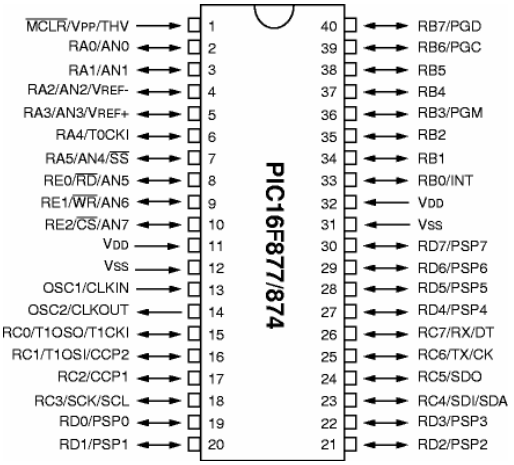
El bit TMR2CON sirve para permitir o prohibir el funcionamiento del TMR2. El bit de más peso no es significativo y los cuatro bits restantes determinan el rango por el que divide la frecuencia el postdivisor:

TOUTPS3-TOUTPS0	RANGO DEL POSTDIVISOR
0000	1:1
0001	1:2
0010	1:3
....	....
1111	1:16

El señalizador de desbordamiento del TMR2 es el bit 1 del registro PIR1. El predivisor y el postdivisor se ponen a 0 al escribir el T2CON o con un Reset. Sin embargo, al escribir en T2CON no se borra el TMR2, pasa a valer 0 al hacer un Reset. El TMR2 tiene asociado un registro de periodo PR2. Cuando el valor del contaje del TMR2 coincide con el valor cargado en PR2 se genera un impulso en la salida EQ y se resetea el TMR2.

CAPTURA, COMPARACIÓN Y MODULACIÓN DE ANCHURA DE PULSOS





➢ *INTRODUCCIÓN A LOS MÓDULOS CCP*

Los microcontroladores PIC 16F87X disponen de dos módulos CCP, llamados CCP1 y CCP2, que solo se diferencian en el "Disparo Especial". Realizan tres funciones especiales:

1º Modo captura: una pareja de registros de un módulo CCPx captura el valor que tiene el TMR1 cuando ocurre un evento especial en la pata RC2/CCP1 o en la RC1/T1OSCI/CCP2.

2º Modo comparación: se compara el valor de 16 bits del TMR1 con otro valor cargado en una pareja de registros de un módulo CCPx y cuando coinciden se produce un evento en la pata RC2/CCP1 o en la RC1/T1OSCI/CCP2.

3º Modo modulación de anchura de pulsos (PWM): dentro del intervalo del periodo de un impulso controla la señal en que la salida vale nivel alto.

El módulo CCP1 utiliza un registro de trabajo de 16 bits, que está formado con la concatenación de los registros CCPR1H, CCPR1L. El registro de control del módulo CCP1 es el CCP1CON. El modulo CCP2 tiene como registros de trabajo a CCPR2H-CCPR2L y como registro de control a CCP2CON. Las parejas de registros son las encargadas de capturar el valor del TMR1, de comparar el valor que tiene con el TMR1 o, en el PWM, de modular la anchura del impulso.

REGISTRO CCPxCON (x puede ser 1 0 2)

-	-	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0

- ***MODO CAPTURA***

La pareja CCPxH-L del módulo CCPx captura el valor de 16 bits que contiene el Timer1 cuando sucede un evento en la pata RC/CCPx de la Puerta C, que previamente ha sido configurada como entrada poniendo a 1 el bit del registro TRISC.

Los eventos que pueden ocurrir sobre la pata RCy/CCPx para producir la captura del valor del TMR1 sobre la pareja de registros CCPxH-L son:

1º Un flanco ascendente
2º Un flanco descendente
3º Cada 4 flancos ascendentes
4º Cada 16 flancos ascendentes

Los 4 bits CCP1M3-0 del registro CCP1CON seleccionan el evento adecuado en el módulo CCP1 y o a su vez en el CCP2.

Al efectuar la captura se activa el señalizador CCP1IF en el registro PIR1. Además, si se pone a 1 el permiso de interrupción PIE1 <CCP1IE>, se genera una petición de interrupción cuando se carga CCPR1H-L el valor del Timer1.

Cuando se emplea el módulo CCP1 en modo Captura, el Timer1 debe estar configurado para trabajar como temporizador o como contador síncrono. Nunca en modo asíncrono.

Si se fueran a cambiar las configuraciones del modulo de captura, convendría detener o desactiva este antes para así evitar que se produzcan falsas interrupciones durante la operación.

Cuando se desactiva el módulo CCP o deja de funcionar en modo captura se borra la codificación del predivisor de frecuencia que determinan los bits CCP1M3-0.

Una aplicación muy interesante del modo captura puede ser la medición de los intervalos de tiempo que existen entre los impulsos que llegan a la pata RC2/CCP1, que se halla configurada como entrada. El TMR1 debe trabajar como entrada de reloj externo sincronizada.

➢ *MODO COMPARACIÓN*

En esta forma de trabajo, la pareja de registros CCPR1H-L compara su contenido de forma continua, con el valor del TMR1. Cuando coinciden ambos valores, la Pata Rc2/CCP1, que se halla configurada como salida, le acontece uno de los siguientes eventos de acuerdo con la programación del los bits CCP1M3-0:

1º Pasa a nivel alto
2º Pasa a nivel bajo
3º No cambia su estado pero se produce una interrupción.

Al coincidir los valores TMR1 con la pareja de registros CCPR1H-L se pone a 1 el señalizador CCP1IF.

El TMR1 debe trabajar en modo temporizador o contador síncrono, nunca en modo asíncrono.

Si el bit de permiso de interrupción está a 1, cuando coinciden los valores mencionados, se origina una petición de interrupción.

Si con los bits CCP1M3-0 se selecciona el modo de trabajado de "Disparo especial", el módulo CCP1 pone a 0 el TMR1 y el CCPR1 funciona como un registro de periodo capaz de provocar periódicamente interrupciones. En ese modo de disparo especial, el CCP2 se pone a 0 y el TMR1 y, además, inicia una conversión en el conversor A/D, con lo que también y con carácter periódico, pueden realizarse conversiones A/D sin el control del programa de instrucciones.

➢ *MODO DE MODULACIÓN DE ANCHURA DE PULSOS (PWM)*

Con este modo de trabajo, se consiguen impulsos lógicos cuya anchura del nivel alto es de duración variable, que son de enorme aplicación en el control de dispositivos tan populares como los motores y los triacs.

La pata RC2/CCP1 está configurada como salida y bascula entre los niveles lógicos 0 y 1 a intervalos variables de tiempo. Lo que se intenta es obtener un pulso cuyo nivel alto tenga una anchura variable dentro del intervalo del periodo de trabajo.

Cuando se trabaja con una precisión de 10 bits, los 2 bits CCP1CON <5:4> se concatenan con los 8 de CCPR1L y, de la misma forma, los 8 bits de más peso del TMR2 se concatenan con los 2 bits de menos peso del reloj interno.

El tiempo que dura el período de la onda depende del valor cargado en PR2, según la fórmula siguiente:

$$\text{Período} = [(PR2) + 1] \cdot 4 \cdot Tos \cdot \text{Valor Predivisor TMR2}$$

Cuando el valor del TMR2 coincide con el del PR2 suceden 3 acontecimientos:

1º Se borra el TMR2
2º La pata RC2/CCP1, se pone a 1.
3º El valor de CCPR1L, se carga en CCPR1H

El tiempo que la pata de salida está a nivel alto, que es la anchura del impuso, depende del contenido cargado en CCP1R y de los 2 bits del CCP1CON <5:4>, cuando se trabaja con una precisión de 10 bits.

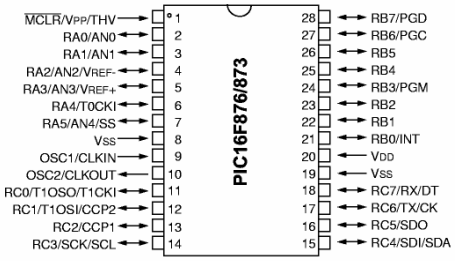
$$\text{Anchura de impulsos} = (CCPR1L{:}CCP1CON\langle 5{:}4\rangle) \cdot Tosc \cdot \text{Valor Predivisor TMR2}$$

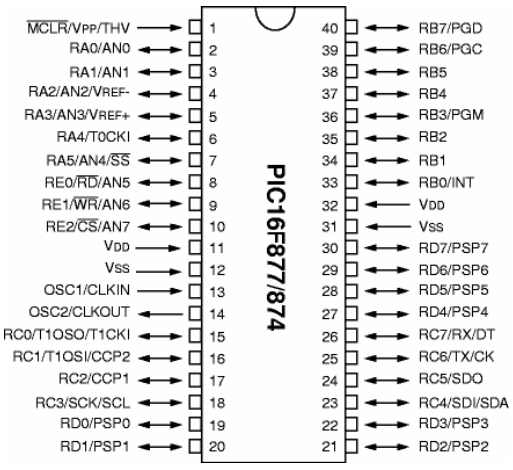
El valor CCPR1:CCP1CON<5:4> puede cargarse en cualquier momento, puesto que el mismo no se traspasa a CCPR1H y se compara hasta que coinciden PR2 con TMR2. En el modo PWM el registro CCPR1L solo puede ser leído.

Los pasos a seguir para realizar la configuración del modo PWM son los siguientes:

1º Asignar el periodo cargando el oportuno valor en PR2.
2º Asignar la anchura del pulso cargando el registro CCP1R y CCP1CON<5:4>
3º Configurar la línea RC2/CCP1 como salida.
4º Asignar el valor del predivisor y activar el TMR2 escribiendo el T2CON.
5º Configurar el Módulo CCP1 en modo PWM

EL CONVERSOR A/D





➢ *PRESENTACIÓN DEL CONVERSOR ANALÓGICO / DIGITAL*

Los microcontroladores PIC 16F87x poseen un conversor A/D de 10 bits de resolución y 5 canales de entrada en los modelos con 28 patas y 8 canales para los de 40 patas.

La resolución que tiene cada bit procedente de la conversión tiene un valor que es función de la tensión de referencia de acuerdo con la siguiente fórmula:

Resolución = (Vref+ - Vref-) / 1.024 = Vref / 1.024

Por ejemplo, si Vref+ es 5V y Vref- es 0V, la resolución será de 4,8 mV por bit. La tensión de referencia determina los límites máximo y mínimo de la tensión analógica que se puede convertir. El voltaje diferencial mínimo es de 2V

A través del canal de entrada seleccionado, se aplica la señal analógica a un condensador de captura y mantenimiento y luego se introduce al conversor, el cual proporciona un resultado digital de 10 bits de longitud usando la técnica de aproximaciones sucesivas.

El conversor A/D es el único dispositivo que puede funcionar en reposo, para ello el reloj del conversor deberá conectarse al oscilador RC interno.

➢ *REGISTROS DE TRABAJO*

El funcionamiento del conversor A/D requiere la manipulación de 4 registros:

1º ADRESH: Parte alta del resultado de la conversión.
2º ADRESL: Parte baja del resultado de la conversión.
3º ADCON0: Registro de control 0.
4º ADCON1: Registro de control 1.

En la pareja ADRESH:ADRESL, se deposita el resultado de la conversión, que al estar compuesta por 10 bits, solo son significativos 10 de los bits de dicha pareja.

El registro ADCON0 controla la operación del C A/D, mientras el ADCON1 sirve para configurar las patas de la Puerta como entradas analógica o E/S digitales.

REGISTRO ADCON0

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE#	-	ADON

REGISTRO ADCON1

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0

Los bits ACON<7:6> sirven para seleccionar la frecuencia reloj que se emplea en la conversión, con la siguiente asignación:

ADCS1:0	FRECUENCIA
00	Fosc/2
01	Fosc/8
10	Fosc/32
11	FRC (Procede del oscilador RC interno)

Se designa como TAD el tiempo que dura la conversión de cada bit en el caso de trabajar con valores digitales de 10 bits. Se requiere un tiempo mínimo de 12·TAD. El valor de TAD se selecciona por software mediante los bits ADCS1:ADCS0 y en los PIC 16F87x nunca debe ser menor de 1,6 microsegundos.

ADCS1:0	TAD
00	2 · Tosc
01	8 · Tosc
10	32 · Tosc
11	Oscilador RC interno en el C A/D

Los bits CHS2-0 seleccionan el canal por el que se introduce la señal a convertir, de acuerdo con el siguiente código:

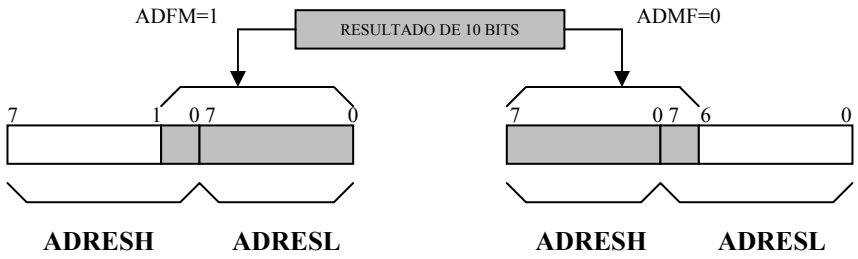
CHS2-0	CANAL
000	Canal 0 (RA0/AN0)
001	Canal 1 (RA1/AN1)
010	Canal 2 (RA2/AN2)
011	Canal 3 (RA3/AN3)
100	Canal 4 (RA5/AN4)
101	Canal 5 (RE0/AN5). Los PIC de 28 patas no la tienen
110	Canal 6 (RE1/AN6). Los PIC de 28 patas no la tienen
111	Canal 7 (RE2/AN7). Los PIC de 28 patas no la tienen

El bit GO/DONE# es el “bit de estado de la conversión”. Poniéndolo a 1 se inicia la conversión y mientras esté a 1 está realizándose la conversión. Cuando GO/DONE# pasa a 0 confirma el final de la conversión y la puesta del resultado en la pareja de registros ADRESH:L.

El bit ADON sirve para activar el C A/D poniéndolo a 1 y para desactivar su funcionamiento, poniéndolo a 0.

➢ *ESTRUCTURA INTERNA Y CONFIGURACIÓN DEL C A/D*

El bit de menos peso (ADFM) del registro ADCON1 selecciona el formato del resultado de la conversión. Si vale 1, el resultado está justificado en el registro ADRESH, que tiene sus 6 bits de más peso a 0; mientras que si vale 0 la justificación se hace sobre el registro ADRESL, que tiene sus 6 bits de menos peso a 0. Esto significa que los 16 bits que forman la unión de los dos registros, unas veces tienen a 0 los 6 bits de más peso y otras los 6 bits de menos peso.



Los restantes 4 bits (PCFG3-0) de ADCON1 se usan para configurar las patitas de los canales de entrada al conversor como analógicas o como E/S digitales, de acuerdo con la siguiente tabla:

PCFG3-0	AN7/ RE2	AN6/ RE1	AN5/ RE0	AN4/ RA5	AN3/ RA3	AN2/ RA2	AN1/ RA1	AN0/ RA0	V_{REF+}	V_{REF-}	CHAN/ REFS
0000	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}	8/0
0001	A	A	A	A	V_{REF+}	A	A	A	RA3	V_{SS}	7/1
0010	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}	5/0
0011	D	D	D	A	V_{REF+}	A	A	A	RA3	V_{SS}	4/1
0100	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}	3/0
0101	D	D	D	D	V_{REF+}	D	A	A	RA3	V_{SS}	2/1
011x	D	D	D	D	D	D	D	D	V_{DD}	V_{SS}	0/0
1000	A	A	A	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}	6/0
1010	D	D	A	A	V_{REF+}	A	A	A	RA3	V_{SS}	5/1
1011	D	D	A	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	4/2
1100	D	D	D	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2	3/2
1101	D	D	D	D	V_{REF+}	V_{REF-}	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}	1/0
1111	D	D	D	D	V_{REF+}	V_{REF-}	D	A	RA3	RA2	1/2

➢ Pasos a seguir para realizar una conversión con el módulo C A/D:

1. Configurar el módulo C A/D

   - Configurar las patas que actuarán como entradas analógicas, las que trabajan como E/S digitales y las usadas para la tensión de referencia (ADCON1).
   - Seleccionar el reloj de la conversión (ADCON0)
   - Seleccionar el canal de entrada A/D (ADCON0)
   - Activar el módulo A/D (ADCON0)

2. Activar, si desea, la interrupción escribiendo sobre PIE1 y PIR1

   - Borrar el señalizador ADIF.
   - Poner a 1 el bit ADIE
   - Poner a 1 los bits habilitadores GIE y PIE

3. Tiempo de espera para que transcurra el tiempo de adquisición

4. Inicio de la conversión

   - Poner a 1 el bit GO/DONE# (ADCON0)

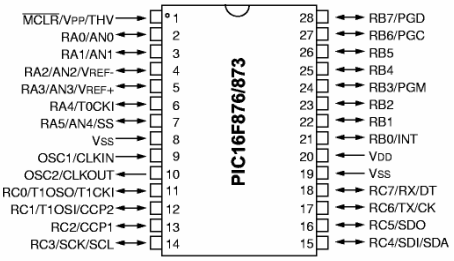
5. Tiempo de espera para completar la conversión A/D que puede detectarse

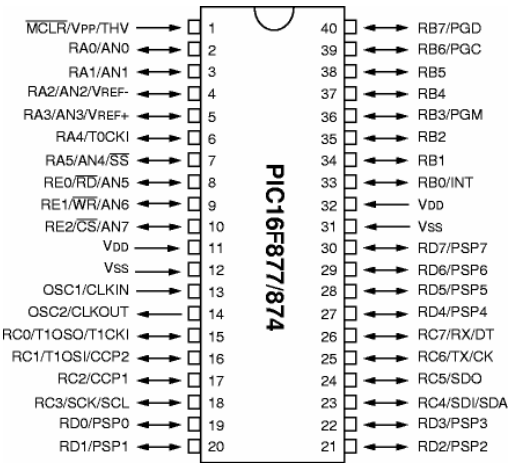
   - Por la exploración del bit GO/DONE#, que al completarse la conversión pasa a valer 0.
   - Esperando a que se produzca la interrupción si se ha programado, al finalizar la conversión.
   - Aunque no se permita interrupción, el señalizador ADIF se pondrá a 1 al finalizar la conversión.

6. Leer el resultado de los 10 bits válidos de ADRSH:L y borrar el flag ADIF

7. Para una nueva conversión regresar al paso 1º o al 2º. El tiempo de conversión por bit está definido por T_{AD}. Se exige esperar un mínimo de $2 \cdot T_{AD}$ para reiniciar una nueva conversión

MÓDULO DE COMUNICACIONES SERIE SÍNCRONA MSSP





➢ *INTRODUCCIÓN*

La comunicación serie es una forma muy apreciada de transferir datos digitales entre sistemas y circuitos integrados, dada la reducida cantidad de líneas que precisa.

En los PIC16F87X se ha implantado el módulo MSSP (*Master Synchronous Serial Port*), que proporciona una excelente interfaz de comunicación de los microcontroladores con otros microcontroladores y diversos periféricos, entre los que destacan la memoria EEPROM serie, los conversores A/D, los controladores de displays, etc.

Además el módulo MSSP admite dos de las alternativas más usadas en la comunicación serie síncrona:

1ª SPI (Serial Peripheral Interface).
2ª I2C (Inter Integrated Circuit).

La comunicación serie en modo SPI la utilizan principalmente las memorias (RAM y EEPROM) y utiliza tres líneas para llevarla a cabo. En el modo I2C sólo se emplean dos líneas y, recientemente, ha conseguido una importante implantación en la comunicación de circuitos integrados, existiendo en el mercado todo tipo de periféricos capaces de trabajar con este protocolo.

El módulo MSSP consta básicamente de dos registros: el SSPSR, que es un registro de desplazamiento que transforma la información serie en paralelo y viceversa, y el registro SSPBUF, que actúa como buffer de la información que se recibe o transmite.

El funcionamiento del módulo MSSP es muy sencillo. En transmisión, el byte que se quiere transmitir se carga en el registro SSPBUF a través del bus de datos interno y automáticamente se pasa al registro SSPSR, que va desplazando bit a bit el dato, sacándolo ordenadamente al exterior al ritmo de los impulsos del reloj. En recepción, los bits van entrando al ritmo del reloj por una pata y se van desplazando en el SSPSR hasta que lo llenan, en cuyo momento la información se traspasa al SSPBUF, donde queda lista para su lectura. Este doble almacenamiento de datos recibidos, permite iniciar la recepción de un nuevo dato antes de que se haya leído el último.

Cuando se han recibido 8 bits durante la recepción en SSPSR, se traspasa dicha información a SSPBUF y entonces el bit señalizador BF (Buffer-Full) se pone a 1, al igual que el flag de interrupción SSPIF. Cualquier escritura en el SSPBUF se ignora durante la transferencia de información y se señaliza poniendo a 1 el bit WCOL. Recae la responsabilidad del programador pasar el bit WCOL a 0 una vez completada la escritura en SSPBUF.

- ***MODO SPI***

Permite la transferencia de datos de 8 bits en serie, que pueden ser transmitidos y recibidos de forma síncrona y simultánea. Para el establecimiento de la comunicación se utilizan tres líneas:

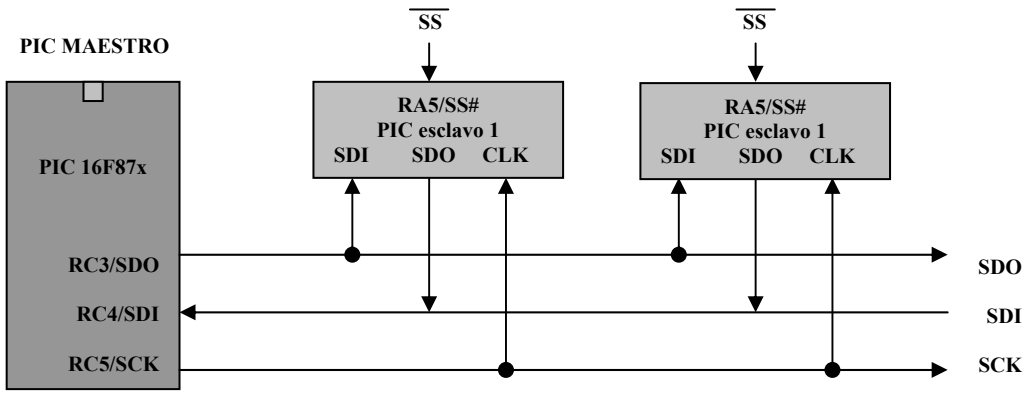
1º SDO (Serial Data Out): Salida de datos en serie.
2º SDI (Serial Data In): Entrada de datos en serie.
3º SCK (Serial Clock): Reloj de sincronización.

Puede ser necesario utilizar una cuarta línea de control más cuando el PIC que se utiliza trabaja en modo esclavo. En este caso, la pata SS# (selección de esclavo) se debe activar a tierra. Las 4 líneas que utilizan se corresponden con las patas multifunción RC3/SDO, RC4/SDI, RC5/SDK y RA5/SS#.

La conexión habitual de PIC maestro se suele realizar con circuitos de memoria con el objeto de ampliar su capacidad. La línea SDO del maestro se corresponde con las SDI de los esclavos y la línea SCK por la que circulan los impulsos de reloj, siempre parten del maestro, que es el encargado de generar y controlar la sincronización.

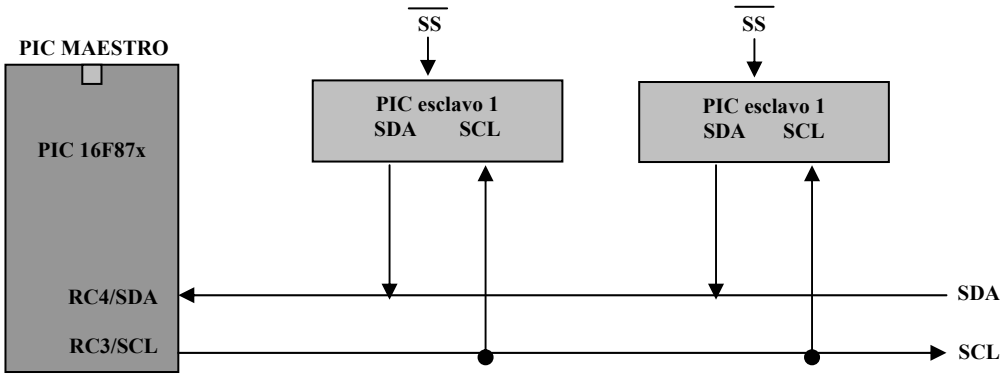
Si, por ejemplo, actuase como esclavo un chip de memoria RAM de 256x8 de tamaño, la comunicación SPI la iniciaría el maestro enviando por la línea SDO un byte con la dirección de la memoria a acceder, seguido de otro byte que especificaría la operación lectura/escritura y un tercero que contendría el dato a escribir en caso de que se tratase de una operación de escritura. En caso de que se tratase de una de lectura, después de enviar los dos bytes iniciales, quedaría a la espera del byte que sacaría el esclavo por su línea SDO y que se introduciría al maestro por su línea SDI. Si fuese una memoria con más posiciones, la dirección se tendría que especificar en más de un byte.

Cuando el PIC trabaja como maestro hay que programar la línea RC3/SDO como salida, la línea RC4/SDI como entrada y la línea RC5/SCK también como salida. Si actuase como esclavo, la línea RC5/SCK debería configurarse como entrada y la RA5/SS# debería conectarse a tierra.



➢ *MODO I^2C*

El protocolo I^2C fue desarrollado por *Philips* para cubrir sus propias necesidades en la implementación de diversos productos electrónicos que requerían una elevada interconexión de circuitos integrados. El protocolo I^2C (Inter-Integrated Circuits) utiliza únicamente dos líneas para la transferencia de información entre los elementos que se acoplan al bus. Una de dichas líneas se dedica a soportar los datos, es bidireccional y se llama SDA; la otra lleva los impulsos de reloj para la sincronización, es unidireccional y recibe el nombre de SCL. Los impulsos de reloj siempre los genera el maestro y tienen la función de sincronizar las transferencias con todos los esclavos colgados a las dos líneas.



Concepto del bus I^2C

Dos líneas, SDA (datos) y SCL (reloj), transportan la información entre los diferentes dispositivos conectados al bus. Cada dispositivo se identifica por una única dirección y puede transmitir o recibir dependiendo de la función que se vaya a realizar. Un controlador de LCD, por ejemplo, sólo recibe mientras que una memoria de tipo RAM puede transmitir o recibir datos en función de que se vaya a leer o a escribir.

Los dispositivos pueden clasificarse en *Maestro* (master o principal) o *Esclavo* (slave o secundario). El *maestro* es el que inicia la transferencia de datos y genera la señal de reloj. Cualquiera de los dispositivos direccionados por un *maestro* se considera *esclavo.*

El I2C es un bus *multi-maestro*; puede haber más de un *maestro* conectado y controlando el bus. Normalmente se trata de microcontroladores o microcomputadores.

Direccionamiento del bus I^2C

- El primer byte que envía el maestro tras la condición de inicio, es el código que determina el esclavo (7 bit de mas peso)
- Código 0000 0000 de llamada general
- Bit de menos peso (R/W#), determina si se realiza una operación de lectura o escritura

- Para activar el bus I^2C, poner el bit SSPEN = 1, que es el bit 5 del registro SSPCON
- Previamente, configurar RC3/SCL y RC4/SDA como entradas

Bits de Control del bus I^2C

1. SSPCON – Registro de control
2. SSPCON2 – Registro de control 2
3. SSPSTAT – Registro de estado
4. SSPBUF – Buffer para los datos
5. SSPSR – Registro de desplazamiento (no accesible)
6. SSPADD – Registro de dirección

SSPCON

WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

- WCOL se pone a 1 si se intenta escribir en SSPBUF en condiciones no válidas.
- SSPOV se pone a 1 si hay desbordamiento en SSPBUF
- SSPEN configuración de la patas RC3/SCL y RC4/SDA
  1 = la puerta serie queda configurada con SCL y SDA
  0 = RC3 y RC4 como entradas y salidas digitales
- CKP para activar el reloj en modo esclavo
- SSPM3-0 selección de la frecuencia del reloj

SSPM3-0	FRECUENCIA DEL RELOJ
0000	Reloj = $F_{OSC}/4$
0001	Reloj = $F_{OSC}/16$
0010	Reloj = $F_{OSC}/61$
0011	Reloj = salida del TMR2/2
1000	Reloj = $(SSPADD + 1) \cdot F_{OSC}$

SSPCON2

GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

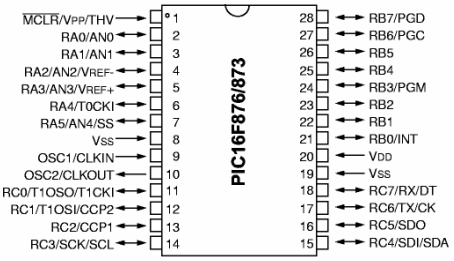
- GCEN solo se usa en modo esclavo
- ACKSTAT si se pone a 1 indica que se ha recibido el bit ACK
- ACKDT bit de reconocimiento de recepción en modo maestro
  1 = el maestro NO ha trasmitido el dato
  0 = el maestro SI ha trasmitido el dato
- ACKEN si vale 1 se inicia la secuencia de generación de la condición de reconocimiento
- RCEN si vale 1 se habilita el modo de recepción del maestro
- PEN si vale 1 se genera la condición de parada de SCL y SDA
- RSEN si vale 1 se inicia la repetición de la condición de inicio
- SEN si vale 1 se inicia la condición de inicio

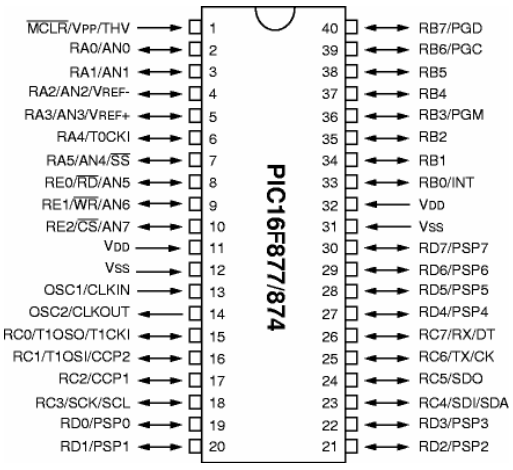
SSPSTAT

SMP	CKE	D/A#	P	S	R/W#	UA	BF

- SMP bit de muestreo
  1 = los bits de datos se muestran al *final* del periodo
  0 = los bits de datos se muestran a *mitad* del periodo
- CKE nivel de SCL y SDA en modo multimaestro
  1 = a nivel alto SCL y SDA en modo maestro y esclavo
  0 = se configuran según especificaciones del I2C
- D/A# indica si el dato recibido es de información o dirección
  1 = de información
  0 = de dirección
- P se pone a 1 cuando detecta la llegada del bit *STOP*
- S se pone a 1 cuando detecta la llegada del bit *START*
- R/W# bit de selección de lectura o escritura
  1 = lectura (READ)
  0 = escritura (WRITE)
- UA tipo de direccionamiento
  1 = dirección de 10 bits
  0 = dirección de 7 bits
- BF señalizador del buffer de datos
  1 = tiene un dato y la transmisión está en proceso
  0 = no tiene ningún dato

USART (SCI)


PIC16F876/873
MCLR/VPP/THV 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
VSS 8
OSC1/CLKIN 9
OSC2/CLKOUT 10
RC0/T1OSO/T1CKI 11
RC1/T1OSI/CCP2 12
RC2/CCP1 13
RC3/SCK/SCL 14
28 RB7/PGD
27 RB6/PGC
26 RB5
25 RB4
24 RB3/PGM
23 RB2
22 RB1
21 RB0/INT
20 VDD
19 VSS
18 RC7/RX/DT
17 RC6/TX/CK
16 RC5/SDO
15 RC4/SDI/SDA


PIC16F877/874
MCLR/VPP/THV 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7/PGD
39 RB6/PGC
38 RB5
37 RB4
36 RB3/PGM
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2

➢ *COMUNICACIÓN SERIE ASÍNCRONA*

Los PIC 16F87x contienen un módulo MSSP con dos puertas para comunicación serie "síncrona", o sea, con señal de reloj. También disponen de un módulo USART, capaz de soportar la comunicación serie síncrona y asíncrona.

El USART, llamado SCI (Serial Comunication Interface), puede funcionar como un sistema de comunicación *full-duplex* o bidireccional asíncrono, adaptándose a multitud de periféricos y dispositivos que transfieren información de esta forma, tales como el monitor CRT o el ordenador PC. También puede trabajar en modo síncrono unidireccional o *half-duplex* para soportar periféricos como memorias, conversores, etc.

Modos de trabajo del USART:

1º. ASÍNCRONA (Full duplex, bidireccional).
2º. SÍNCRONA-MAESTRO (Half duplex, unidireccional).
3º. SÍNCRONA-ESCLAVO (Half duplex, unidireccional).

En el modo asíncrono las transferencias de información se realizan sobre dos líneas TX (transmisión) y RX (recepción), saliendo y entrando los bits por dichas líneas al ritmo de una frecuencia controlada internamente por el USART. En el modo síncrono, la comunicación se realiza sobre dos líneas, la DT que traslada en los dos sentidos los bits a la frecuencia de los impulsos de reloj que salen por la línea CK desde el maestro. En ambos modos, las líneas de comunicación son las dos de más peso de la Puerta C: RC6/TX/CK y RC7/RX/DT.

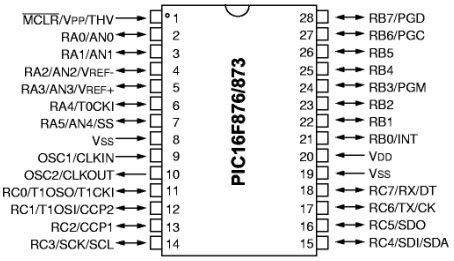
En esta forma de comunicación serie, se usa la norma RS-232-C, donde cada palabra de información o datos se envía independientemente de los demás. Suele constar de 8 o 9 bits y van precedidos por un bit de START (inicio) y detrás de ellos se coloca un bit de STOP (parada), de acuerdo con las normas del formato estándar NRZ (NonReturn-to-Zero). Los bits se transfieren a una frecuencia fija y normalizada.

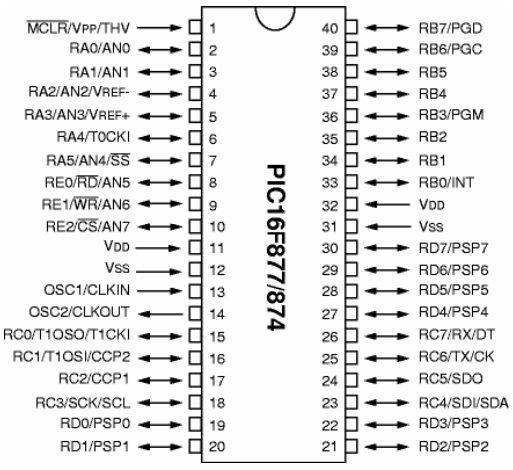
Los cuatro bloques que configuran la arquitectura de USART en modo asíncrono son:

1º Circuito de muestreo.
2º Generador de baudios.
3º Transmisor asíncrono.
4º Receptor asíncrono.

El circuito de muestreo actúa sobre la pata RC7/RX/DT, que es por donde se recibe el bit de información o control y se encarga de muestrear tres veces su valor, para decidir este por mayoría.

PROGRAMAS (ASM)





PROGRAMA 1

Lee el estado de los 6 interruptores del entrenador (RA5-RA0) y reflejar el nivel lógico de los mismos sobre los leds RB5-RB0 conectados a la puerta B

```
            List        p=16F876
            include     "P16F876.INC"

            org         0x05

Inicio      clrf        PORTB       ;Borra el Puerto B
            bsf         STATUS,RP0  ;Selecciona banco 1
            clrf        TRISB       ;Puerta B se configura como salida
            movlw       0x06
            movwf       ADCON1      ;Puerta A configurada como digital
            movlw       b'00111111'
            movwf       TRISA       ;Puerta A se configura como entrada
            bcf         STATUS,RP0  ;Selecciona banco 0

Aqui        movf        PORTA,W     ;Leer las entradas RA0-RA5
            movwf       PORTB       ;Reflejar en las salidas
            goto        Aqui        ;Bucle de lectura

            end
```

PROGRAMA 2

Control de los leds RB0 y RB1 desde el interruptor RA0. RB0 refleja el estado de RA0, RB1 el complemento de RA0

```
                List        p=16F876
                include     "P16F876.INC"


                org         0x05

Inicio          clrf        PORTB       ;Borra el Puerto B
                bsf         STATUS,RP0  ;Selecciona banco 1
                clrf        TRISB       ;Puerta B se configura como salida
                movlw       0x06
                movwf       ADCON1      ;Puerta A como entrada digital
                movlw       b'00011111'
                movwf       TRISA       ;Puerta A se configura como entrada
                bcf         STATUS,RP0  ;Selecciona banco 0

Aqui            btfsc       PORTA,0     ;RA0 = 1 ??
                goto        RA0_es_1    ;Si
                bcf         PORTB,0     ;No, desconecta RB0
                bsf         PORTB,1     ;Conecta RB1
                goto        Aqui        ;Buble

RA0_es_1        bsf         PORTB,0     ;Activa RB0
                bcf         PORTB,1     ;Activa RB1
                goto        Aqui        ;Bucle

                end
```

PROGRAMA 3

Se trata de realizar una rotación secuencial en el encendido de cada led conectados a la puerta B del entrenador. Si RA0 = 0, la rotación será de derecha a izquierda y viceversa. Cada led permanece encendido 0.25 segundos (250 mS)

```
                List        p=16F876        ;Tipo de procesador
                include     "P16F876.INC"   ;Definiciones de registros internos

Contador        equ         0x20            ;Variable para la temporización

                org         0x05
                goto        Inicio

;-------------------------------------------------------------------------------------------------------------
;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. Se basa en repetir 25 veces la temporización
;de 10mS que se empleó en el ejercicio anterior.

Delay           movlw       b'10'
                movwf       Contador        ;Carga el contador con 10
Delay_0         bcf         INTCON,T0IF     ;Desconecta el flag del TMR0

                movlw       b'195'
                movwf       TMR0            ;carga el TMR0 con 195
Delay_1         btfss       INTCON,T0IF     ;Rebosamiento del TMR0 ??
                goto        Delay_1         ;No. Todavía no han pasado los 1* mS
                decfsz      Contador,F      ;Decrementa contador.

                goto        Delay_0         ;Todavía no, temporiza otros 10 ms
                return                      ;Ahora si
;-------------------------------------------------------------------------------------------------------------

Inicio          clrf        PORTB           ;Borra el Puerto B
                bsf         STATUS,RP0      ;Selecciona banco 1
                clrf        TRISB           ;Puerta B se configura como salida
                movlw       0x06
                movwf       ADCON1          ;Puerta A digital
                movlw       b'00011111'
                movwf       TRISA           ;Puerta A se configura como entrada
                movlw       b'00000111'
                movwf       OPTION_REG      ;Preescaler de 256 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 derecha
                goto        Loop

                end
```

PROGRAMA 4

La interrupción del TMR0.

Se trata de comprobar la interrupción provocada por el TMR0. El programa lee el estado de los interruptores conectados a RA0 y RA4 para reflejarlo en los leds conectados a RB0 y RB4 respectivamente. Al mismo tiempo el TMR0 genera una interrupción cada 0.05 seg. (10 mS) que se repetirá 50 veces con objeto de hacer intermitencia de 500 mS sobre el led conectado a RB7.

```
                List        p=16F876        ;Tipo de procesador
                include     "P16F876.INC"   ;Definiciones de registros internos

Contador        equ         0x020           ;Variable para la temporización

                org         0x04            ;Vector de interrupción
                goto        Inter

                org         0x05
                goto        Inicio

Inter           bcf         INTCON,T0IF     ;Repone flag del TMR0
                decfsz      Contador,F      ;Decrementa el contador. Ha habido 50
                                            interrupciones ??
                goto        Seguir          ;No, no han pasado los 500 mS
Con_si_0        movlw       b'50'
                movwf       Contador        ;Repone el contador nuevamente para
                                            ;contar 50 interrupciones
                movlw       b'10000000'
                xorwf       PORTB,F         ;RB0 cambia de estado
Seguir          movlw       b'195'
                movwf       TMR0            ;Repone el TMR0 con 195
                retfie                      ;Retorno de interrupción

Inicio          clrf        PORTB           ;Borra el Puerto B
                bsf         STATUS,RP0      ;Selecciona banco 1
                clrf        TRISB           ;Puerta B se configura como salida
                movlw       0x06
                movwf       ADCON1          ;Puerta A digital
                movlw       b'00111111'
                movwf       TRISA           ;Puerta A se configura como entrada
                movlw       b'00000111'
                movwf       OPTION_REG      ;Preescaler de 256 para el TMR0
                bcf         STATUS,RP0      ;Selecciona banco 0

;El TMR0 se carga con 195. Con un preescaler de 256 y a una frecuencia de 20MHz se obtiene
;una interrupción cada 10mS. Se habilita la interrupción del TMR0.

                movlw       b'195'
                movwf       TMR0            ;Carga el TMR0 con 195
                movlw       b'50'
                movwf       Contador        ;Nº de veces a repetir la interrupción
                movlw       b'10100000'
                movwf       INTCON          ;Activa la interrupción del TMR0
```

```
;Este es el cuerpo principal del programa. Consiste en leer constantemente el estado de RA0 y
;RA1 para visualiza sobre RB0 y RB1.

Loop        btfsc       PORTA,0     ;Testea el estado de RA0
            goto        RA0_ES_1
            bcf         PORTB,0     ;Desactiva RB0
            goto        TEST_RB1
RA0_ES_1    bsf         PORTB,0     ;Activa RB0
TEST_RB1    btfsc       PORTA,1     ;Testea el estado de RA1
            goto        RA1_ES_1
            bcf         PORTB,1     ;Desactiva RB1
            goto        Loop
RA1_ES_1    bsf         PORTB,1     ;Activa RB1
            goto        Loop

            end
```

PROGRAMA 5

La interrupción externa RBO/INT.

Se trata de comprobar la interrupción externa que se aplica a través del pin RBO/INT. El programa principal está en un ciclo cerrado en modo SLEEP (standby de bajo consumo). Cada vez que se detecta un flanco descendente en RB0 se provoca una interrupción cuyo tratamiento hace iluminar las salidas RB7-RB1 durante 1 seg.

```
                List        p=16F876        ;Tipo de procesador
                include     "P16F876.INC"   ;Definiciones de registros internos

Contador        equ         0x20            ;Variable para la temporización

                org         0x04            ;Vector de interrupción
                goto        Inter

                org         0x05
                goto        Inicio

Inter           bcf         INTCON,INTF     ;Repone flag de la interrupción exetrna
                movlw       b'11111110'
                movwf       PORTB           ;Activa las salidas
                movlw       b'100'
                movwf       Contador        ;Inicia contador de temporizaciones de 10
                                            ;ms con 100 (1")
Seguir          bcf         INTCON,T0IF     ;Reponer flag del TMR0
                movlw       b'195'
                movwf       TMR0            ;Repone el TMR0 con 195
Delay_10ms      btfss       INTCON,T0IF     ;Han transcurrido 10 mS ??
                goto        Delay_10ms      ;No, esperar
                decfsz      Contador,F      ;Decrementa el contador.
                goto        Seguir          ;No
                clrf        PORTB           ;Si, han pasado 1", se desconecta la salida
                retfie                      ;Retorno de interrupción

Inicio          clrf        PORTB           ;Borra el Puerto B
                bsf         STATUS,RP0      ;Selecciona banco 1
                movlw       b'00000001'
                movwf       TRISB           ;RB7-RB1 salidas, RB0/INT entrada
                movlw       b'00000111'
                movwf       OPTION_REG      ;Preescaler de 256 para el TMR0
                bcf         STATUS,RP0      ;Selecciona banco 0
                movlw       b'10010000'
                movwf       INTCON          ;Activa la interrupción externa RB0/INT

;Este es el cuerpo del programa principal. Se mantiene en estado SLEEP hasta que se produce
;interrupción

Loop            sleep
                nop
                goto        Loop

                end
```

PROGRAMA 6

El Display de 7 segmentos del entrenador. Decodificador hex. BCD a 7 segmentos.

Mediante los cuatro interruptores RA0-RA3 se introduce un valor hexadecimal de 4 bits que debe visualizarse sobre el display.

```
                List        p=16F876        ;Tipo de procesador
                include     "P16F876.INC"   ;Definiciones de registros internos

                org         0x05
                goto        Inicio

;-------------------------------------------------------------------------------------------------------------
;Tabla: Esta rutina convierte el código binario presente en los 4 bits de menos peso del reg. W en
;su equivalente a 7 segmentos. Para ello el valor de W se suma al valor actual del PC. Se obtiene
;un desplazamiento que apunta al elemento deseado de la tabla.El código 7 segmentos retorna
;también en el reg. W.

Tabla:          addwf       PCL,F           ;Desplazamiento sobre la tabla
                retlw       b'00111111'     ;Dígito 0
                retlw       b'00000110'     ;Dígito 1
                retlw       b'01011011'     ;Dígito 2
                retlw       b'01001111'     ;Dígito 3
                retlw       b'01100110'     ;Dígito 4
                retlw       b'01101101'     ;Dígito 5
                retlw       b'01111101'     ;Dígito 6
                retlw       b'00000111'     ;Dígito 7
                retlw       b'01111111'     ;Dígito 8
                retlw       b'01100111'     ;Dígito 9
                retlw       b'01110111'     ;Dígito A
                retlw       b'01111100'     ;Dígito B
                retlw       b'00111001'     ;Dígito C
                retlw       b'01011110'     ;Dígito D
                retlw       b'01111001'     ;Dígito E
                retlw       b'01110001'     ;Dígito F

Inicio          clrf        PORTB           ;Borra el Puerto B
                bsf         STATUS,RP0      ;Selecciona banco 1
                clrf        TRISB           ;Puerta B se configura como salida
                movlw       0x06
                movwf       ADCON1          ;Puerta A digital
                movlw       b'00111111'
                movwf       TRISA           ;Puerta A se configura como entrada

                bcf         STATUS,RP0      ;Selecciona banco 0

Loop            movf        PORTA,W
                andlw       b'00001111'     ;Lee el código de RA0-RA3
                call        Tabla           ;Convierte a 7 segmentos
                movwf       PORTB           ;Visualiza sobre el display
                goto        Loop

                end
```

PROGRAMA 7

La memoria EEPROM de datos

Se trata de imitar el funcionamiento de las máquinas tipo "SU TURNO" habituales en múltiples comercios. Sobre el display se visualizará el número del turno actual. Este se incrementa a cada pulso aplicado por RA0. En la memoria EEPROM del PIC16F876 se almacena el último número visualizado, de forma que, en caso de haber un fallo de alimentación, se reanude la cuenta en el último número.

Si se parte de que el sistema se emplea por vez primera , se visualiza el 0

```
                List    p=16F876            ;Tipo de procesador
                include "P16F876.INC"       ;Definiciones de registros internos

Contador        equ     0x20                ;Variable para el contador

                org     0x05
                goto    Inicio

;-------------------------------------------------------------------------------------------------------------------
;EE_Write: Graba un byte en la EEPROM de datos. La dirección será la contenida en EEADR y el dato
;se le supone previamente metido en EEDATA

EE_Write        bsf     STATUS,RP0
                bsf     STATUS,RP1          ;Selecciona banco 3
                bcf     EECON1,EEPGD        ;Acceso a EEPROM de datos
                bsf     EECON1,WREN         ;Permiso de escritura
                movlw   b'01010101'
                movwf   EECON2
                movlw   b'10101010'
                movwf   EECON2              ;Secuencia establecida por Microchip
                bsf     EECON1,WR           ;Orden de escritura
Wait            btfsc   EECON1,WR           ;Testear flag de fin de escritura
                goto    Wait
                bcf     EECON1,WREN         ;Desconecta permiso de escritura
                bcf     EECON1,EEIF         ;Reponer flag de fin de escritura
                bcf     STATUS,RP0
                bcf     STATUS,RP1          ;Selecciona banco 0
                return

;-------------------------------------------------------------------------------------------------------------------
;EE_Read: Leer un byte de la EEPROM. Se supone al registro EEADR cargado con la dirección a leer.
;En EEDATA aparecerá el dato leído.

EE_Read         bsf     STATUS,RP0
                bsf     STATUS,RP1          ;Selección de banco 3
                bcf     EECON1,EEPGD        ;Selecciona EEPROM de datos
                bsf     EECON1,RD           ;Orden de lectura
                bcf     STATUS,RP0
                bcf     STATUS,RP0          ;Selección de banco 0
                return
```

```
;-------------------------------------------------------------------------------------------------------------------------
;Tabla: Esta rutina convierte el código BCD presente en los 4 bits de menos peso del reg. W en su
;equivalente a 7 segmentos. El código 7 segmentos retorna también en el reg. W

Tabla:      addwf   PCL,F           ;Desplazamiento sobre la tabla
            retlw   b'00111111'     ;Dígito 0
            retlw   b'00000110'     ;Dígito 1
            retlw   b'01011011'     ;Dígito 2
            retlw   b'01001111'     ;Dígito 3
            retlw   b'01100110'     ;Dígito 4
            retlw   b'01101101'     ;Dígito 5
            retlw   b'01111101'     ;Dígito 6
            retlw   b'00000111'     ;Dígito 7
            retlw   b'01111111'     ;Dígito 8
            retlw   b'01100111'     ;Dígito 9

;-------------------------------------------------------------------------------------------------------------------------
;Delay_10_ms: Esta rutina de temporización tiene por objeto eliminar rebote.

Delay_10_ms:    bcf     INTCON,T0IF ;Desconecta el flag de rebosamiento
                movlw   b'195'
                movwf   TMR0        ;carga el TMR0 con 195
Delay_10_ms_1   btfss   INTCON,T0IF ;Rebasamiento del TMR0 ??
                goto    Delay_10_ms_1 ;Todavía no
                bcf     INTCON,T0IF ;Ahora si, reponer el flag
                return
;-------------------------------------------------------------------------------------------------------------------------

Inicio      clrf    PORTB           ;Borra el Puerto B
            bsf     STATUS,RP0      ;Selecciona banco 1
            clrf    TRISB           ;Puerta B se configura como salida
            movlw   0x06
            movwf   ADCON1          ;Puerta A digital
            movlw   b'00111111'
            movwf   TRISA           ;Puerta A se configura como entrada
            movlw   b'00000111'
            movwf   OPTION_REG      ;Preescaler de 256 para el TMR0
            bcf     STATUS,RP0      ;Selecciona banco 0


            bsf     STATUS,RP1      ;Selecciona banco 2
            clrf    EEADR           ;Selecciona dirección 00 de EEPROM

            call    EE_Read         ;Lee byte de la EEPROM
            bsf     STATUS,RP1      ;Selecciona banco 2
            movlw   0x09
            subwf   EEDATA,W
            btfsc   STATUS,C        ;Mayor de 9 ??
            goto    Ini_0           ;Si, poner a 0 el contador
            goto    Ini_1           ;No
Ini_0       bcf     STATUS,RP1      ;Banco 0
            clrf    Contador        ;Poner a 0 el contador
            goto    Loop
Ini_1       movf    EEDATA,W
            bcf     STATUS,RP1      ;Banco 0
            movwf   Contador        ;Iniciar contador
```

```
Loop        movf    Contador,W
            call    Tabla       ;Convierte contador a 7 segmentos
            movwf   PORTB       ;Visualiza sobre el display

Wait_0      btfss   PORTA,0     ;RA0 está a "1" ??
            goto    Wait_0      ;No, esperar
            call    Delay_10_ms ;Eliminar rebotes

Wait_1      btfsc   PORTA,0     ;RA0 está a "0" ??
            goto    Wait_1      ;No, esperar
            call    Delay_10_ms ;Eliminar rebotes. Ha habido un pulso

            incf    Contador,F  ;Incrementa contador
            movlw   b'10'
            subwf   Contador,W
            btfsc   STATUS,Z    ;Contador mayor de 9 ??
            clrf    Contador    ;Si, vuelta a 00
            movf    Contador,W
            bsf     STATUS,RP1  ;Selecciona el Banco 2
            movwf   EEDATA
            call    EE_Write    ;Graba el nuevo valor del contador en la EEPROM
            goto    Loop

            end
```

PROGRAMA 8

El manejo de la pantalla LCD

Este ejemplo pretende introducirnos en el manejo de la pantalla LCD, para la visualización de diferentes mensajes.

```
                List    p=16F876        ;Tipo de procesador
                include "P16F876.INC"   ;Definiciones de registros internos

Lcd_var         equ     0x20            ;Variables (2) de las rutinas de manejo del LCD
Delay_Cont      equ     0x22            ;Variable para la temporización
Temporal_1      equ     0x23            ;Variable temporal
Temporal_2      equ     0x24            ;Variable temporal

                org     0x05
                goto    Inicio

                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

;-----------------------------------------------------------------------------------------------------------------------
;La directiva dt genera tantas intsrucciones retlw como bytes o caracteres contenga

Mens_0          equ     $
                dt      "Trabajo / Explic",0x00

Mens_1          equ     $
                dt      "PIC 16F87x",0x00

Mens_2          equ     $
                dt      "Sebastian Martin",0x00

Mens_3          equ     $
                dt      "  Andoni  Beraza",0x00


;-----------------------------------------------------------------------------------------------------------------------
;Delay_var: Esta rutina de propósito general realiza una temporización variable entre 10 mS y 2.5". Se
;emplea un preescaler de 256 y al TMR0 se le carga con 195. La velocidad de trabajo es de 20Mhz y por
;tanto el TMR0 se incrementa cada 200nS. De esta forma, el TMR0 debe contar 195 eventos que, con un
;preescaler de 256 hace un intervalo total de 10000 uS (195 * 256 * 0,2). Este intervalo de 10 mS se repite
;tantes veces como indique la variable "Delay_cont", es por ello que el delay mínimo es de 10 mS
;("Delay_cont=1) y el máximo de 2.5" (Delay_cont=255).

Delay_var:      bcf     INTCON,T0IF     ;Desconecta el flag de rebosamiento
                movlw   b'195'
                movwf   TMR0            ;carga el TMR0 con 195
Intervalo       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 10 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 el Puerto B
                bsf     STATUS,RP0      ;Selecciona banco 1
                clrf    TRISB           ;Puerta B se configura como salida
                movlw   0x06
                movwf   ADCON1          ;Puerta A digital
                movlw   b'00110001'
                movwf   TRISA           ;RA1-RA3 salidas
                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   b'11000000'
                call    LCD_REG         ;Coloca cursor en 2ª fila del LCD
                movlw   Mens_1
                call    Mensaje         ;Visualiza mensaje 1
                movlw   b’200’
                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_2
                call    Mensaje         ;Visualiza el mensaje 2
                movlw   b'11000000'
                call    LCD_REG         ;Coloca cursor en 2ª fila del LCD
                movlw   Mens_3
                call    Mensaje         ;Visualiza el mensaje 3
                movlw   .200
                movwf   Delay_Cont
                call    Delay_var       ;Temporiza 2 segundos
                goto    Loop

                end
```

PROGRAMA 9

Introducción al manejo del teclado

Haciendo uso de las rutinas incluidas en el fichero TECLADO.INC, se trata de leer el teclado y, visualizar sobre los leds de la puerta B el código BCD de la tecla pulsada. La visualización se mantiene estable durante dos segundos hasta una nueva pulsación.

Se trata de un ejemplo en el que la Puerta B se reconfigura dinámicamente. Inicialmente es configurada como salida para presentación del resultado. Posteriormente, la rutina de exploración del teclado reconfigura RB0-RB3 como salidas y RB4-RB7 como entradas.

```
                List    p=16F876        ;Tipo de procesador
                include "P16F876.INC"   ;Definiciones de registros internos

Key_var         equ     0x20            ;Inicio de las 6 variables empleadas por las rutinas de manejo
del teclado
Delay_Cont      equ     0x26            ;Variable para la temporización

                org     0x05
                goto    Inicio

                include "TECLADO.INC"           ;Incluye rutinas de manejo del teclado

;-----------------------------------------------------------------------------------------------------------------
Delay_var:      bcf     INTCON,T0IF     ;Desconecta el flag de rebosamiento
                movlw   b'195'
                movwf   TMR0            ;carga el TMR0 con 195
Intervalo       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 10 mS
                return
;-----------------------------------------------------------------------------------------------------------------

Inicio          clrf    PORTB           ;Borra los latch de salida
                bsf     STATUS,RP0      ;Selecciona banco 1
                clrf    TRISB           ;Puerta B se configura como salida
                movlw   b'00000111'
                movwf   OPTION_REG      ;Preescaler de 256 para el TMR0
                bcf     STATUS,RP0      ;Selecciona banco 0

Loop            call    Key_Scan        ;Explora el teclado
                movlw   0x80
                subwf   Tecla,W
                btfsc   STATUS,Z        ;Hay alguna pulsada ??
                goto    Loop            ;No

                movf    Tecla,W         ;Lee el código de la tecla pulsada
                movwf   PORTB           ;Lo visualiza sobre los leds de la Puerta B
                movlw   b'200'
                movwf   Delay_Cont
                call    Delay_var       ;Temporiza 2 segundos
                clrf    PORTB           ;Desactiva visualización
                goto    Loop

                end
```

PROGRAMA 10

El teclado y el LCD.

Haciendo uso de las rutinas incluidas en los ficheros TECLADO.INC y LCD_CXX.INC, se trata de leer el teclado y, visualizar sobre el módulo LCD la tecla pulsada.

El ejemplo pretende mostrar la interrupción por cambio de estado en cualquiera de las líneas RB4-RB7 del PIC el sistema se mantiene en el modo SLEEP de bajo consumo y sólo reacciona cuando tiene lugar la pulsación de cualquier tecla.

```
                List    p=16F876            ;Tipo de procesador
                include "P16F876.INC"       ;Definiciones de registros internos

Lcd_var         equ     0x20                ;Inicio de las variables para el LCD
Key_var         equ     0x22                ;Inicio de las variables del teclado
Temporal_1      equ     0x28                ;Variable temporal nº 1
Temporal_2      equ     0x29                ;Variable temporal nº 2
Temporal_3      equ     0x2a                ;Variable temporal nº 3

                org     0x04
                goto    Interrupcion        ;Vector de interrupción
                org     0x05
                goto    Inicio

                include "LCD_CXX.INC"       ;Incluir rutinas de manejo del LCD
                include "TECLADO.INC"       ;Incluir rutinas de manejo del teclado

Tabla_Mensajes  movwf   PCL                 ;Desplazamiento sobre la tabla

Mens_0          equ     $
                dt      "Se ha pulsado: ",0x00

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

;Se incluye una temporización de 10mS para eliminar los rebotes de las teclas

Delay_10ms      bcf     INTCON,T0IF         ;Desactiva flag del TMR0
                movlw   b'195'
                movwf   TMR0                ;Carga el TMR0 con 195
Delay_1         btfss   INTCON,T0IF         ;Han transcurrido 10mS ??
                goto    Delay_1             ;No, esperar
                return
```

;Programa de tratamiento de la interrupción por cambio de estado

```
Interrupcion    bcf     INTCON,RBIE     ;Desactiva mascara RBIE
                call    Key_Scan        ;Explora el teclado
                movf    Tecla,W
                movwf   Temporal_3      ;Salva la tecla temporalmente
                call    Delay_10ms      ;Elimina rebotes

Inter_1         call    Key_Scan        ;Explora el teclado
                movlw   0x80
                subwf   Tecla,W
                btfss   STATUS,Z        ;Se ha liberado la tecla pulsada ?
                goto    Inter_1         ;No, esperar que se libere
                call    Delay_10ms      ;Eliminar rebotes

                call    UP_LCD          ;Configura Puertas A y B como salidas para manejo
del LCD
                movlw   0x8f
                call    LCD_REG         ;Posiciona el cursor del LCD
                movf    Temporal_3,W    ;Recupera la tecla que se pulsó
                sublw   b'9'
                btfss   STATUS,C        ;Es mayor que 9 (A, B,C,D,E,F)?
                goto    Mayor_que_9     ;Si
                movf    Temporal_3,W    ;No
                addlw   0x30            ;Ajuste ASCII de los caracteres del 0 al 9
                call    LCD_DATO        ;Visualizar sobre el LCD
                goto    Inter_Fin

Mayor_que_9     movf    Temporal_3,W
                addlw   0x37            ;Ajuste ASCII de los caracteres de la A a la F
                call    LCD_DATO        ;Visualiza sobre el LCD

Inter_Fin               clrf    PORTA
                clrf    PORTB
                bsf     STATUS,RP0      ;Selecciona banco 1
                movlw   b'11110000'
                movwf   TRISB           ;RB0-RB3 salidas, RB4-RB7 entradas
                nop
                nop                     ;Tiempo de espera para estabilizar la puerta B
                bcf     STATUS,RP0      ;Selecciona banco 0

                bcf     INTCON,RBIE     ;Desconecta máscara de interrupción RBIE
                movf    PORTB,W         ;Lee estado actual de reposo de las entradas
                bcf     INTCON,RBIF     ;Reponer el flag de interrupción
                bsf     INTCON,RBIE     ;Activa máscara de interrupción RBIE
                retfie

Inicio:         bsf     STATUS,RP0      ;Selecciona página 1 de datos
                movlw   0x06
                movwf   ADCON1          ;Puerta A digital
                movlw   b'00000111'
                movwf   OPTION_REG      ;Activa Pull-Up para las entradas de la puerta B
                bcf     STATUS,RP0      ;Selecciona página 0 de datos

                call    UP_LCD          ;Configura Puerta A y B como salidas
                call    LCD_INI         ;Rutina de inicialización del LCD
                movlw   b'00001100'
                call    LCD_REG         ;LCD en ON
                movlw   b'00000001'
```

```
                call    LCD_REG         ;Borra LCD y HOME

;Salida del mensaje "Tecla pulsada:"

                movlw   Mens_0
                call    Mensaje         ;Visualiza el mensaje

                clrf    PORTA
                clrf    PORTB
                bsf     STATUS,RP0      ;Selecciona banco 1
                movlw   b'11110000'
                movwf   TRISB           ;RB0-RB3 salidas, RB4-RB7 entradas
                nop
                nop                     ;Tiempo de espera para estabilizar la puerta B
                bcf     STATUS,RP0      ;Selecciona banco 0

                bcf     INTCON,RBIE     ;Desconecta máscara de interrupción RBIE
                movf    PORTB,W         ;Lee estado actual de reposo de las entradas
                bcf     INTCON,RBIF     ;Reponer el flag de interrupción
                bsf     INTCON,RBIE     ;Activa máscara de interrupción RBIE
                bsf     INTCON,GIE      ;Activa interrupciones


Loop            sleep
                nop
                goto    Loop

                end
```

PROGRAMA 11

El TMR1 en el modo contador. Frecuencímetro

Mediante un generador de onda cuadrada, se aplican pulsos por la línea RC0/T1CKI.

El TMR1 cuenta los pulsos durante un intervalo de 1s. Al resultado de la cuenta representa el número de pulsos por segundo o herzios.

Dicha frecuencia se visualiza por la pantalla LCD del entrenador

```
                List    p=16F876        ;Tipo de procesador
                include "P16F876.INC"   ;Definiciones de registros internos

Lcd_var         equ     0x20            ;Inicio de variables de las rutinas LCD
Byte_L          equ     0x22            ;Parte baja del byte a convertir
Byte_H          equ     0x23            ;Parte alta del byte a convertir
BCD_2           equ     0x24            ;Byte 2 de conversión a BCD
BCD_1           equ     0x25            ;Byte 1 de conversión a BCD
BCD_0           equ     0x26            ;Byte 0 de conversión a BCD
Contador        equ     0x27            ;Variable de contaje
Temporal        equ     0x28            ;Variable temporal
Delay           equ     0x29            ;Variable para la temporización

                org     0x04            ;Vector de interrupción
                goto    Inter

                org     0x05
                goto    Inicio

                include "LCD_CXX.INC"           ;Incluye rutinas de manejo del LCD

;Visualizar: Visualiza sobre la pantalla LCD los cinco dígitos situados en las variables BCD_0, BC_1 y
;BCD_2

Visualizar      movlw   0x80
                call    LCD_REG         ;Posiciona el cursor
                movlw   3
                movwf   Contador        ;Inicia contador de bytes a convertir
                movlw   BCD_0
                movwf   FSR             ;Inicia puntero índice
Visual_loop     swapf   INDF,W
                andlw   0x0f
                iorlw   0x30            ;Convierte a ASCII el nible de más peso
                call    LCD_DATO        ;Lo visualiza
                movf    INDF,W
                andlw   0x0f
                iorlw   0x30            ;Convierte a ASCII el nible de menos peso
                call    LCD_DATO        ;Lo visualiza
                decf    FSR,F           ;Siguiente byte
                decfsz  Contador,F
                goto    Visual_loop
                return
```

```
;16Bits_BCD: Esta rutina convierte un número binario de 16 bits situado en Cont_H y Cont_L y, lo
;convierte en 5 dígitos BCD que se depositan en las variables BCD_0, BCD_1 y BCD_2, siendo esta
;última la de menos peso.

Bits16_BCD  bcf     STATUS,C
            clrf    Contador
            bsf     Contador,4      ;Carga el contador con 16
            clrf    BCD_0
            clrf    BCD_1
            clrf    BCD_2           ;Puesta a 0 inicial

Loop_16     rlf     Byte_L,F
            rlf     Byte_H,F
            rlf     BCD_2,F
            rlf     BCD_1,F
            rlf     BCD_0,F         ;Desplaza a izda. (multiplica por 2)
            decfsz  Contador,F
            goto    Ajuste
            return

Ajuste      movlw   BCD_2
            movwf   FSR             ;Inicia el índice
            call    Ajuste_BCD      ;Ajusta el primer byte
            incf    FSR,F
            call    Ajuste_BCD      ;Ajusta el segundo byte
            incf    FSR,F
            call    Ajuste_BCD
            goto    Loop_16

Ajuste_BCD  movf    INDF,W
            addlw   0x03
            movwf   Temporal
            btfsc   Temporal,3      ;Mayor de 7 el nibble de menos peso ??
            movwf   INDF            ;Si, lo acumula
            movf    INDF,W
            addlw   0x30
            movwf   Temporal
            btfsc   Temporal,7      ;Mayor de 7 el nibble de menos peso ??
            movwf   INDF            ;Si, lo acumula
            return

;Programa de tratamiento de la interrupción que se provoca cuando el TMR0 temporice 10mS.
;Trabajando a 20MHz el TMR0 evoluciona cada 0.2 uS. Con un preescaler de 256, hay que cargar el
;valor 195 para provocar una interrupción cada 10 mS. Esta se repite 100 veces para obtener una
;temporización total de 1"

Inter       decfsz  Delay,F         ;Ha pasado 1000mS (1") ??
            goto    No_1000_mS      ;No
Si_1000_mS  bcf     T1CON,0         ;TMR1 en Off, cuenta de pulsos externos detenida
            bcf     STATUS,C
            movf    TMR1L,W
            movwf   Byte_L          ;Salva parte baja del contador
            movf    TMR1H,W
            movwf   Byte_H          ;Salva parta alta del contador
            call    Bits16_BCD      ;Convierte a BCD el resultado de la cuenta
            call    Visualizar      ;Visualiza el resultado en el LCD
            movlw   b'195'
            movwf   TMR0            ;Repone el TMR0 para temporizar 10 ms
            movlw   b'100'
            movwf   Delay           ;Repone variable para temporizar otro segundo
```

```
                bcf     INTCON,2        ;Repone flag del TMR0
                clrf    TMR1L
                clrf    TMR1H           ;Borra el TMR1
                bsf     T1CON,0         ;TMR1 en On, se inicia la nueva cuenta de pulsos externos
                retfie

No_1000_mS      movlw   b'195'
                movwf   TMR0            ;Repone para temporizar otros 10mS
                bcf     INTCON,2        ;Repone el flag del TMR0
                retfie

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

Inicio          clrf    PORTB           ;Borra los latch de salida
                clrf    PORTA           ;Borra los latch de salida
                bsf     STATUS,RP0      ;Selecciona banco 1
                movlw   b'00000110'
                movwf   ADCON1          ;Puerta A E/S digitales
                clrf    TRISB           ;Puerta B se configura como salida
                clrf    TRISA           ;Puerta A se configura como salida
                movlw   b'11000111'
                movwf   OPTION_REG      ;Preescaler de 256 asociado al TMR0
                movwf   TRISC           ;Puerta C como entrada
                bcf     STATUS,RP0      ;Selecciona banco 0

;El TMR1 actúa como contador externo asíncrono y con un preescaler de 1:1

                movlw   b'00000010'
                movwf   T1CON           ;TMR1 Off

                clrf    TMR1L
                clrf    TMR1H           ;Puesta a 0 del TMR1

                call    UP_LCD          ;Configura puerto para el LCD
                call    LCD_INI         ;Inicia el LCD
                movlw   b'00001100'
                call    LCD_REG         ;LCD On, cursor y blink Off

                movlw   b'100'
                movwf   Delay           ;Prepara temporización total de 1000mS (1")
                movlw   b'195'
                movwf   TMR0            ;TMR0 comienza a temporizar 10 ms

                bsf     T1CON,0         ;TMR1 en On, comienza la cuenta de pulsos externos
                movlw   b'10100000'
                movwf   INTCON          ;Habilita interrupción del TMR0

Loop            nop
                goto    Loop            ;Bucle

                end
```

PROGRAMA 12

El TMR2. Temporización simple, segundero

El TMR2 provoca una interrupción cada 10 mS. Transcurridas 100 interrupciones, el tiempo total transcurrido es de 1 segundo.

El display conectado a la puerta B, cuenta intervalos de 1 segundo.

```
                List    p=16F876        ;Tipo de procesador
                include "P16F876.INC"   ;Definiciones de registros internos

Delay           equ     0x20            ;Variable de temporización
Contador        equ     0x21            ;Variable del contador de segundos

                org     0x04            ;Vector de interrupción
                goto    Inter

                org     0x05
                goto    Inicio

;-------------------------------------------------------------------------------------------------------------------
;Tabla: Esta rutina convierte el código binario presente en los 4 bits de menos peso del reg. W en su
;equivalente a 7 segmentos. El código 7 segmentos retorna también en el reg. W

Tabla:          addwf   PCL,F           ;Desplazamiento sobre la tabla
                retlw   b'00111111'     ;Dígito 0
                retlw   b'00000110'     ;Dígito 1
                retlw   b'01011011'     ;Dígito 2
                retlw   b'01001111'     ;Dígito 3
                retlw   b'01100110'     ;Dígito 4
                retlw   b'01101101'     ;Dígito 5
                retlw   b'01111101'     ;Dígito 6
                retlw   b'00000111'     ;Dígito 7
                retlw   b'01111111'     ;Dígito 8
                retlw   b'01100111'     ;Dígito 9

;Programa de tratamiento de la interrupción que provoca el TMR2 cada 10mS.

Inter           decfsz  Delay,F         ;Ha pasado un segundo ??
                goto    No_es_1_seg     ;No
                incf    Contador,F      ;Si, incrementa el contador de segundos
                movlw   b'10'
                subwf   Contador,W
                btfsc   STATUS,Z        ;Contador > 9 ??
                clrf    Contador        ;Si, ponerlo a 0
                movf    Contador,W
                call    Tabla           ;Convierte a 7 segmentos
                movwf   PORTB           ;Visualiza sobre el display
                movlw   b'100'
                movwf   Delay           ;Reinicia variable delay
No_es_1_seg     bcf     PIR1,TMR2IF     ;Repone el flag del TMR2
                retfie
```

```
Inicio      clrf    PORTB       ;Desconecta salidas
            bsf     STATUS,RP0  ;Selecciona banco 1
            clrf    TRISB       ;Puerta B se configura como salida
            bsf     PIE1,TMR2IE ;Habilita interrupción del TMR2
            movlw   b'195'
            movwf   PR2         ;Carga registro de periodos con 195
            bcf     STATUS,RP0  ;Selecciona banco 0

;El TMR2 emplea un preescaler y un postcaler de 1:16 (total 1:256). Trabajando a una frecuencia de 20
;MHZ el TMR2 evoluciona cada 3.2uS (preescaler 1:16). La cuenta avanza hasta alcanzar el valor del
;registro de periodos (195), con lo que el tiempo transcurrido es de 624 uS. Este lapsus se repite 16 veces
;(postcaler 1:16) antes de provocar la interrupción (al de 9984 uS).

            movlw   b'01111111'
            movwf   T2CON       ;TMR2 On, preescaler/postcaler = 1:16
            clrf    TMR2        ;Inicia el TMR2

            movlw   b'100'
            movwf   Delay       ;Inicia variable de delay

            movlw   b'11000000'
            movwf   INTCON      ;Habilita interrupciones

            clrf    Contador    ;Inicia el contador de segundos

Loop        nop
            goto    Loop        ;Bucle

            end
```

PROGRAMA 13

Los módulos CCPx. Modo de Captura.

Medida del tiempo entre un pulso y el siguiente.

El ejemplo emplea el módulo CCP1 y trata de capturar el valor del TMR1 cada vez que lleguen un flanco descendente y otro ascendente por la línea RC2/CCP1. Conocida la velocidad a la que evoluciona el TMR1, se puede determinar el lapsus de tiempo transcurrido entre ambos flancos, lo que nos dará el tiempo transcurrido entre el final de un pulso y el comienzo del siguiente.

La pantalla LCD visualiza dicho lapsus de tiempo expresado en microsegundos.

```
                List    p=16F876        ;Tipo de procesador
                include "P16F876.INC"   ;Definiciones de registros internos

Lcd_var         equ     0x20            ;Variables para las rutinas de manejo del LCD
Byte_L          equ     0x22            ;Parte baja del byte a convertir
Byte_H          equ     0x23            ;Parte alta del byte a convertir
BCD_2           equ     0x24            ;Byte 2 de conversión a BCD
BCD_1           equ     0x25            ;Byte 1 de conversión a BCD
BCD_0           equ     0x26            ;Byte 0 de conversión a BCD
Contador        equ     0x27            ;Variable de contaje
Temporal        equ     0x28            ;Variable temporal
Captura         equ     0x29            ;Nº de capturas
Cap_1_L         equ     0x2a
Cap_1_H         equ     0x2b            ;Variables temporales para las capturas

                org     0x04
                goto    Inter           ;Vector de interrupción

                org     0x05
                goto    Inicio

                include "LCD_CXX.INC"           ;Incluye rutinas de manejo del LCD

;Visualizar: Visualiza sobre la pantalla LCD los cinco dígitos situados en las variables
;BCD_0, BC_1 y BCD_2

Visualizar      movlw   0x80
                call    LCD_REG         ;Posiciona el cursor
                movlw   3
                movwf   Contador        ;Inicia contador de bytes a convertir
                movlw   BCD_0
                movwf   FSR             ;Inicia puntero índice
Visual_loop     swapf   INDF,W
                andlw   0x0f
                iorlw   0x30            ;Convierte a ASCII el nible de más peso
                call    LCD_DATO        ;Lo visualiza
                movf    INDF,W
                andlw   0x0f
                iorlw   0x30            ;Convierte a ASCII el nible de menos peso
                call    LCD_DATO        ;Lo visualiza
                decf    FSR,F           ;Siguiente byte
                decfsz  Contador,F
                goto    Visual_loop
                movlw   ' '
```

```
                call    LCD_DATO    ;Visualiza ' '
                movlw   0xe4
                call    LCD_DATO    ;Visualiza micro
                movlw   'S'
                call    LCD_DATO    ;Visualiza 'S'
                return
```

;16Bits_BCD: Esta rutina convierte un número binario de 16 bits situado en Byte_H y Byte_L y, en 5 ;dígitos BCD que se depositan en las variables BCD_0, BCD_1 y BCD_2, siendo esta última la de menos ;peso.

```
Bits16_BCD      bcf     STATUS,C
                clrf    Contador
                bsf     Contador,4  ;Carga el contador con 16
                clrf    BCD_0
                clrf    BCD_1
                clrf    BCD_2       ;Puesta a 0 inicial

Loop_16         rlf     Byte_L,F
                rlf     Byte_H,F
                rlf     BCD_2,F
                rlf     BCD_1,F
                rlf     BCD_0,F     ;Desplaza a izda. (multiplica por 2)
                decfsz  Contador,F
                goto    Ajuste
                return

Ajuste          movlw   BCD_2
                movwf   FSR         ;Inicia el índice
                call    Ajuste_BCD  ;Ajusta el primer byte
                incf    FSR,F
                call    Ajuste_BCD  ;Ajusta el segundo byte
                incf    FSR,F
                call    Ajuste_BCD
                goto    Loop_16

Ajuste_BCD      movf    INDF,W
                addlw   0x03
                movwf   Temporal
                btfsc   Temporal,3  ;Mayor de 7 el nibble de menos peso ??
                movwf   INDF        ;Si, lo acumula
                movf    INDF,W
                addlw   0x30
                movwf   Temporal
                btfsc   Temporal,7  ;Mayor de 7 el nibble de menos peso ??
                movwf   INDF        ;Si, lo acumula
                return
```

;Programa de tratamiento de la interrupción que provoca el módulo CCP1 cada vez que se detecta, ;primero un flanco descendente y, luego un ascendente por la línea RC2/CCP1.

```
Inter           bcf     PIR1,CCP1IF ;Repone el flag del módulo CCP1
                btfsc   Captura,0   ;Es la captura del flanco ascendente ??
                goto    Medir       ;Si, medir el tiempo transcurrido entre ambas
                incf    Captura,F   ;No, ha sido la captura del flanco descendente
                movf    CCPR1L,W
                movwf   Cap_1_L
                movf    CCPR1H,W
                movwf   Cap_1_H     ;Salvar, temporalmente, el 1er valor capturado
                bsf     CCP1CON,0   ;Capturar al flanco ascendente
```

```
                retfie

Medir           movf    Cap_1_L,W       ;Es la captura del flanco ascendente
                subwf   CCPR1L,W
                movwf   Byte_L
                btfss   STATUS,C
                incf    Cap_1_H,F
                movf    Cap_1_H,W
                subwf   CCPR1H,W
                movwf   Byte_H          ;Restar el tiempo entre la 2ª captura y la 1ª
                call    Bits16_BCD      ;Convertir a BCD
                incf    Captura,F       ;Capturar el 1er. flanco
                call    Visualizar      ;Salida a pantalla LCD
                bcf     CCP1CON,0       ;Captura al flanco descendente
                retfie

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

Inicio          clrf    PORTB           ;Desconecta salidas
                clrf    PORTA
                bsf     STATUS,RP0      ;Selecciona banco 1
                movlw   b'00000110'
                movwf   ADCON1          ;Puerta A E/S digitales
                clrf    TRISB           ;Puerta B se configura como salida
                clrf    TRISA           ;Puerta A salidas
                movlw   b'11111111'
                movwf   TRISC           ;Puerta C entrada
                bsf     PIE1,CCP1IE     ;Habilita interrupción del módulo CCP1
                bcf     STATUS,RP0      ;Selecciona banco 0

                call    UP_LCD          ;Configura puertos para el LCD
                call    LCD_INI         ;Inicia el LCD
                movlw   b'00001100'
                call    LCD_REG         ;LCD On, cursor y blink Off


;El TMR1 actúa en el modo temporizador con reloj interno y un preescaler 1:8 evoluciona cada 1.6uS.
;Según esto, el periodo máximo que se puede medir será en torno a los 100mS (10Hz). El periodo mínimo
;estará en torno a los 1.6 uS (62KHz). Para otros rangos se debe seleccionar un preescaler diferente.

                movlw   b'00110001'
                movwf   T1CON           ;TMR1 en On, preescaler 1:8

                movlw   b'11000000'
                movwf   INTCON          ;Habilita interrupciones

;El módulo CCP1 actúa en modo de captura al flanco descendente

                movlw   b'00000100'
                movwf   CCP1CON         ;Módulo CCP en On

                clrf    Captura         ;Inicia captura en el 1er. flaco descendente

Loop            nop
                goto    Loop            ;Bucle

                end
```

PROGRAMA 14

Los módulos CCPx. Modo de Comparación.

El TMR1 cuenta tantos pulsos externos como se indique mediante los interruptores RA5-RA0. Cada vez que se alcanza el valor fijado, la salida RB0 cambia de estado.

```
        List    p=16F876        ;Tipo de procesador
        include "P16F876.INC"   ;Definiciones de registros internos

        org     0x04
        goto    Inter           ;Vector de interrupción
        org     0x05
        goto    Inicio

;Programa de tratamiento de la interrupción que provoca el módulo CCP1 cada vez que el TMR1 cuenta
;tantos pulsos externos como los prefijados mediante las entradas RA5-RA0.

Inter   bcf     PIR1,CCP1IF     ;Repone el flag del módulo CCP1
        bcf     T1CON,TMR1ON    ;TMR1 en Off
        clrf    TMR1L
        clrf    TMR1H           ;Puesta a 0 del TMR1
        movlw   b'00000001'
        xorwf   PORTB,F         ;RB0 cambia de estado
        bsf     T1CON,TMR1ON    ;TMR1 en On
        retfie

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

Inicio  clrf    PORTB           ;Desconecta salidas
        clrf    PORTA
        bsf     STATUS,RP0      ;Selecciona banco 1
        movlw   b'00000110'
        movwf   ADCON1          ;Puerta A E/S digitales
        clrf    TRISB           ;Puerta B se configura como salida
        movlw   b'00111111'
        movwf   TRISA           ;Puerta A entradas
        movwf   TRISC           ;RC0 entrada
        bsf     PIE1,CCP1IE     ;Habilita interrupción del módulo CCP1
        bcf     STATUS,RP0      ;Selecciona banco 0


;El TMR1 actúa en el modo contador de pulsos externos sensible al flanco ascendente y con un preescaler
;de 1:1. Estos pulsos pueden ser suministrador por el generador del entrenador

        movlw   b'00000010'
        movwf   T1CON           ;TMR1 en Off

        movlw   b'11000000'
        movwf   INTCON          ;Habilita interrupciones
```

```
;El módulo CCP1 actúa en modo de comparación e interrupción al coincidir

                movlw   b'00001010'
                movwf   CCP1CON         ;Módulo CCP en modo comparación

                clrf    CCPR1H          ;Puesta a 0 de la parte alta del valor a comparar
                clrf    TMR1L
                clrf    TMR1H           ;Puesta a 0 del TMR1

                bsf     T1CON,TMR1ON    ;TMR1 en On, comienza a contar

Loop            clrf    CCPR1H          ;Pone a 0 la parte alta del valor a comparar
                movf    PORTA,W
                andlw   b'00111111'     ;Lee las entradas RA5-RA0
                movwf   CCPR1L          ;Ajusta la parte baja del valor a comparar
                goto    Loop            ;Bucle

                end
```

PROGRAMA 15

Los módulos CCPx. Modo PWM. Modulación de anchura de pulsos.

Consiste en generar una señal de onda cuadrada por la línea RC2/CCP1 cuyo periodo puede ser modificado así como la anchura del pulso (Duty Cycle). El periodo se determina según la fórmula T=(PR2+1)*4*Tosc*TMR2 preescaler. La duración del pulso o "Duty Cycle" (d) se determina según d=(CCPR1L:CCPCON1<5:4>)*Tosc*TMR2 preescaler.

El ejemplo emplea al módulo CCP1 con salida de señal por la línea RC2/CCP1 y un preescaler de 16. La señal de salida tiene un periodo de 640uS. La anchura del ciclo "Duty" es variable y se determina, según el valor binario de los interruptores del entrenador (RA5-RA0)

```
                List    p=16F876        ;Tipo de procesador
                include "P16F876.INC"   ;Definiciones de registros internos

Temporal        equ     0x20            ;Variable temporal

Periodo         equ     b'200'          ;Periodo 640uS (200*Preescaler de 16*0.2)

                org     0x05

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

Inicio          clrf    PORTC           ;Borra salidas
                bsf     STATUS,RP0      ;Selecciona banco 1
                movlw   0x06
                movwf   ADCON1          ;Puerta A digital
                movlw   b'00111111'
                movwf   TRISA           ;Puerta A entrada
                movlw   b'11111011'
                movwf   TRISC           ;RC2 salida
                movlw   Periodo-1
                movwf   PR2             ;Carga el registro de periodos
                bcf     STATUS,RP0      ;Selecciona banco 0

;El módulo CCP1 actúa en el modo PWM con salida de señal por RC2/CCP1

                movlw   b'00001100'
                movwf   CCP1CON

;El TMR2 trabaja con un preescaler 1:16 por lo que con una frecuencia de 20MHz evoluciona
;cada 3.2uS ((4*Tosc)*16)

                movlw   b'00000111'
                movwf   T2CON           ;T2 en On

Loop            movf    PORTA,W
                andlw   b'00111111'
                movwf   CCPR1L          ;Carga la anchura del pulso (n*8*Prescaler de 16)
                goto    Loop            ;Bucle infinito

                end
```

PROGRAMA 16

El módulo conversor ADC.

Los dispositivos PIC16F87X disponen de un convertidor A/D de 10 bits de resolución y 5 u 8 canáles de entrada analógica. Con 5 Vref=4.8 mV/Bit; con 2.5 Vref=2.4 mV/Bit

El ejemplo propone realizar la conversión de la tensión presente en el canal RA0/AN0. Esta se puede variar con el potenciómetro P1 del entrenador. El resultado de la conversión se visualiza, en binario, sobre la pantalla LCD.

```
                List    p=16F876        ;Tipo de procesador
                include "P16F876.INC"   ;Definiciones de registros internos

Lcd_var         equ     0x20            ;Variables de las rutinas LCD
Temporal_1      equ     0x22            ;Variable temporal

                org     0x05
                goto    Inicio

                include "LCD_CXX.INC"           ;Incluye rutinas de manejo del LCD

;Visualiza: Esta rutina coge los 10 bits resultantes de la conversión, presentes en ADRESH y ADRESL,
;los convierte a caracteres ACII (0 o 1) y los visualiza sobre el LCD.

Visualiza:      movlw   0x80
                call    LCD_REG         ;Sitúa el cursor del LCD
                movlw   b'9'
                movwf   Temporal_1      ;Nº de caracteres a visualizar
Visual_loop     bsf     STATUS,RP0
                rlf     ADRESL,F
                bcf     STATUS,RP0
                rlf     ADRESH,F        ;Rotación del siguiente bit
                btfsc   STATUS,C        ;Testea el bit a visualizar
                goto    Bit_1           ;Está a 1
                movlw   '0'
                goto    Visu_1
Bit_1           movlw   '1'
Visu_1          call    LCD_DATO        ;Visualiza el "0" o el "1" sobre el LCD
                decfsz  Temporal_1,F    ;Siguiente caracter
                goto    Visual_loop
                return

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

Inicio          clrf    PORTA
                clrf    PORTB
                bsf     STATUS,RP0      ;Selecciona banco 1
                movlw   b'00000110'
                movwf   ADCON1          ;Puerta A E/S digitales
                clrf    TRISB           ;Puerta B se configura como salida
                clrf    TRISA           ;RA5-RA0 salidas
                bcf     STATUS,RP0      ;Selecciona banco 0

                call    UP_LCD          ;Configura E/S para el LCD
                call    LCD_INI         ;Secuencia de inicio del LCD
                movlw   b'00001100'
                call    LCD_REG         ;LCD On, cursor y blink Off
```

```
;Se activa el ADC y se selecciona el canal RA0/AN0. Frecuencia de trabajo Fosc/32

                movlw   b'10000001'
                movwf   ADCON0          ;ADC en On, seleciona canal AN3

Loop            bsf     STATUS,RP0      ;Selecciona página 1
                movlw   b'00111111'
                movwf   TRISA           ;Puerta A entradas
                clrf    ADCON1          ;Puerta A entradas analógicas
                bcf     STATUS,RP0      ;Selecciona página 0
                bcf     PIR1,ADIF       ;Restaura el flag del conversor AD
                nop
                bsf     ADCON0,GO       ;Inicia la conversión

ADC_Wait        btfss   PIR1,ADIF       ;Fin de conversión ??
                goto    ADC_Wait        ;Todavía no

;Las líneas de la Puerta A se reconfiguran como salidas digitales para la visualización

                bsf     STATUS,RP0      ;Selecciona banco 1
                movlw   b'11000111'
                movwf   ADCON1          ;Puerta A digital
                bcf     STATUS,RP0      ;Selecciona banco 0
                call    UP_LCD          ;Reconfigura E/S para el LCD
                call    Visualiza       ;Visualiza el resultado de la conversión
                goto    Loop

                end
```