MICROCONTROLADORES PIC

INTRODUCCIÓN

A NIVEL HARDWARE

Un microcontrolador es un circuito integrado programable que integra en un solo chip las unidades de memoria para el almacenamiento de datos, aritmética – lógica para el cálculo de operaciones, las unidades de entrada y salida para comunicación con otros periféricos, temporizadores y el controlador de interrupciones.

La memoria generalmente está constituida por memoria RAM compuesta por registros que almacena datos temporales, memoria EEPROM para el almacenamiento del programa que se debe ejecutar.

La unidad aritmética lógica ALU es la encargada de realizar las operaciones aritméticas suma, resta y multiplicación y las operaciones lógicas como And, Or, Or- exclusivo.

Las unidades de entrada/salida se refieren a los puertos que tiene el micro para recibir o enviar datos en forma serie o en forma paralela. Cuenta además con módulos especiales para convertir señales analógicas a digitales o de digitales a analógicas.

Generalmente tienen arquitectura Harvard que es aquella en donde existes dos buses independientes para mejorar la velocidad de transferencia de información interna: el bus de datos y el bus de direcciones. El bus de datos puede ser de 8, 16, 32 bits y el de dirección depende de la cantidad de memoria del micro.

Los microcontroladores para temporizar sus operaciones de programación tienen internamente un reloj implementado que con solo añadir un cristal y un par de capacitores se genera la frecuencia requerida.

Para inicializar el micro después de conectar la alimentación, existe una señal de Reset que generalmente es activo bajo para limpiar registros internos y colocar bits de control.

A NIVEL SOFTWARE

Para funcionar el microcontrolador dispone de un conjunto de instrucciones que son traducidas a lenguaje de máquina (1's y 0's) por un programa que se llama Ensamblador.

Igualmente existen Compiladores que se encargan de traducir un lenguaje de alto nivel como el lenguaje C a lenguaje o código de máquina. En ambos casos es el código ejecutable que se debe grabar en la memoria del micro (EEPROM) para que se ejecute el programa y desarrolle la aplicación que se quiere.

Para editar (escribir) un programa se usa un sistema de desarrollo que el más utilizado es el MPLAB que sirve para editar el programa, sumularlo, corregirlo y posteriormente enviarlo a una tarjeta que se encargará de enviar el código de máquina al micro. Este proceso se llama programar el micro o sencillamente quemarlo. En la simulación que debe realizarse previamente, se puede observar la operación de todos los registros del micro, los puertos de entrada y de salida, observar el almacenamiento en memoria, etc.

Los fabricantes de micros, Intel, Motorola, Microchip, Texas, NEC y otros tienen kits especiales (hardware y software) para desarrollar estos procesos de programación de micros.

PARÁMETROS A CONSIDERAR

Los parámetros más importantes en un microcontrolador son:

- Bus de datos: 8, 16, 32 bits
- Capacidad de memoria: Tamaño de la memoria RAM y de la memoria EEPROM en kilobytes KB
- Velocidad: Numero de instrucciones a ejecutar por segundo. Depende de la frecuencia del oscilador del micro.
- Puertos: Puertos de entrada salida de forma paralela y serial para comunicación externa.
- Módulos: Para conversión A/D, D/A, PWM, USB, CAN, I2C, SPI, UART, USART, etc

FAMILIAS DE MICROCONTROLADORES DE MICROCHIP

Tomado de:
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2551

Microchip ofrece soluciones para microcontroladores de gama completa de 8-bits , 16 bits y 32 bits , con una poderosa arquitectura, tecnologías flexibles de la memoria, herramientas de desarrollo fácil de usar, documentación técnica completa y apoyo al diseño a través de una red de ventas y distribución . Los beneficios obtenidos por la selección de soluciones de microcontroladores de Microchip son:

- Migración fácil a través de familias de productos
- Bajo riesgo de desarrollo de productos
- Reducción del costo total del sistema
- Centros de apoyo regionales de capacitación en todo el mundo
- Programación de la producción de servicios
- Certificado de calidad
- Uso práctico de microchipDIRECT
- Microchip es el # 1 en todo el mundo en microcontroladores de 8 bits

Migración fácil a través de las familias de productos

Todos las opciones PIC ® y dsPIC MCUs ® DSC de migración de apoyo, permiten la ampliación de sus diseños arriba o hacia abajo. Los factores principales de selección de la plataforma son:

- Portafolio completo: de 6 a 100 pines, 384B a 512 KB de memoria de programa.
- Operación hasta 80 MHz
- Arquitecturas compatibles
- Compatibilidad en pines que facilita el reemplazo
- Rango de las tecnologías de memoria: Auto Programación Flash, OTP, ROM
- Fácil migración a través de 8, 16 y familias de 32 bits

Bajo riesgo de desarrollo de productos y más rápida al mercado

Microchip ofrece soluciones con clase de herramientas de desarrollo potentes y asequibles para el desarrollo de aplicaciones. Compiladores, ensambladores, programadores, simuladores, emuladores in-circuit, son los componentes disponibles de MPLAB ® Integrated Development Environment (IDE). Los aspectos más destacados de desarrollo de estas herramientas de Microchip son:

- MPLAB ® IDE es un entorno libre de desarrollo sencillo y potente, que da apoyo a todos los productos MCU y DSC
- Ediciones para estudiantes gratuitas disponibles como compiladores de C

- Bibliotecas de software probada para aplicaciones específicas y diseños de referencia
- Gráfica de usuario Interfaz de usuario (GUI) basada en paquetes de software de diseño
- Desarrollo de herramientas en software y hardware de desarrollo disponibles

Sistema de costo total más bajo

Adecuados niveles de integración de periféricos analógicos y digitales de simple a complejo con un número mínimo de componentes para reducir el costo total del sistema con una mayor fiabilidad. Algunos ejemplos de la funcionalidad integrada en nuestro MCUs y DSC son:

- Periféricos de Comunicaciones: SPI, I^2C ™, UART, CAN, USB, Ethernet, IrDA ® , LIN
- Control de periféricos: captura / comparación, contadores, reloj en tiempo real y calendario, control de motores y fuente de alimentación PWM
- Manejadores (drivers) para pantalla integrada: LED, LCD
- Osciladores internos en un chip y PLL
- Periféricos analógicos: convertidores A / D, comparadores, amplificadores operacionales, detección de brown-out y reset, detección de baja tensión, sensores de temperatura, conversores D / A, reguladores de voltaje
- Microchip también ofrece una variedad de memoria , interfaz y productos análogos .

Soporte de entrenamiento en Centros Regionales de Formación en el mundo

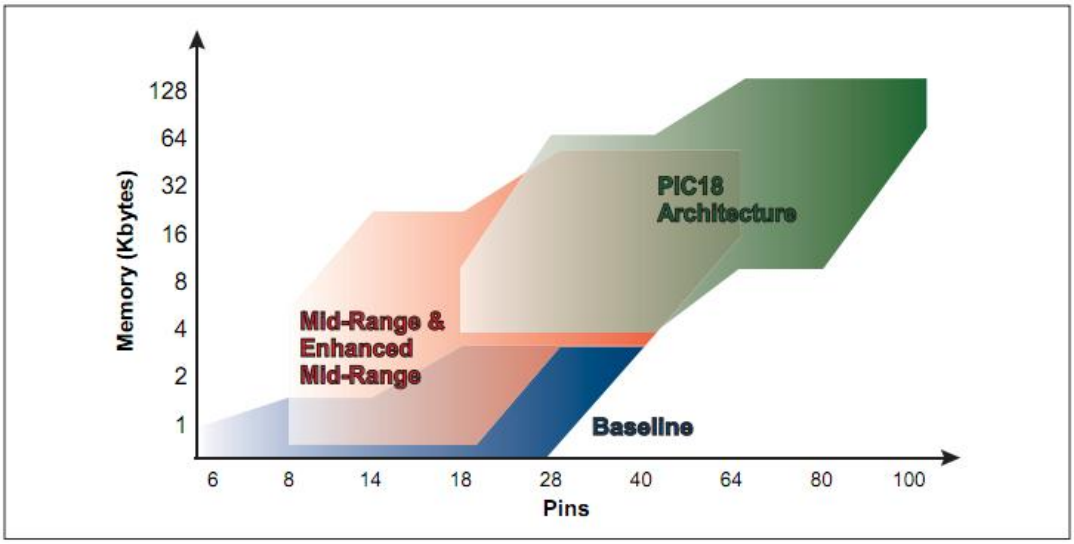
Microchip ofrece la línea de recursos de soporte técnico cuando lo necesite en global apoyo técnico . Además, Microchip ofrece estándar de bibliotecas de software , diseños de referencia, notas de aplicación , y seminarios en línea y en Centros Regionales de Capacitación .

Productos de la microchip PIC (Peripherical Interface Controller)

- Microcontroladores PIC de 8 bits
- Microcontroladores PIC, MCU Y dsPIC de 16 bits
- Microcontroladores PIC de 32 bits

Microcontroladores PIC de 8 bits

8-bit PIC MCU Architectures



	Baseline Architecture	Mid-Range Architecture	Enhanced Mid-Range Architecture	PIC18 Architecture
Pin Count	6-40	8-64	8-64	18-100
Interrupts	No	Single interrupt capability	Single interrupt capability with hardware context save	Multiple interrupt capability with hardware context save
Performance	5 MIPS	5 MIPS	8 MIPS	Up to 16 MIPS
Instructions	33, 12-bit	35, 14-bit	49, 14-bit	83, 16-bit
Program Memory	Up to 3 KB	Up to 14 KB	Up to 28 KB	Up to 128 KB
Data Memory	Up to 134B	Up to 368B	Up to 1.5 KB	Up to 4 KB
Hardware Stack	2 level	8 level	16 level	32 level
Features	• Comparator • 8-bit ADC • Data Memory • Internal Oscillator	In addition to Baseline: • SPI/I²C™ • UART • PWMs • LCD • 10-bit ADC • Op Amp	In addition to Mid-Range: • Multiple Communication Peripherals • Linear Programming Space • PWMs with Independent Time Base	In addition to Enhanced Mid-Range: • 8x8 Hardware Multiplier • CAN • CTMU • USB • Ethernet • 12-bit ADC
Highlights	Lowest cost in the smallest form factor	Optimal cost to performance ratio	Cost effective with more performance and memory	High performance, optimized for C programming, advanced peripherals
Total Number of Devices	16	58	29	193
Families	PIC10, PIC12, PIC16	PIC12, PIC16	PIC12F1XXX, PIC16F1XXX	PIC18

Las características se pueden encontrar en:
http://ww1.microchip.com/downloads/en/DeviceDoc/39630g.pdf

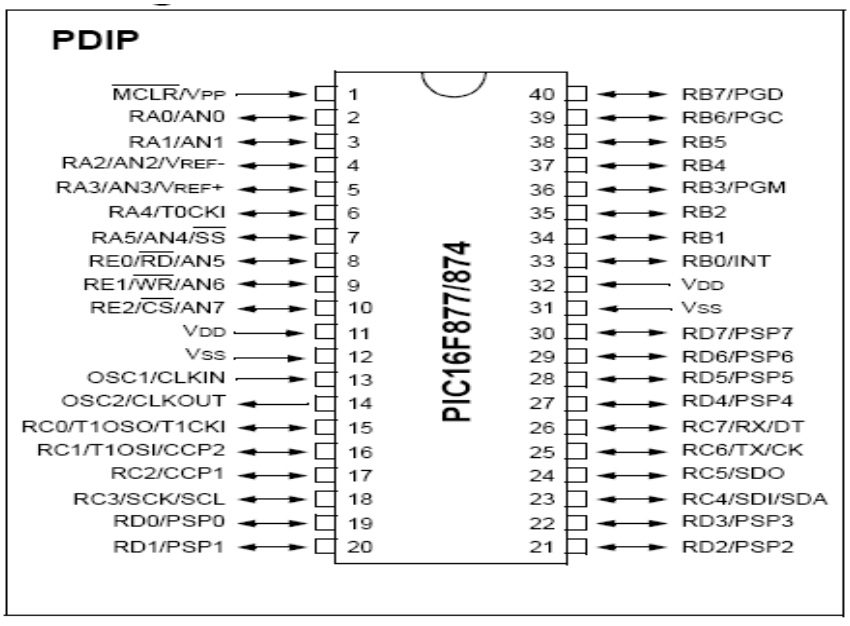
MICROCONTROLADOR PIC 16F87X

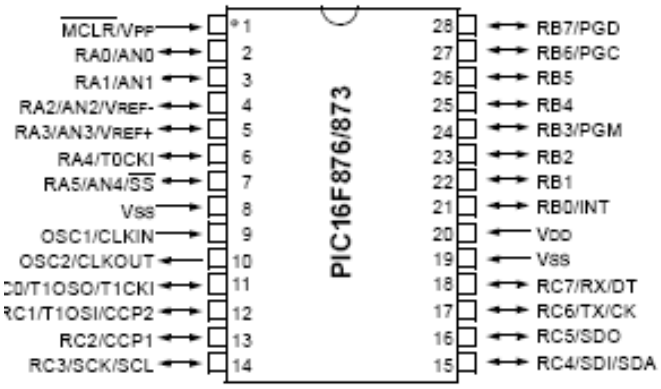
1. GENERALIDADES

1.1 CARACTÉRÍSTICAS

- CPU tipo RISC (conjunto de instrucciones reducidas)
- Modelos 873/6: 28 pines con tres puertos PA, PB, PC con 22 líneas de E/S y conversor A/D de 5 canales
- Modelos 874/7: 40 pines con cinco puertos PA, PB, PC, PD, PE con 33 líneas de E/S y conversor A/D de 8 canales
- Conversor A/D de 10 bits
- 35 instrucciones de 14 bits
- Ejecución de una instrucción en un ciclo, excepto las de bifurcación que la hacen en dos
- Frecuencia de 20 Mhz
- Memoria de programa flash hasta 8K x 14bits y memoria de datos RAM hasta 368 bytes, EEPROM hasta 256 bytes
- Hasta 14 fuentes de interrupción internas y externas
- Programación serie in-circuit en dos pines
- Bajo consumo 2mA para 5V
- Tres timers: timer0, timer1, timer2
- Dos módulos de captura-comparación-pwm (CCP1 ,CCP2)
- Puerto serie síncrono (SSP) con SPI y I2C
- USART
- Puerto paralelo esclavo (PSP) para los de 40 pines

1.2 DISTRIBUCIÓN DE PINES

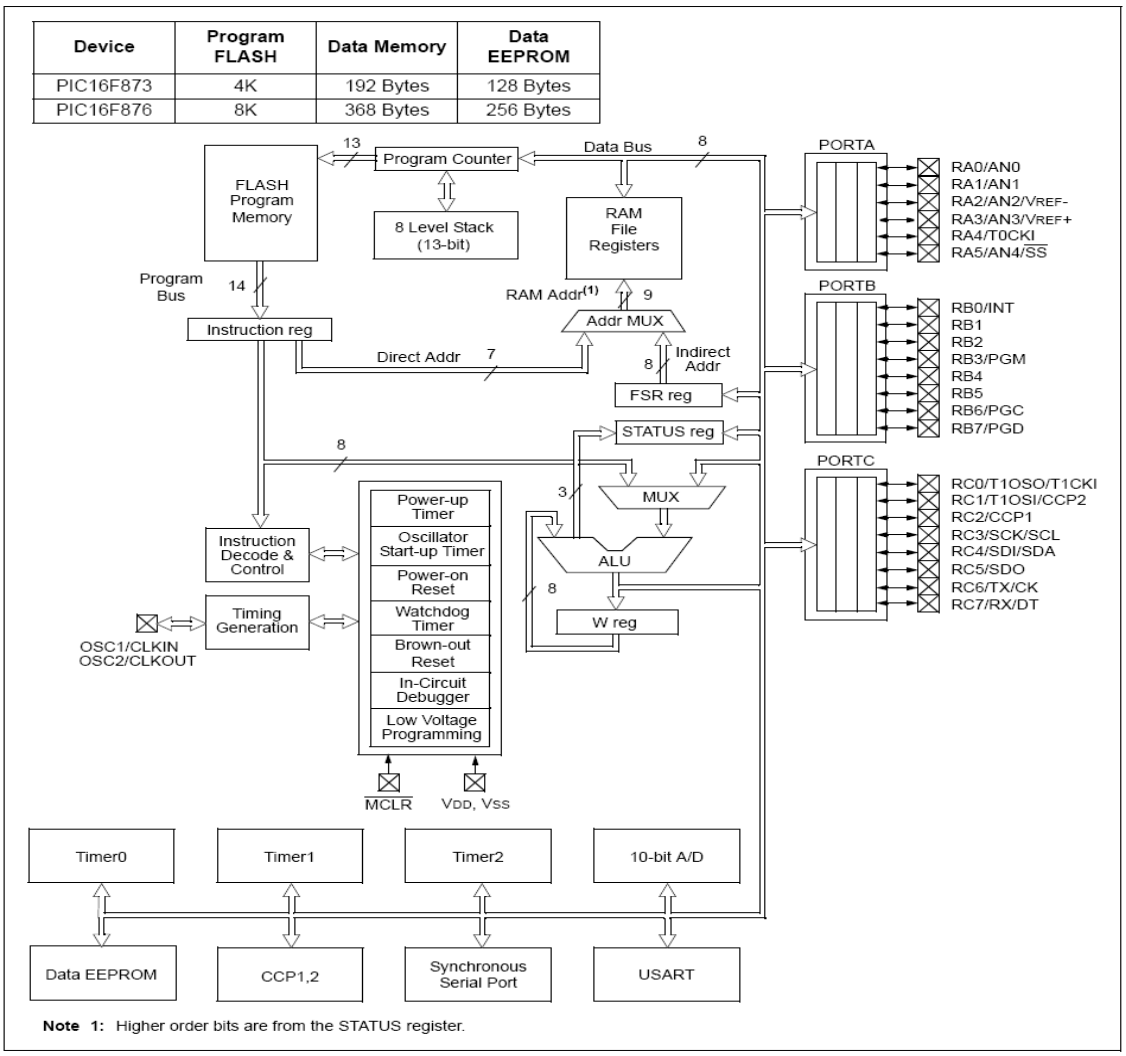




1.3 DIAGRAMA EN BLOQUES

16F873/16F876

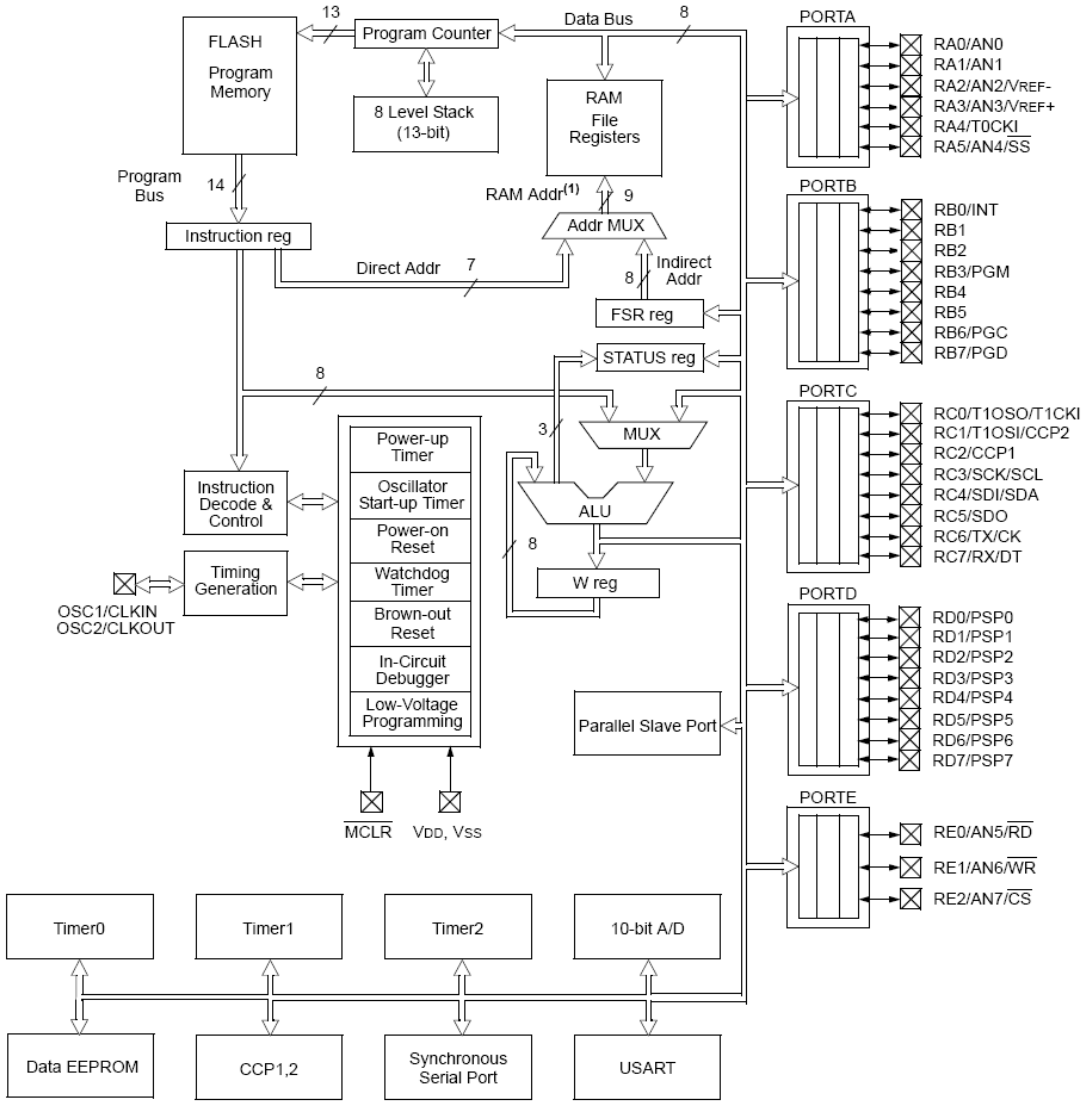
Device	Program FLASH	Data Memory	Data EEPROM
PIC16F873	4K	192 Bytes	128 Bytes
PIC16F876	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

16F874/16F877

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



Note 1: Higher order bits are from the STATUS register.

1.4 MEMORIA DE PROGRAMA

- La memoria de programa es de tipo flash y puede tener una capacidad de 4K u 8K dividida en bancos de 2K y direccionada por el contador de programa PC de 13 bits de modo que puede acceder hasta 8Kx14 bits. El 16F873/874 tiene un memoria de 4Kx14 bits (2 páginas) y el 16F876/877 una memoria de 8Kx14 bits (4 páginas). Cada página direccionada por 11 bits (2^{11} = 2K) tiene un tamaño de 07FF.

- El vector reset se encuentra en la posición 000h y el vector de interrupción en la 004h.

FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK

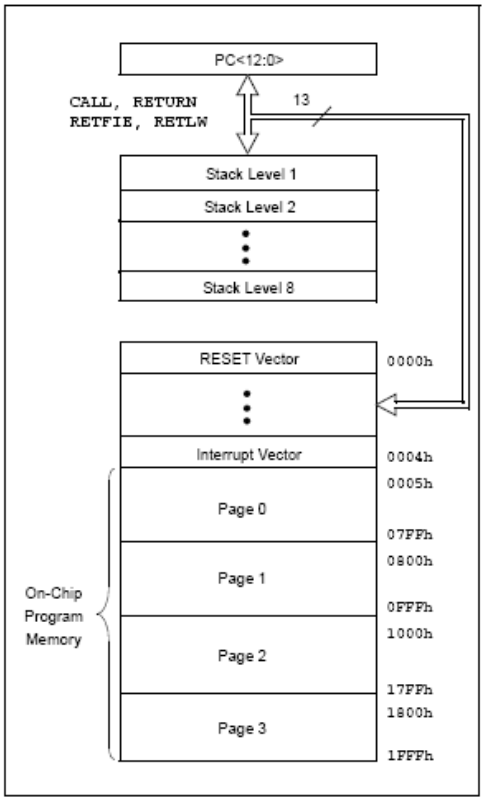
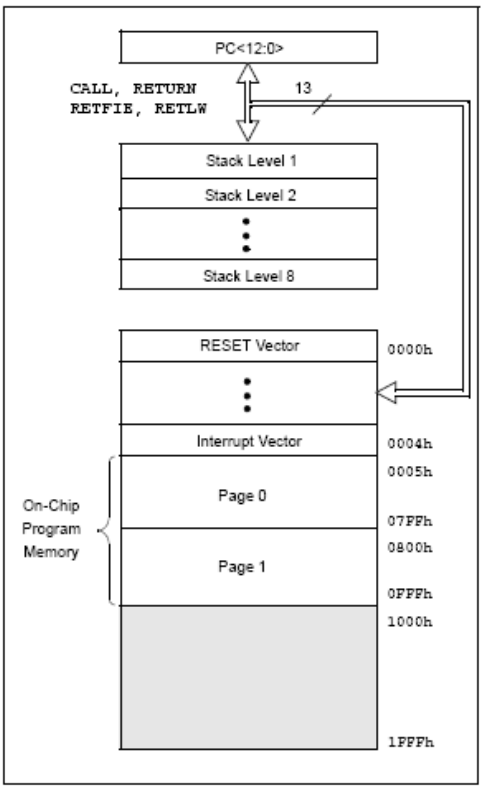


FIGURE 2-2: PIC16F874/873 PROGRAM MEMORY MAP AND STACK



Ejemplo: Pasar de página0 a página1

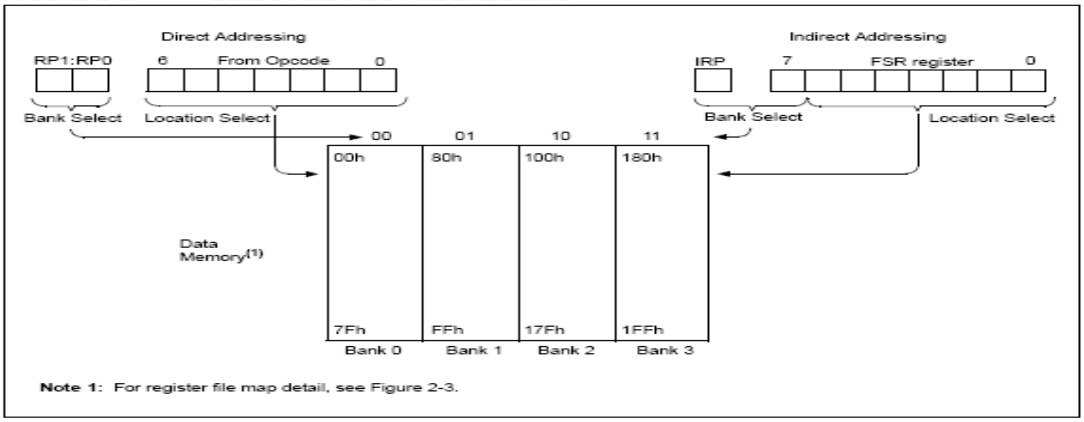
```
        ORG     0x500
        bcf     PCLATH,4
        bsf     PCLATH,3        ;selecciona página1
        call    SUBRUT1         ;subrutina de página1
        ----
        ----
        ORG     0x900           ;página1
SUBRUT1 ----
        ----
        return                  ;retorno a página0
```

1.5 MEMORIA DE DATOS

- La constituye la RAM y la EEPROM. consta de 4 bancos de 128 bytes cada uno seleccionados por los bits RP1, RP0 del reg-status. Algunos tienen 192 bytes de RAM y otros 368 bytes.
- Las instrucciones **call** y **goto** sólo proporcionan 11 bits de dirección (2k un banco), por tanto para salir del banco actual se deben programar los bits 4, 3 del reg PCLATH.
- En direccionamiento directo los bits RP1, RP0 seleccionan el banco. Para direccionamiento indirecto se usa el registro FSR, el banco lo determina el bit de más peso de FSR concatenado con el bit IRP del registro status.
- Est{a particionada en múltiples bancos que contienen los registros de funciones especiales (SFR) y los registros de propósito general (GPR). Cada banco tiene 128 bytes (7Fh). Los registros especiales más frecuentemente usados tienen espejo en otro banco con el fin de hacer más rápido el acceso y reducir código de programa.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note 1: For register file map detail, see Figure 2-3.

Ejemplo: Borrar la memoria RAM de la posición 20h a la 2Fh

```
        movlw   0x20    ;inicializar apuntador
        movwf   FSR     ;a la ram
NEXT    clrf    INDF    ;borrar registro INDF
        Incf    FSR,F   ;incrementar apuntador
        btfss   FSR,4   ;ya terminó?
        Goto    NEXT
```

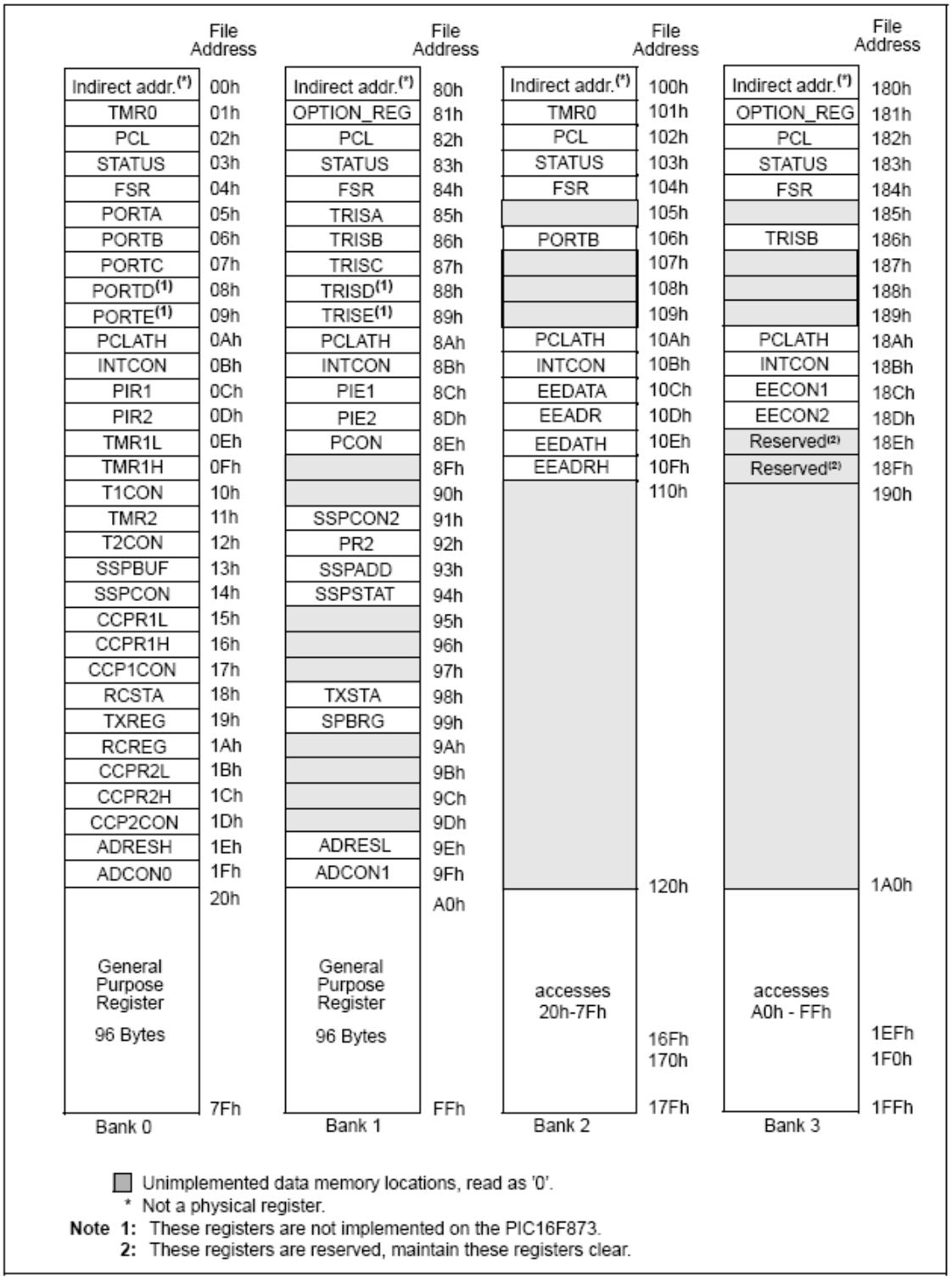
GURE 2-4: PIC16F874/873 REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	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(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	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	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h				
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
CCPR2L	1Bh		9Bh				
CCPR2H	1Ch		9Ch				
CCP2CON	1Dh		9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh				
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes	7Fh	General Purpose Register 96 Bytes	FFh	accesses 20h-7Fh	16Fh 170h 17Fh	accesses A0h - FFh	1EFh 1F0h 1FFh

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873.
2: These registers are reserved, maintain these registers clear.

FIGURE 2-4: PIC16F874/873 REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.

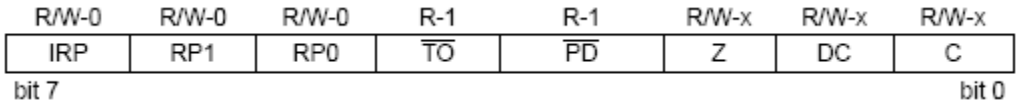
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873.

2: These registers are reserved, maintain these registers clear.

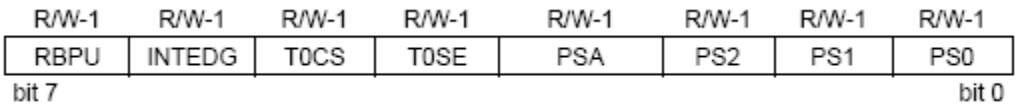
2. REGISTROS DE FUNCIONES ESPECIALES

2.1 REGISTRO DE STATUS



- ESTÁ EN LOS 4 BANCOS : 03H, 83H, 103H, 183H
- **PD-**: SE PONE A 0 AL EJECUTARSE INSTR. SLEEP
- **TO-**: SE PONE EN 0 EN OVERFLOW DE WDT
- **IRP**: SELECCIONA RAM EN MODO INDIRECTO
  0: BANCO 0 y 1 (000H-0FFH)
  1: BANCO 2 y 3 (100H-1F0H)

2.2 REGISTRO DE OPTION

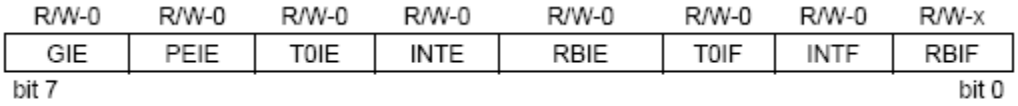


- **RBPU-:**HABILITACIÓN DE RESISTENCIAS PULL-UP EN PORTB
- **INTEDG**: FLANCO DE INTERRUP. (0-BAJADA, 1- SUBIDA). Pin RB0/INT
- **T0CS**: RELOJ DE TMR0 (0-RELOJ INTERNO, 1- RELOJ EXTERNO). Pin RA4/TOCKIN
- **TOSE**: FLANCO PARA EL RELOJ (0-SUBIDA,1- BAJADA). Pin RA4/TOCKIN
- **PSA**: PREESCALAMIENTO (0 -TMR0, 1- WDT)
- **PS2,PS1,PS0**: DEFINE EL VALOR DE LA PRESECALA SEGÚN LA SIGUIENTE TABLA:

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

2.3 REGISTRO INTCON

- ES EL REGISTRO UTILIZADO PARA PROGRAMAR INTERRUPCIONES Y APARECE EN LAS POSICIONES DE LA RAM EN OOBH, 08BH, 10BH 18BH
- IGUAL AL 16F84, SÓLO CAMBIA EL BIT PEIE POR EEIE.

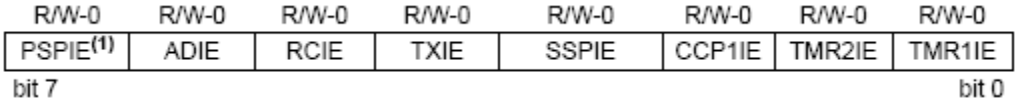


R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

- **GIE:** BIT DE INTERRUPCIÓN GLOBAL
- **PEIE**: INTERRUPCIÓN DE PERIFÉRICOS
- **T0IE**: INTERRUMPE CON OVERFLOW DEL TMR0
- **INTE**: INTERRUPCIÓN EXTERNA POR RB0/INT
- **RBIE**: INTERRUPCIÓN POR CAMBIO EN RB7:RB4
- **T0IF**: BANDERA DE INTERRUPCIÓN DEL TMR0
- **INTF**: BANDERA DE INTERRUPCIÓN DEL RB0/INT
- **RBIF**: BANDERA DE INTERRUPCIÓN DEL RB

2.4 REGISTRO PIE1

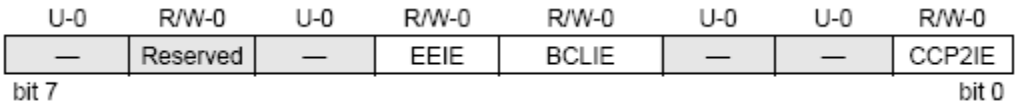
- ESTÁ EN LA POSICIÓN 8CH (BANCO 1)
- HABILITA O PROHIBE INTERRUPCIÓN EN PERIFÉRICOS. SE REQUIERE QUE PEIE = 1
-



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

- **PSPIE**: PTO PARALELO ESCLAVO(16F874)
- **ADIE**: AL TERMINAR LA CONVERSIÓN EN A/D
- **RCIE**: RECEPTOR USART CUANDO SE LLENA EL BUFFER
- **TXIE**: TRASM. USART CUANDO BUFFER SE VACÍA
- **SSPIE**: PUERTO SERIE SÍNCRONO
- **CCP1IE**: CUANDO HAY CAPTURA O COMP. POR CCP1
- **TMR2IE**: CUANDO HAY OVERFLOW EN TIMER2
- **TMR1IE**: CUANDO HAY OVERFLOW EN TIMER1

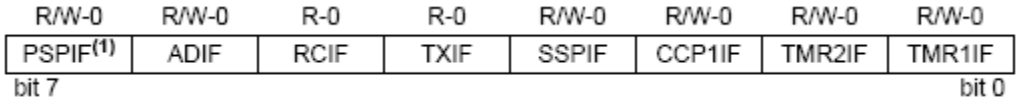
2.5 REGISTRO PIE2



U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	Reserved	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

- ESTÁ EN LA POSICIÓN 8CH (BANCO 1)
- HABILITA O PROHIBE INTERRUPCIÓN EN PERIFÉRICOS. SE REQUIERE QUE PEIE = 1
- **EEIE**: FIN DE ESCRITURA DE LA EEPROM
- **BCLIE**: SSP- COLISIÓN DE BUS CUANDO DOS MAESTROS TRATAN DE TRANSFERIR AL MISMO TIEMPO
- **CCP2IE**: CUANDO HAY CAPTURA O COMP. POR CCP2

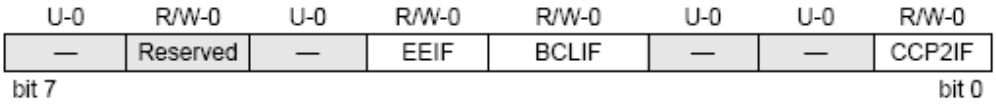
2.6 REGISTRO PIR1



BANDERAS DE INTERRUPCIÓN DE PERIFÉRICOS

- **PSPIF**: PTO PARALELO ESCLAVO(16F874)
- **ADIF**: AL TERMINAR LA CONVERSIÓN EN A/D
- **RCIF**: RECEPTOR USART CUANDO SE LLENA EL BUFFER
- **TXIF**: TRASM. USART CUANDO BUFFER SE VACÍA
- **SSPIF**: PUERTO SERIE SÍNCRONO
- **CCP1IF**: CUANDO HAY CAPTURA O COMP. POR CCP1
- **TMR2IF**: CUANDO HAY OVERFLOW EN TIMER2
- **TMR1IF**: CUANDO HAY OVERFLOW EN TIMER1

2.7 REGISTRO PIR2



BANDERAS DE INTERRUPCIÓN DE PERIFÉRICOS

- **EEIF**: FIN DE ESCRITURA DE LA EEPROM
- **BCLIF**: SSP- COLISIÓN DE BUS CUANDO DOS MAESTROS TRATAN DE TRANSFERIR AL MISMO TIEMPO
- **CCP2IF**: CUANDO HAY CAPTURA O COMP. POR CCP2

2.8 REGISTRO PCON

ES EL REGISTRO DE CONTROL DE POTENCIA

- **POR**: STATUS DEL POWER-ON-RESET (0= OCURRIÓ UN **POR**)
- **BOR**: STATUS DEL BROWN-OUT-RESET (0= OCURRIÓ UN **BOR**)

Power-On Reset (POR). Ocurre cuando se detecta la subida de VDD y genera un nivel aceptable ocn la temporización de 72 msg que hace el Power_up Timer (PWRT)

Oscillator Start-up Timer (OST). Provee un retardo de 1024 ciclos después que el PWRT es habilitado. Ayuda a estabilizar el oscilador de cristal o resonador.

Brown-out Reset (BOR). Si el VDD cae por debajo de 4V se resetea el micro.

3. CONFIGURACIÓN DE LOS PUERTOS DE E/S

- El 16F873/876 tiene tres puertos de entrada/salida: Puerto A de 6 bits (RA5:RA0), Puerto B de 8 bits (RB7:RB0), y el Puerto C de 8 bits (RC7:RC0).

- El 16F874/877 tiene cinco puertos de entrada/salida: Puerto A de 6 bits (RA5:RA0), Puerto B de 8 bits (RB7:RB0), el Puerto C de 8 bits (RC7:RC0), el Puerto D de 8 bits (RD7:RD0), y el Puerto E de 3 bits (RE2:RE0).

- Los registros de dirección del puerto se hacen a través del correspondiente registro TRIS: TRISA, TRISB, TRISC, TRISD, TRISE. Si el bit de TRIS =1 corresponde a una entrada y si el bit TRIS =0 es de salida. Ejemplo: TRISB =10010010, entonces, RB0(salida), RB1(entrada), RB2(salida), RB3(salida), RB4(entrada), RB5(salida), RB6(salida), RB7(entrada).

- El puerto A tiene configuración especial. Primero se programa los pines como E/S a través del registro ADCON1 en los bits: PCFG3, PCFG2, PCFG1, PCFG0

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

PCFG3: PCFG0	AN7[1] RE2	AN6[1] RE1	AN5[1] RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refc[2]
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog Input D = Digital I/O

Ejemplo: Configuración del puerto A

```
bcf     STATUS, RP0
bcf     STATUS, RP1     ;banco0
```

```
clrf    PORTA       ;borra latches de salida
bsf     STATUS, RP0 ;banco1
movlw   0x06        ,configura todos los pines
movwf   ADCON1      ;como entradas digitales
movlw   0xCF        ;inicializa dirección del dato
movwf   TRISA       ,pone RA<3:0> como entradas
                    ;y RA<5:4> como salidas
;TRISA<7:6> siempre son leídos como ceros
```

PUERTO PARALELO ESCLAVO PSP

- No está implementado en el 16F873/876
- El puerto D opera como puerto paralelo esclavo o como puerto de E/S a través del bit TRISE<PSPMODE> = 1
- Puede ser leíble o escribible asicrónicamente mediante los bits de control RE0/RD-, RD1/WR-, RE2/CS-. Los correspondientes bits de dirección TRISE<2:0> deben ser configurados como entradas y RE2:RE0 como E/S digitales en el registro ADCON1<3.0>. Como el aparato externo (microcontrolador maestro) controla la dirección del dato TRISD es ignorado.

- **REGISTRO TRISE**

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	Bit2	Bit1	Bit0
bit 7							bit 0

Input buffer full:
IBF. UNA PALABRA SE HA RECIBIDO Y ESOERA QUE LA LEA EL CPU
Output buffer full:
OBF. BUFFER DE SALIDA MANTIENE LA PALABRA ESCRITA
Input buffer overflow:
IBOV. SE ESCRIBIÓ CUANDO AÚN NO SE HABÍA LEÍDO LA PALABRA
Parallel slave port mode:
PSPMODE. PUERTO D EN MODO PSP =1, MODO DE E/S = 0
BIT2, BIT1, BIT0. CONTROL DE DIRECCIÓN RE2, RE1, RE0 (1 =ENTRADA, 0 =SALIDA)

4. CONJUNTO DE INSTRUCCIONES

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

5. PUERTOS DE E/S

PUERTO B:

- **TIENE RB7..RB0 COMO ENTRADAS DIGITALES BIDIRECCIONALES**
- **DISPONEN DE UNA RESISTENCIA INTERNA DE PULL-UP AL POSITIVO DE LA ALIMENTACIÓN QUE SE CONECTA CUANDO EL BIT RBPU- DEL REG-OPTION ES 0. EN RESET SE DESCONECTAN ESTAS RESISTENCIAS.**
- **RB7..RB4: PUEDEN PROGRAMARSE PARA GENERAR INTERRUPCIÓN CUANDO UNA DE ELLAS CAMBIA DE ESTADO.**
- **RB0/INT: PUEDE SER PROGRAMADO PAR GENERAR INTERRUPCIÓN EXTERNA.**

PUERTO C:

- **RC0/T1OSO/T1CKI: E/S DIGITAL/ SALIDA DE TMR1/RELOJ DE TMR1**
- **RC1/T1OSI/CCP2: E/S DIGITAL/ ENTRADA OSC TMR1/ENTRADA A CCP2; SALIDA DE COMPARADOR2; SALIDA PWM2.**
- **RC2/CCP1: E/S DIGITAL/ENTRADA A CCP1; SALIDA COMPARADOR1;SALIDA PWM1.**
- **RC3/SCK/SCL: E/S DIGITAL/RELOJ MODO SPI/RELOJ EN MODO I2C**
- **RC4/SDI/SDA: E/S DIGITAL,/ENTRADA MODO SPI/ENTRADA MODO I2C**
- **RC5/SDO: E/S DIGITAL/SALIDA MODO SPI**
- **RC6/TX/CK: E/S DIGITAL/TRANSM. USART/RELOJ TRANSM. SERIE SÍNCRONA**
- **RC7/RX/DT: E/S DIGITAL, RECEPCIÓN USART/TRANSM. SERIE SÍNCRONA**

PUERTO D:

- **RD7..RD0: BIDIRECCIONALES CON ENTRADA EN TRIGGER-SCHMITT.**
- **ADEMÁS SE PUEDE IMPLEMENTAR COMO PUERTO PARALELO ESCLAVO (PSP) PARA COMUNICACIÓN PARALELA**

PUERTO E:

- **RE0/RD-/AN5: E/S DIGITAL/LECTURA DE PSP/CANAL5 A/D**
- **RE1/WR-/AN6: E/S DIGITAL/ESCRITURA DE PSP/CANAL6 A/D**
- **RE2/CS-/AN7: E/S DIGITAL/ CHIP SELECT DE PSP/CANAL7 A/D**
- **PARA ACTIVAR EL PSP SE DEBE PONER A 1 EL BIT PSPMODE DEL REG-TRISE**

6. EJERCICIOS: UNIDAD TEMÁTICA I

1. Leer un teclado sencillo de 8 interruptores conectados en el puerto B en un display de 7 segmentos conectados en el puerto A mediante un decodificador (R13:RA0). Si no hay tecla oprimida desplegar FFh, poner retardo de 100mseg en el despliegue de la tecla.

2. Lectura de un teclado matricial de 4x4 en un display de 7 segmentos que usa decodificador conectado en RB7:RB4. Las filas de teclado están conectadas a un puerto de salida RA3:RA0 y las columnas a un puerto de entrada RB3:RB0. Características: a) Después de la detección de la tecla pulsada eliminar antirrobote generando un retardo de 100 msg b) desplegar tecla con retardo de 200msg

3. Lectura de 4 teclas de un teclado matricial en 4 display de 7 segmentos multiplexando cada display cada display 3 msg sin usar decodificador. Características: a) Seleccionar el display activando un transistor conectado en cátodo común b) Utilizando tabla de valores de decodificación de los 7 segmentos usando las instrucciones *addwf PCL* y *retlw K*

4. Hacer circular un mensaje en un display LCD utilizando bus de datos de 8 bits. El mensaje se encuentra en una tabla en forma de caracteres ASCII. Mensaje “INGENIERIA ELECTRONICA USCO”.

5. Repetir ejercicio anterior con las siguientes características: a) Bus de datos de 4 bits b) Mensaje fijo.

6. Realizar un semáforo programable por medio de dos interruptores como se indica en la siguiente tabla:

INTERRUPTORES		SEMAFORO (segundos)		
I2	I1	VERDE	AMARILLO	ROJO
0	0	5	2	5
0	1	5	2	10
1	0	8	2	8
1	1	10	2	5

   Interruptores conectados en el puerto A: **I1** en RA0, **I2** en RA1
   Leds del semáforo en el puerto B: **A** en RB1, **V** en RB2, **R** en RB3

7. Realizar un dado electrónico en un display de 7 segmentos conectado en RB7:RB0 utilizando un pulsador en RA0 (en reposo RA0=0). Cada vez que se presione el pulsador se produce un número aleatorio entre 1 y 6

8. Hacer un temporizador programable (0 – 7 seg) utilizando 3 interruptores (RA2:RA0) y un display de 7 segmentos (RB7:RB0). Un pulsador en RA4 comienza a contar del tiempo.

7. ENSAMBLADOR

Para realizar un programa en lenguaje ensamblador, además de las instrucciones propias del microcontrolador se requiere el uso de las directivas del ensamblador a utilizar. Par los micros de la microchips se utiliza el MPASM.

7.1 DIRECTIVAS - MPASM

Son comandos del ensamblador que aparecen en el código fuente pero no son trasladados directamente a códigos de operación. Son usados para controlar el ensamblador.

a) DIRECTIVAS DE CONTROL

Controlan cómo el código es ensamblado.

CONSTANT: Declara un símbolo constante

CONSTANT <label> [=<expresión>]

Ejemplo: CONSTANT longbuffer =512

#DEFINE: Define sustitución de texto

#DEFINE <nombre> [<argumento>, <valor>]

Ejemplo: #DEFINE control CONTA, 5
…….
…….
bsf control

END: Fin de programa

EQU: Define constante de ensamblaje

<label> EQU <expresión>

#INCLUDE: Incluye archivo fuente adicional

#INCLUDE “<incluye.file>”

Ejemplo: #INCLUDE “C:/ sys/ archivo. inc”

ORG: Pone origen de programa

<label> ORG <expresión>

PROCESSOR: Pone tipo de procesador

Ejemplo: PROCESSOR 16F873

RADIX: Especifica la raíz por defecto (Hex, Dec, Octal)

Ejemplo: RADIX Hex

SET: Define una variable en ensamblador

<label> SET <expresión>

```
Ejemplo:    Ancho     SET   0x12
            Long      SET   0x14
            Area      SET   Log*Ancho
```

VARIABLE: Declara un símbolo variable

VARIABLE <label> =<expresión>

Ejemplo: VARIABLE LongRegistro =64

b) DIRECTIVAS CONDICIONALES

Permite ensamblar código de secciones condicionales.

ELSE: Alternativa de bloque con IF
ENDIF: Fin de bloque condicional con IF
ENDW: Fin de bloque con WHILE
IF: Comienzo de bloque condicional
IFDEF: Ejecuta si el símbolo está definido
Ejemplo:
```
#DEFINE  testeo 1
……
……
IFDEF     testeo
……
ENDIF
```
WHILE: Realiza un loop si la condición es verdadera
Ejemplo:
```
PRUEBA        macro        cuenta
 VARIABLE     i
 i =0
 WHILE  i < cuenta
  i = i+1
 ENDW
 ENDM
START
 PRUEBA       5
 END
```

```
Ejemplo:    VARIABLE  temp =1
            IF        temp = 0
                      movlw       0x0A
```

```
        ELSE
                movlw       0x1E

        ENDIF
        #DEFINE     test
        IFDEF       test
                movlw       0x01
        IFNDEF      test
                movlw       0x02
        ENDIF
```

c) DIRECTIVAS MACROS

ENDM: Fin de definición de macro
EXITM: Salir de una macro

Ejemplo:
```
PRUEBA      macro       archivo
IF      archivo == 1
exitm
else
error   “archivo dañado”
endif
endm
```

LOCAL: Declara variable local en macro

Ejemplo:
```
long        equ     10
Tamaño      equ     20
Prueba      macro       tamaño
Local       long, label
Long        set     tamaño
Label       res     long
Endm
```

MACRO: Declara definición de macro

```
<label>     macro       [<arg>,….<arg>]
```

Ejemplo:
```
MULTIPLY    macro       arg1,dest
LOCAL       i = 0
movlw       arg1
movwf       multiplicar
WHILE       i < 8
addwf       dest
;colocar código aquí
;repetir 8 veces
i += 1
ENDW
ENDM
```

d) DIRECTIVAS DE DATOS

Controlan posiciones de memoria.

CBLOCK: Define un bloque de constantes
--CONFIG: Pone fusibles de configuración
DATA: Crea dato numérico y texto
Ejemplo: DATA 1, 2, label

DB: Declara dato de un byte
DE: Declara dato de eeprom
DT: Define tabla
DW: Declara dato de una palabra
ENDC: Fin de bloque
RES: Reserva memoria
Ejemplo: buffer RES 64

e) DIRECTIVAS DE LISTAMIENTO

ERROR: Error en mensaje
Ejemplo: if ARG >= 55
Error ”fuera de rango”
Endif

LIST: Lista opciones
MESSG: Mensaje definido por el usuario
Messg “<texto del mensaje>”
PAGE: Inserta página en el listado
SPACE: Inserta líneas blancas
Ejemplo: space 3 ;inserta 3 líneas blancas
SUBTITTLE: Especifica subtítulo en el programa
Subtittle “<texto del subtítulo>”
TITTLE: Especifica título del programa
Tittle “<texto del título>”

f) DIRECTIVAS DE ARCHIVO OBJETO

Son usadas esta directivas solamente cuando se crea el archivo objeto.

BANKISEL: Selecciona un banco de RAM con direccionamiento indirecto.
Ejemplo: movlw var
Movwf FSR
Bankisel var

Movwf INDF

BANKSEL: Selecciona banco de RAM
Ejemplo: banksel var
Movwf var

CODE: Empieza sección de código ejecutable

```
Ejemplo: reset      code   h'01FF'
         Goto       Start
```

EXTERN: Declara un label externo

```
Ejemplo: extern     function
         ------------------------
         Call       function
```

GLOBAL: Exporta un label definido

```
Global      <label>,……
```

PAGESEL: Genera selcción de código en la ROM

```
Ejemplo: pagesel    label
         Goto       label
```

7.2 EJEMPLOS - MPASM

;**EJEMPLO 1: PROGRAMACIÓN DE PUERTOS**

```
;LEER 4 INTERRUPTORES Y ENVIAR SU INFORMACIÓN A 4 LED's LOS
;INTERRUPTORES (N.A.=1) ESTÁN EN RA3:RA0 Y LOS LED's EN RB3:RB0

                LIST        P=16F873
                RADIX       HEX
                INCLUDE     "P16F873.INC"


            RESET     ORG       0x00   ;el vector de reset
            goto      INICIO             ;se salta al inicio del programa
;el programa empieza en la posición de memoria de programa 05H
                      ORG       0x05

INICIO      bsf       STATUS,RP0
            bcf       STATUS,RP1         ;banco1
            movlw     0xF0
            movwf     TRISB              ;RB3:RB0 entradas
            movlw      0x06
            movwf   ADCON1               ;Puerto A como e/s  digitales
            movlw     0x0F
            movwf     TRISA              ;RA3:RA0 salidas

; Programa principal
            bcf       STATUS,RP0         ;banco0
CICLO       movf      PORTA,W            ;leer puerto B
            xorlw     0xFF               ;invertir dato
            movwf     PORTB              ;enviar a puerto A
            goto      CICLO              ;repetir lectura
            END
```

; **EJEMPLO 2: RUTINA PARA BORRAR MEM. RAM DE LA 20h A LA 2Fh**

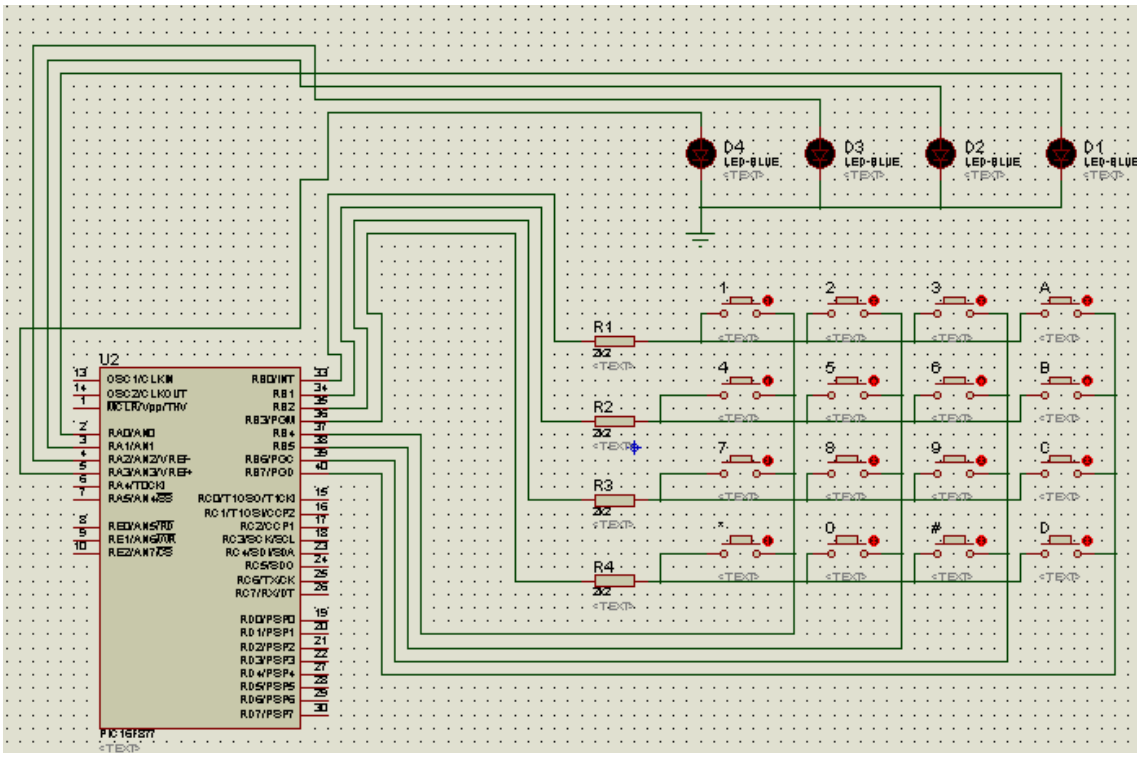
; DIRECCIONAMIENTO IDIRECTO USANDO LOS REGISTROS FSR e INDF

```
                LIST        P=16F873
                RADIX           HEX
                INCLUDE     "P16F873.INC"

;Declaración de registros que guardan las constantes
                CBLOCK      0x20
                            DATO1
                            DATO2
                            DATO3
                            DATO4
                            DATO5
                            DATO6
                            DATO7
                ENDC

        RESET   ORG         0x00        ;el vector de reset en 00h
                goto         INICIO     ;se salta al inicio del programa
                ORG         0x05        ;el programa empieza 0x5h

INICIO      movlw       0x01            ;carga de los valores a la RAM
            movwf       DATO1
            movlw       0x02
            movwf       DATO2
            movlw       0x03
            movwf       DATO3
            movlw       0x04
            movwf       DATO4
            movlw       0x05
            movwf       DATO5
            movlw       0x06
            movwf       DATO6
            movlw       0x07
            movwf       DATO7

BORRAR      movlw       0x20            ; apuntador en posición 20h
            movwf       FSR             ; apuntador dejarlo en FSR
OTRO        clrf        INDF            ; cargar reg INDF con 00h
            incf        FSR,1           ; incrementar posición de RAM
            btfss       FSR,4           ;saltar si FSR<4>=1
            goto        OTRO
            END
```

```
; EJEMPLO 3: REALIZAR UN CONTADOR DECIMAL DE 0 - 9
; EL PULSADOR ESTÁ CONECTADO A RB0, AL PULSAR SE PONE 0
; EL DECODIFICADOR DE BINARIO A 7 SEGMENTOS ESTÁ CONECTADO
; A RC3:RA0 (SALIDAS)

                LIST    P=16F873
                RADIX   HEX
                INCLUDE "P16F873.INC"

; Declaración de registros de trabajo
                CONTA   EQU  0x20       ;almacena el conteo
                LOOP1   EQU  0x21
                LOOP2   EQU  0x22
 ; direccionar memoria de programa
        RESET   ORG     0x00            ; el vector de reset 00h
                goto    INICIO          ; inicio de programa
                ORG     0x05

;Rutina de retardo 100 msg para fx=4 Mhz
RETARDO         movlw   D'100'
                movwf   LOOP1
        TOP1    movlw   D'110'
                movwf   LOOP2
        TOP2    nop
                nop
                nop
                nop
                nop
                nop
                decfsz  LOOP2
                goto    TOP2

; 9 ciclos x 110 usg=990 usg = 0.99 msg
                decfszLOOP1
                goto    TOP1
                retlw   0

;Programación de puertos
INICIO          bsf     STATUS,RP0
                bcf     STATUS,RP1      ;banco1
                movlw   0x01
                movwf   TRISB           ;RB0 entrada
                movlw   0xF0
                movwf   TRISC           ;RC3:RC0 salidas
                bcf     STATUS,RP0      ;banco0

;Programa principal
        BORRAR  clrf    CONTA
        CICLO   movf    CONTA,W
                movwf   PORTC           ;desplieque CONTA
```

```
                call    RETARDO
PULSA           btfsc   PORTB,0         ;salta si PORTB<0>=0
                goto    PULSA           ;espera que presione
                                        ;pulsador
                call    RETARDO
                btfsc   PORTB,0
                goto    PULSA           ;comprueba pulsador
                incf    CONTA
                movf    CONTA,W
                xorlw   0x0A
                btfsc   STATUS,Z        ;salta si Z=0 (W<0Ah)
                goto    BORRAR
                goto    CICLO
                END
```

7.3 MANEJO DE TECLADO MATRICIAL



; ARCHIVO RETARDO.INC

```
RETARDO   movlw   d'100'
          movwf   BUCLE1
TOP1      movlw   d'110'
          movwf   BUCLE2
TOP2      nop
          nop
          nop
          nop
          nop
          nop
```

```
            decfsz  BUCLE2
            goto    TOP2
            decfsz  BUCLE1
            goto    TOP1
            return
```

;ARCHIVO TECLAD.INC

```
TECLAD
TECL_SCAN                           ; Escanea el teclado
            clrf    TECLA           ;Borra Tecla y
            incf    TECLA,F         ;prepara Tecla para primer código.
            movlw       b'00001110' ;Saca 0 a la primera fila
            movwf       PORTB       ;de la Puerta B
            nop                     ;No operar para estabilizaci¢n de señal.
CHEQ_COLUM                          ;Chequeamos columnas
            btfss       PORTB,4             ;Primera columna = 0
            goto        ANTIREBOTES     ;Sale si se ha pulsado tecla.
            incf        TECLA,F         ;Si no tecla pulsada, incrementa tecla.
            btfss       PORTB,5         ;Segunda columna = 0
            goto        ANTIREBOTES     ;Sale si se ha pulsado tecla.
            incf        TECLA,F         ;Si no tecla pulsada, incrementa tecla.
            btfss       PORTB,6         ;Tercera columna = 0
            goto        ANTIREBOTES     ;Sale si se ha pulsado tecla.
            incf        TECLA,F         ;Si no tecla pulsada, incrementa tecla.
            btfss       PORTB,7         ;Cuarta columna = 0
            goto        ANTIREBOTES     ;Sale si se ha pulsado tecla.
            incf        TECLA,F         ;Si no tecla pulsada,incrementa Tecla.

CAMBIO      movlw       d'17'           ; Carga W con  de Teclas + 1.
            subwf       TECLA,W         ; y lo compara con el valor actual
            btfsc       STATUS,Z        ;No salta si no se ha pulsado ninguna
            goto        NOPULSADA       ;No ha sido pulsada ninguna tecla.
            bsf         STATUS,C        ;Pone a 1 el bit de carry.
            rlf         PORTB,f         ;Rota para escaneo a otra fila.
            goto        CHEQ_COLUM      ;Vuelve a revisar columnas

; Para apagar los led después de dejar de pulsar la tecla
NOPULSADA   clrf    PORTA
            goto    TECL_SCAN       ; Vuelve al inicio del programa
            Return
```

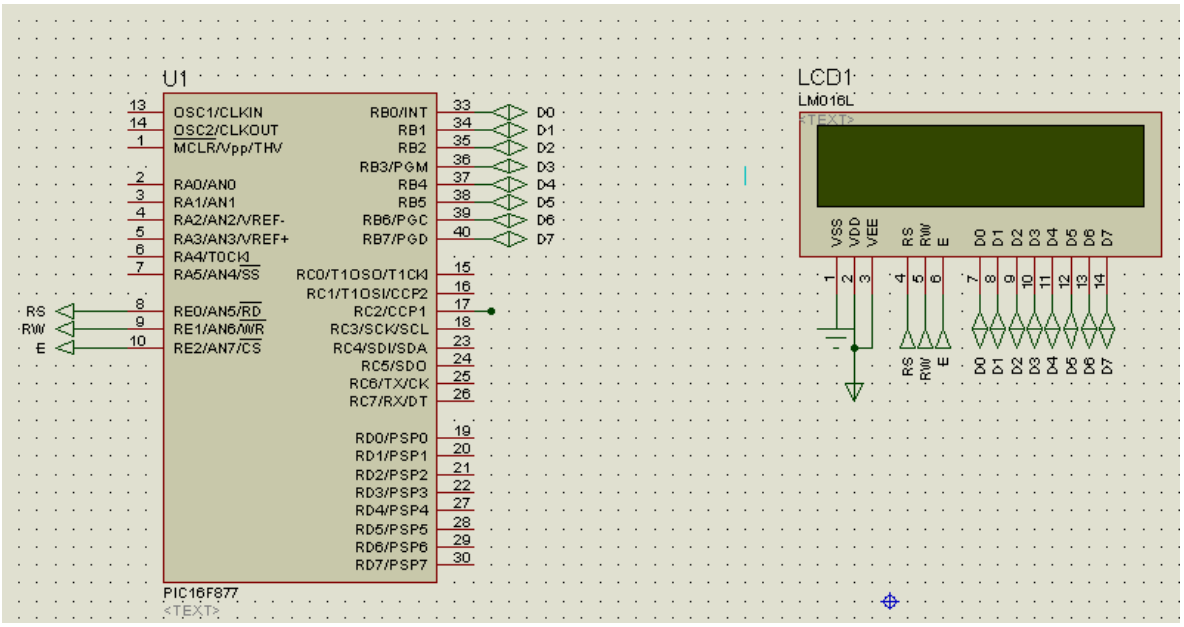
;PROGRAMA PRINCIPAL: TECLADO MATRICIAL ALFANUMERICO

```
; Programa RB0-RB3 como salidas y RB4-RB7 como entradas
; resistencias de polarización habilitadas.

        LIST            p = 16F877A
       INCLUDE          "p16F877A.inc"
       RADIX            hex
```

```
; registros de trabajo
                    TECLA       EQU   0x20
                    BUCLE1      EQU   0x21
                    BUCLE2      EQU   0x22

; iniciación del programa
                    ORG         0x00
                    goto        INICIO
                    ORG         0x05

IINCLUDE     "retardo.inc"
INCLUDE     "teclad.inc"

; Iniciación del programa
INICIO      bsf         STATUS,RP0        ;Selecciona Banco 1
            bcf         STATUS,RP1
            movlw       0x06
            movwf       ADCON1
            clrf        TRISA             ; Puero A como salidas
            movlw       b'11110000'       ; PB4-PB7 como entradas
            movwf       TRISB             ; PB0-PB3 como salidas
;Habilita R de polarizacion
            bsf         OPTION_REG,NOT_RBPU
            bcf         STATUS,RP0              ;Vuelve al banco 0.
            clrf        PORTA
            clrf        PORTB
            call        TECLAD

; ahora se espera a que la tecla sea soltada para evitar rebotes
ANTIREBOTES
            call    RETARDO
            movf    TECLA,W
            call    TABLA_CONV      ;llama a la tabla de conversión y retorna
            movwf   PORTA           ;con el valor lo pone en el puerto A.
            movwf   TECLA           ;con el valor en hexadecimal
            goto    TECL_SCAN       ;vuelve al inicio del programa principal.


TABLA_CONV
            addwf   PCL,1
            nop
            retlw    b'00000001'    ;Tecla nº1 = 1
            retlw   b'00000010'    ;Tecla nº2 = 2
            retlw   b'00000011'    ;Tecla nº3 = 3
            retlw   b'00001010'    ;Tecla nº4 = A
            retlw   b'00000100'    ;Tecla nº5 = 4
            retlw   b'00000101'    ;Tecla nº6 = 5
            retlw   b'00000110'    ;Tecla nº7 = 6
```

```
        retlw   b'00001011'     ;Tecla nº8 = B
        retlw   b'00000111'     ;Tecla nº9 = 7
        retlw   b'00001000'     ;Tecla nº10 = 8
        retlw   b'00001001'     ;Tecla nº11 = 9
        retlw   b'00001100'     ;Tecla nº12 = C
        retlw   b'00001110'     ;Tecla nº14 = 0
        nop
        retlw   b'00001111'     ;Tecla nº15 = #
        retlw   b'00001101'     ;Tecla nº16 = D
END
```

7.4 MANEJO DE DISPLAY LCD A 8 BITS



; RUTINA BANCO.INC

```
BANCO0 macro
        bcf     STATUS,RP0
        bcf     STATUS,RP1
endm
BANCO1 macro
        bsf     STATUS,RP0
        bcf     STATUS,RP1
endm
BANCO2 macro
        bcf     STATUS,RP0
        bsf     STATUS,RP1
endm
BANCO3 macro
        bsf     STATUS,RP0
        bsf     STATUS,RP1
endm
```

```
; RUTINA  RETARDO.INC
;RUTINA DE RETARDO DE n SEGUNDOS
SEGUNDOS
            movwf   TIEMPO
LOOP call   SEGUNDO
            decfsz  TIEMPO,F
            goto    LOOP
            return

;RETARDO DE 1 SEGUNDO
SEGUNDO
            movlw   d'10'
            movwf   CONTA1
LOOP1       call    RETAR_100_MS
            decfsz  CONTA1
            goto    LOOP1
            return

;RETARDO DE 100 ms
RETAR_100_MS
            movlw   d'100'
            movwf   CONTA2
LOOP3       movlw   d'110'
            movwf   CONTA3
LOOP4       nop
            nop
            nop
            nop
            nop
            nop
            decfsz  CONTA3
            goto    LOOP4
            decfsz  CONTA2
            goto    LOOP3
            return

;definiciones
                    #define ENABLE          bsf PORTE,2         ;Activa
                    #define DISABLE         bcf PORTE,2         ;Desactiva
                    #define LEER            bsf PORTE,1         ;Pone    LCD
en Modo RD
                    #define ESCRIBIR  bcf PORTE,1       ;Pone    LCD    en
Modo WR
                    #define COMANDO         bcf PORTE,0         ;Desactiva
RS (modo comando)
                    #define DATO            bsf PORTE,0         ;Activa    RS
(modo datos)
```

```
;RUTINAS DE LA    LCD.INC

LCD_BUSY
        BANCO0
        LEER            ;RW = 1
        BANCO1
        movlw 0xFF
        movwf TRISB
        BANCO0
        ENABLE              ;E = 1
        nop
        nop
L_BUSY
        btfsc   PORTB,7
        goto    L_BUSY
        DISABLE             ;E = 0
        BANCO1
        clrf    TRISB
        BANCO0
        ESCRIBIR    ;RW = 0
        return
LCD_E
        BANCO0
        ENABLE              ;E = 1
        nop
        nop
        nop
        DISABLE             ;E = 0
        return
LCD_DATO
        BANCO0
        COMANDO
        movwf PORTB
        call    LCD_BUSY
        DATO
        call    LCD_E
        return
LCD_REG
        BANCO0
        COMANDO                 ;RS = 0
        movwf PORTB
        call    LCD_BUSY
        call    LCD_E
        return
LCD_INI
        BANCO0
        movlw 0x38
        call    LCD_REG
        call    LCD_DELAY
        movlw 0x38
```

```
        call    LCD_REG
        call    LCD_DELAY
        movlw   0x38
        call    LCD_REG
        call    LCD_DELAY
        return
LCD_OFF
        movlw   b'00001000'
        call    LCD_REG
        return
LCD_BORRA
        movlw   0x01
        call    LCD_REG
        return
CUR_OFF
        movlw   0x0C            ; LCD on, cursor off.
        call    LCD_REG
        return
CUR_ON_SP                               ; Cursor ON (Si Parpadea)
        movlw   0x0F            ; LCD on, cursor on.
        call    LCD_REG
        return
CUR_ON_NP                               ; Cursor ON (No Parpadea)
        movlw   0x0E            ; LCD on, cursor on.
        call    LCD_REG
        return
LCD_IZQUIERDA
        movlw   0x10            ; Desplazamiento de cursor una posicion a la
izquierda.
        call    LCD_REG
        return
LCD_DERECHA
        movlw   0x14            ; Desplazamiento de cursor una posicion a la
izquierda.
        call    LCD_REG
        return
LINEA1
        movlw   0x02
        call    LCD_REG
        return
LINEA2
        movlw   0xC0
        call    LCD_REG             ; Coloca cursor en 2ª fila del LCD
        return
LCD_DELAY
        movlw   0x10
        movwf   TEMP1
        clrf    TEMP2
LCD_DELAY_1
        decfsz  TEMP2,F
```

```
        goto   LCD_DELAY_1
        decfsz TEMP1,F
        goto   LCD_DELAY_1
        nop
        nop
        return
```

; PROGRAMA PRINCIPAL LCD_8.ASM

```
;LIBRERIA PARA EL MANEJO DE LA LCD A 8 BITS
;USA PARA LOS BITS DE CONTROL EL PUERTO E
;Y PARA LOS BITS DE DATOS EL PUERTO B

                LIST   P=16F877
                RADIX        HEX
                INCLUDE     "P16F877.INC"
                INCLUDE "BANCOS.INC"

;CONFIGURACION DE BITS PARA EL PIC 16F877
            __CONFIG   _CP_OFF   &   _WDT_OFF   &   _BODEN_OFF   &
_PWRTE_ON   &   _XT_OSC   &   _WRT_ENABLE_OFF   &   _LVP_OFF   &
_DEBUG_OFF & _CPD_OFF



;registros de trabajo
                CBLOCK      0x20
CONTA1,CONTA2,CONTA3,MENSAJE,TIEMPO,TEMP1,TEMP2,TEMP3,CON
TEO
                ENDC

;INICIO DE LA MEMORIA DE  PROGRAMA

        org     0x00
        goto    INICIO
        org     0x05

        INCLUDE     "LCD.INC"
        INCLUDE     "RETARDOS.INC"
; TABLA PARA CONVERTIR EL CODIGO BCD A EL CODIGO  DE LOS
DISPLAYS 7 SEGMENTOS

ASCII
        addwf PCL,F
        DT      "0123456789"

;MENSAJES PARA VISUALIZAR EN LA LCD

MENSAJES
        movf   MENSAJE,W
```

```
        addwf PCL,F
        nop
        goto   MENSAJE1
        goto   MENSAJE2

MENSAJE1
        movf   CONTEO,W
        addwf PCL,F
        dt      "--UNIVERSIDAD---",0xFF,"-SURCOLOMBIANA--",0
MENSAJE2
        movf   CONTEO,W
        addwf PCL,F
        dt      "   INGENIERIA   ",0xFF,"  ELECTRONICA   ",0

;VISUALIZAR MENSAJES EN LA LCD

VISUALIZAR
        call    LCD_BORRA
        clrf    CONTEO
        clrf    TEMP1
        clrf    TEMP2
VISUAL
        call    MENSAJES
        movwf TEMP1
        movwf TEMP2
        movf   TEMP1,W
        btfss   STATUS,Z
        goto   NO_ES_CERO
        return
NO_ES_CERO
        movwf TEMP2
        xorlw  0xFF
        btfss   STATUS,Z
        goto   MISMA_LINEA
        call    LINEA2
        incf    CONTEO,F
        goto   VISUAL
MISMA_LINEA
        movf   TEMP2,W
        call    LCD_DATO
        incf    CONTEO,F
        goto   VISUAL

;PROGRAMA PRINCIPAL
INICIO
; CONFIGURACION DE BANCOS
        BANCO1
        movlw 0x06
        movwf ADCON1
        clrf    TRISA
```

```
        clrf    TRISC
        clrf    TRISB
        clrf    TRISD
        clrf    TRISE
        BANCO0
        clrf    PORTA
        clrf    PORTB
        clrf    PORTC
        clrf    PORTD
        clrf    PORTE

        call    LCD_INI
        call    LCD_BORRA
        call    CUR_OFF

NEW     movlw   D'1'
        movwf   MENSAJE
        call    VISUALIZAR

        movlw   D'3'
        movwf   TIEMPO
        call    SEGUNDOS
        call    LCD_BORRA

        movlw   d'2'
        movwf   MENSAJE
        call    VISUALIZAR

        movlw   d'3'
        movwf   TIEMPO
        call    SEGUNDOS
        call    LCD_BORRA

        goto    NEW

        END
```

7.5 MANEJO DE DISPLAY LCD A 4 BITS

8. MEMORIA EEPROM

- EEPROM DE DATOS HASTA 256 BYTES
- LA LECTURA-ESCRITURA SE REALIZA A TRAVÉS DE LOS REGISTROS **EECON1** (18CH) Y **EECON2** QUE NO ESTÁ IMPLEMENTADO FÍSICAMENTE Y SÓLO SE UTILIZA EN LA ESCRITURA.
- LA DIRECCIÓN SE COLOCA EN EL REGISTRO ESPECIAL **EEADR** Y EL DATO A ESCRIBIR O LEÍDO EN EL REGISTRO **EEDATA**

- ANTES DE INICIAR LA ESCRITURA SE ESCRIBE EN EECON2 PRIMERO EL DATO 55H Y LUEGO EL AAH.

REGISTRO EECON1:

EEPGD	---	---	---	WRERR	WREN	WR	RD

- EEPGD: SELECCIONA EL ACCESO A FLASH (1) O A LA EEPROM (0)
- WRERR: SEÑALA ERROR DE ESCRITURA
- WREN: HABILITA ESCRITURA
- WR: INICIA ESCRITURA
- RD: INICIA LECTURA

MANEJO DE MEMORIA EEPROM

```
        LIST        P=16F873
        RADIX       HEX
        INCLUDE     "P16F873.INC"

; Variables o regidtros de trabajo
        LOCAL_VAR   EQU   0x20      ;Direc. de comienzo var. locales
        CBLOCK      LOCAL_VAR
                    CONT            ;Contadores
                    CONT1
                    CONT2
                    CONT3
                    ADDR_H          ;MSB de direccion Flash
                    ADDR_L          ;LSB de direcc flash y eeprom
                    DATA_H          ;MSB dato flash
                    DATA_L          ;LSB dato flash y eeprom
                    DEBUG           ;Reg de debugging
            ENDC

            ORG     0x00
            goto    INICIO
            ORG     0x05

    ; Rutina de escritura en la eeprom
    ESCR_EEPROM
            movf    ADDR_L,W
            bsf     STATUS,RP1          ;Banco2
            movwf   EEADR               ;Posicion 010D
            bcf     STATUS,RP1          ;Banco0
            movf    DATA_L,W
            bsf     STATUS,RP1          ;Banco2
            movwf   EEDATA              ;Posicion 010C
            bsf     STATUS,RP0          ;Banco3
    ;   EECON1:EEPGD, -, -, -, WRERR, WREN2, WR, RD
```

```
            bcf     EECON1,EEPGD        ;Seleccia eeprom
            bsf     EECON1,WREN2        ;Inicia ciclo de escrit.
            bcf     PIR2,EEIF
            movlw   0x55
            movwf   EECON2
            movlw   0xAA
            movwf   EECON2
            bsf     EECON1,WR           ;Orden de escritura
            bcf     STATUS,RP0
            bcf     STATUS,RP1          ;Banco0
; PIR2: -, 0, -, EEIF, BCLIF, -, -,CCP2IF
ESPERA
            btfss   PIR2,EEIF
            goto    ESPERA
            bsf     STATUS,RP0
            bsf     STATUS,RP1          ;Banco3
            bcf     EECON1,WREN2        ;Deshabilita escritura
            return
```

; Rutina de lectura de la eeprom

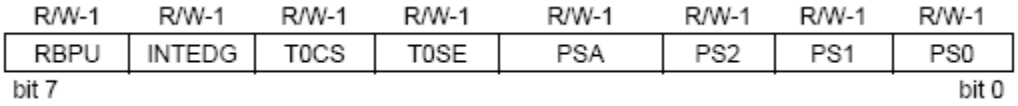
```
LEER_EEPROM
            movf    ADDR_L,W
            bsf     STATUS,RP1          ;Banco2
            movwf   EEADR
            bsf      STATUS,RP0         ;Seleccion de Banco3
            bcf     EECON1,EEPGD        ;Seleccion de EEPROM
            bsf     EECON1,RD
            bcf     STATUS,RP0          ;Banco2
            movf    EEDATA,W
            bcf     STATUS,RP1          ;Banco0
            movwf   DATA_L
            return
```

;Programa principal

```
INICIO
            bcf     STATUS,RP0
            bcf     STATUS,RP1          ;Banco0
            movlw   0x01
            movwf   ADDR_L              ;Direccion 01 a eeprom
            movlw   0x27
            movwf   DATA_L              ;Dato a eeprom=27
            call    ESCR_EEPROM
            bcf     STATUS,RP0
            bcf     STATUS,RP1          ;Banco0
            call    LEER_EEPROM         ;Dato lo deja en DATA_L
            nop
   END
```

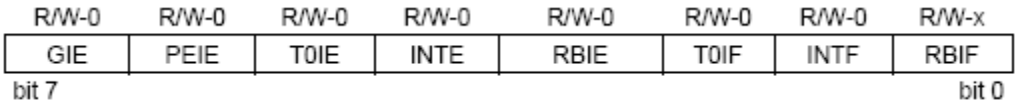
9. MANEJO DE INTERRUPCIONES

REGISTRO OPTION



R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

REGISTRO INTCON



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

; Rutina de interrupción externa pin RB0/INT

; Esta rutina incrementa un contador decimal en un display de 7 segmentos
; cada vez que el hay una interrupción por el pin RB0/INT

```
RESET       ORG   0x00          ; el vector de reset es la dirección 00h
            goto    INICIO      ; se salta al inicio del programa
            ORG     0x04        ; aquí se atiende la interrupción

        call    RETARDO         ;retardo para confirmar pulsador (100 msg)
        btfsc   PORTB,0         ;pregunta por el pin RB0
        goto    SALE            ;si no está  oprimido regresa
        btfss   INTCON,1        ;confirma si la interrupción fue por el pin INT
        goto    SALE            ;si no lo es sale
        incf    CONTA           ;incrementa el contador
        movf    CONTA,W         ;carga el registro W con el valor del conteo
        xorlw   0x0A            ;hace operación xor para ver si es igual a 0Ah
        btfsc   STATUS,Z        ;prueba si el contador llegó a 0Ah (diez)
        clrf    CONTA           ;si llegó a diez pasa el conteo a 0
        movf    CONTA,W         ;pasa el dato al display
        movwf   PORTA
        call    RETARDO         ;retardo de 100 milisegundos

SALE
        bcf     INTCON,1        ;la bandera de la interrupción se debe
                                ;poner en 0 antes de regresar (INTF)
        retfie          ;regresa al programa principal y habilita nuevamente
                        ;la interrupción al poner el bit GIE en 1

INICIO
        ; seleccionar banco1
        ; programar puertos
        movlw   0x80            ;en el registro OPTION sólo se programa
        movwf   OPTION_REG      ;el flanco de bajada para el pin INT
        ; seleccionar banco0
```

```
        movlw   0x90            ;en el registro INTCON se habilita la
        movwf   INTCON          ;interrupción por el pin INT
        clrf    CONTA           ;inicia contador en cero
        movf    CONTA,W         ;el valor del contador pasa al registro W
        movwf   PORTA           ;pasa el valor de W al puerto A (display)

CICLO
        nop                     ;espera que se presente una interrupción
        nop
        goto    CICLO
        END
```

10. TEMPORIZADORES

10.1 TIMER0

CARACTERÍSTICAS

- ES UN CONTADOR-TEMPORIZADOR DE 8 BITS
- LEIBLE Y ESCRIBIBLE
- RELOJ INTERNO Y EXTERNO
- SELECCIÓN DEL FLANCO EN EL RELOJ EXTERNO
- PREDIVISOR DE FRECUENCIA DEL RELOJ
- GENERACIÓN DE INTERRUPCIÓN EN EL OVERFLOW DE FFh A 00h PONE EL BIT TOIF = 1 DEL REG. INTCON. DEBE SER BOARRADO POR SOFTWARE ANTES DE SALIR DE LA INTERRUPCIÓN.
- REGISTROS ASOCIADOS: INTCON . OPTION_REG, TMR0

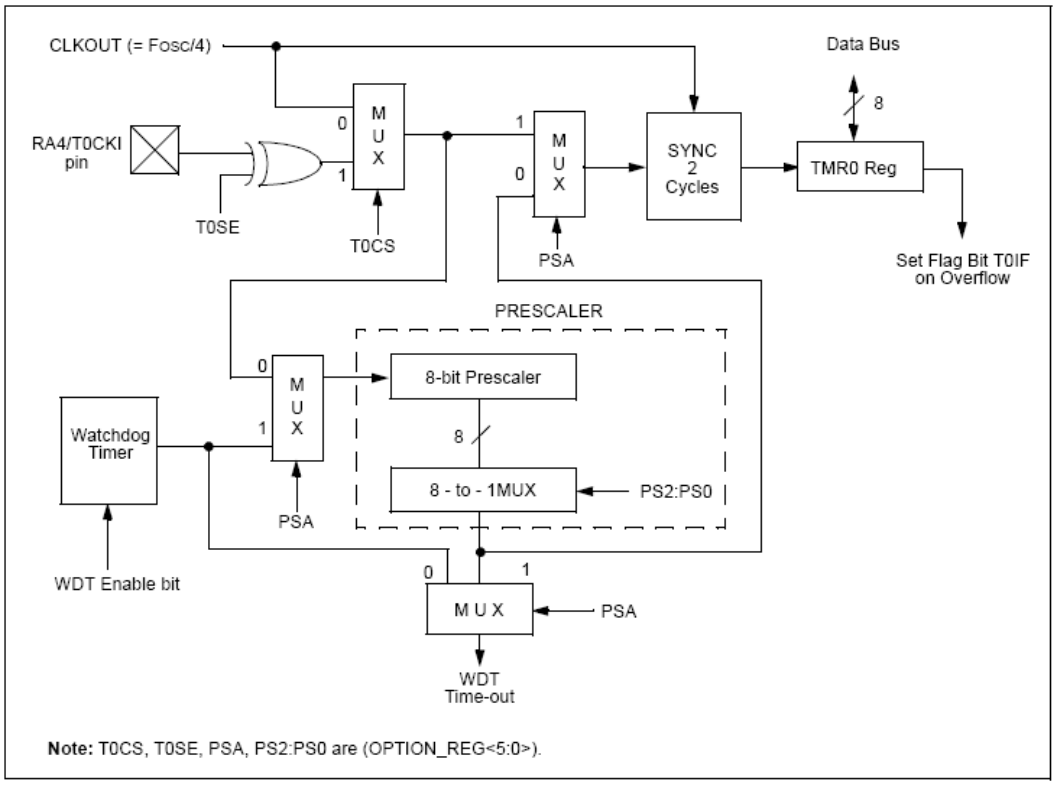
REGISTRO OPTION

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

- **RBPU-:**HABILITACIÓN DE RESISTENCIAS PULL-UP EN PORTB
- **INTEDG**: FLANCO DE INTERRUP. (0-BAJADA, 1- SUBIDA). Pin RB0/INT
- **T0CS**: RELOJ DE TMR0 (0-RELOJ INTERNO - *TEMPORIZADOR*, 1-RELOJ EXTERNO - *CONTADOR*). Pin RA4/TOCKIN
- **TOSE**: FLANCO PARA EL RELOJ (0-SUBIDA,1-BAJADA). Pin RA4/TOCKIN
- **PSA**: PREESCALAMIENTO (0 -TMR0, 1- WDT)
- **PS2,PS1,PS0**: DEFINE EL VALOR DE LA PRESECALA SEGÚN LA SIGUIENTE TABLA:

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

DIAGRAMA EN BLOQUES



10.2 TIMER1

CARACTERÍSTICAS

- ES UN CONTADOR-TEMPORIZADOR DE 16 BITS CON REGISTROS TMR1H:TMR1L
- CUANDO EL TIMER1 ES HABILITADO LOS PINES RC1/T1OSI/CCP2 Y RC0/T1OSO/T1CKI SON ENTRADAS
- EL MODO CONTADOR PUEDE SER SINCRONO O ASÍNCRONO
- EN MODO CONTADOR, EL MÓDULO DEBE TENER PRIMERO UN FLANCO DE BAJADA ANTES DE QUE EMPIECE A INCREMENTAR
- EN MODO CONTADOR EL RELOJ ESTÁ SOBRE EL PIN RC1/T1OSI/CCP2 SI TOSCEN = 1, O SOBRE EL PIN RC0/T1OSO/T1CKI SI TOSCEN = 0
- LEIBLE Y ESCRIBIBLE
- RELOJ EXTERNO PARA CONTADOR ES CON FLANCO ASCENDENTE
- SELECCIÓN DE RELOJ INTERNO Y EXTERNO
- PREDIVISOR DE FRECUENCIA
- INTERRUPCIÓN POR OVERFLOW FFFFh → OOOOh

- REGISTROS ASOCIADOS: INTCON (GIE,PEIE), PIR1 (TMR1IF), PIE1 (TMR1IE), TMR1L, TMR1H, T1CON

REG. T1CON

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

TICKPS1:T1CHPS0: SELECCIONA LA PRESCALA DEL RELOJ DE ENTRADA. 11= 1:8
10 = 1:4
01 = 1:2
00 = 1:1

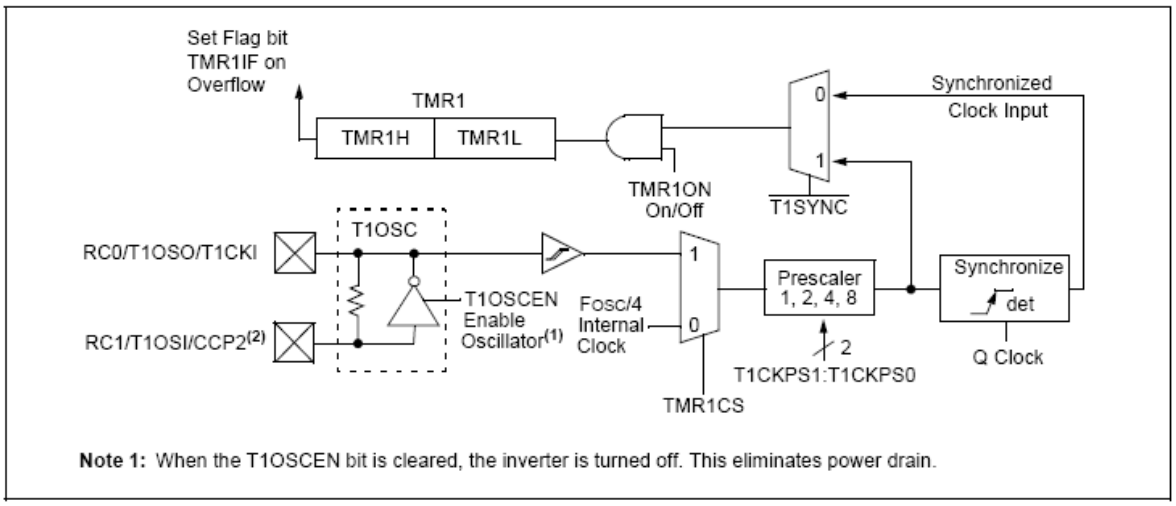
T1OSCEN: HABILITA OSCILADOR. 1= HABILITA, 0 = DESHABILITA

T1SYNC-: SINCRONIZA RELOJ EXTERNO (MODO CONTADOR). 1= NO SINCRONIZA, 0 = SINCRONIZA

TMR1CS: SELECCIONA FUENTE DEL RELOJ. 1 = EXTERNO (CONTADOR), 0 = INTERNO (TEMPORIZADOR Fos/4)

TMR1ON: HABILITA TIMER1

DIAGRAMA EN BLOQUES



10.3 TIMER2

CARACTERÍSTICAS

- ES UN TEMPORIZADOR DE 8 BITS
- DISPONE DE UN REGISTRO DE PERIODOS DE 8 BITS (PR2)
- LEIBLE Y ESCRIBIBLE

- PREDIVISOR DE FRECUENCIA PROGRAMABLE
- POSTDIVISOR DE FRECUENCIA PROGRAMABLE
- INTERRUPCIÓN AL COINCIDIR TMR2 Y PR2
- PUEDE SER USADO COMO BASE DE TIEMPO PAR EL MODO PWM
- TMR2 SE INCREMENTA DESDE 00h HASTA PR2 Y LUEGO PASA A 00h
- REGISTROS ASOCIADOS: INTCON (GIE, PEIE), PIR1 (TMR2IF), PIE1 (TMR2IE), T2CON, PR2

REG. T2CON

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

TOUTPS3:TOUTPS0: SELECCIONA POSTSCALA DE SALIDA

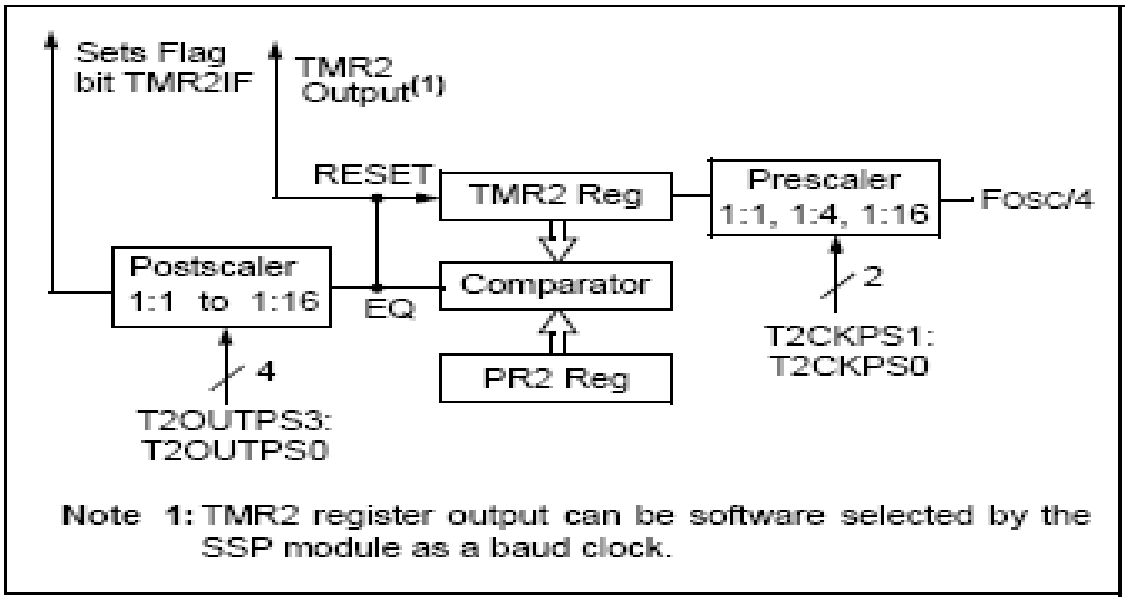
0000 = 1:1
0001 = 1:2
0010 = 1:3
•
•
•
1111 = 1:16

TMR2ON: HABILITA TIMER2. 1 = HABILITA, 0 = DESHABILITA

T2CKPS1:T2CKPS0: PRESCALA DEL RELOJ

00 = 1
01 = 4
1x = 16

DIAGRAMA EN BLOQUES



11. MÓDULOS CCP

CARACTERÍSTICAS

- CADA MÓDULO CCP1 Y CCP2 TIENE UN REGISTRO DE 16 BITS QUE PUEDE OPERAR COMO:
  - MODO CAPTURE
  - MODO COMPARE
  - MODO PWM
- LOS MÓDULOS CCP1 Y CCP2 SON IDÉNTICOS EXCEPTO EN LA OPERACIÓN "*Especial Event Trigger*"
- EL MÓDULO CCP1 TIENE EL REGISTRO DE TRABAJO *CCPR1*= *CCPR1H:CCPR1L* DE 16 BITS CONTROLADO POR EL REG *CCP1CON*. EL DISPARO ESPECIAL ES GENERADO POR UN COMPARE Y RESETEA EL TIMER1.
- EL MÓDULO CCP2 TIENE EL REGISTRO DE TRABAJO *CCPR2*= *CCPR2H:CCPR1L* CONTROLADO POR EL REG *CCP2CON*. EL DISPARO ESPECIAL ES GENERADO POR UN COMPARE, RESETEA EL TIMER1 Y ARRANCA LA CONVERSIÓN A/D.

MODOS DE OPERACIÓN

- LOS TIMER ASOCIADOS A LOS MODOS DE OPERACIÓN SON:
  - MODO CAPTURE → TIMER1
  - MODO COMPARE→ TIMER1
  - MODO PWM → TIMER2
- INTERACCIÓN ENTRE LOS DOS MÓDULOS:
  - CAPTURE, CAPTURE
  - CAPTURE, COMPARE
  - PWM, PWM
  - PWM, CAPTURE
  - PWM, COMPARE

REGISTRO CCP1CON

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

- **CCP1X:CCP1Y**: BITS MENOS SIGNIFICATIVOS DE PWM

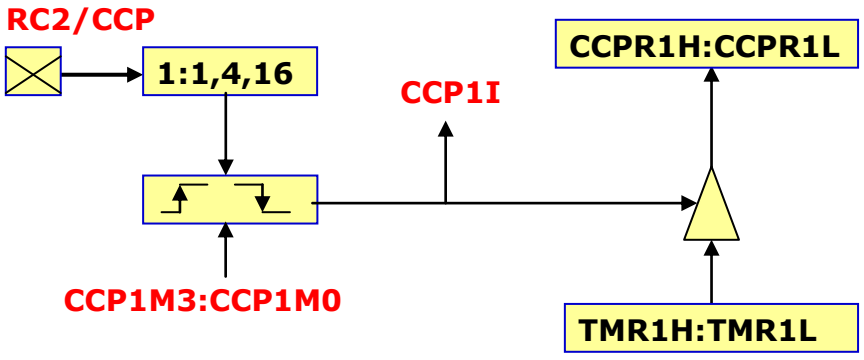
- **CCP1M3:CCP1M0**: SELECCIONA MODO DE CCP1
- 0000: CCP1 DESHABILITADO
- 0100: CAPTURE, CADA FLANCO DE BAJADA EN RC2/CCP1
- 0101: CAPTURE, CADA FLANCO DE SUBIDA EN RC2/CCP1
- 0110: CAPTURE, CADA 4º FLANCO DE SUBIDA EN RC2/CCP1
- 0111: CAPTURE, CADA 16º FLANCO DE SUBIDA EN RC2/CCP1
- 1000: COMPARE, SALIDA =1 (CCPIF =1)
- 1001: COMPARE, SALIDA =0 (CCPIF =1)
- 1010: COMPARE, SE GENERA INTERRUPCIÓN (CCPIF =1)
- 1011: COMPARE, DISPARO ESPECIAL, DIFERENTE A CCP2
- 11XX: PWM

11.1 MODO CAPTURE

CARACTERÍSTICAS

- EL REGISTRO CCPR1H:CCPR1L CAPTURA EL VALOR DE TMR1 (16 BITS) CUANDO OCURRE UN EVENTO EN EL PIN RC2/CCP1. CUANDO SE REALIZA LA CAPTURA CCP1IF=1 (REG PIR1). ESTE FLAG DEBE SER BORRADO POR SOFTWARE. EL PIN RC2/CCP1 DEBE SER CONFIGURADO COMO ENTRADA EN TRISC.
- EL TIMER1 DEBE ESTAR EN MODO TEMPORIZADOR O EN MODO CONTADOR SINCRÓNICO
- SI CCP1IE=1 SE GENERA INTERRUPCIÓN EN MODO CAPTURE
- CUANDO EL MODO CAPTURE ES CAMBIADO PUEDE GENERARSE UNA INTERRUPCIÓN FALSA, DEBE EVITARSE HACIENDO CCP1IE=0 Y CCP1IF=0
- PARA CAMBIAR PRESCALA SE RECOMIENDA HACER:
  - clrf CCP1CON
  - movlw NUEVA_ESCALA
  - movwf CCP1CON

DIAGRAMA EN BLOQUES

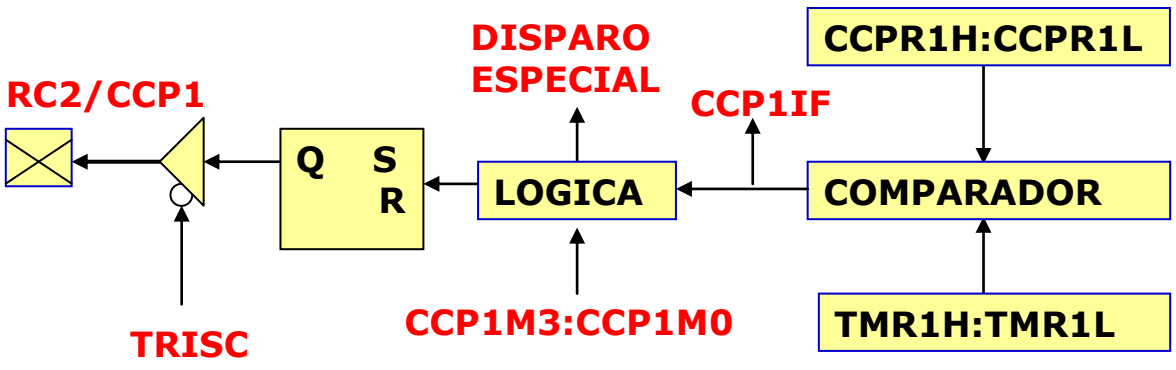


11.2 MODO COMPARE

CARACTERÍSTICAS

- EL REGISTRO CCPR1 SE COMPARA CONTINUAMENTE CON EL TMR1. CUANDO COINCIDEN LOS VALORES EL PIN RC2/CCP1 (CCP1IF=1):
  - PASA A NIVEL ALTO
  - PASA A NIVEL BAJO
  - NO CAMBIA, PERO PRODUCE INTERRUPCIÓN
  - GENERA DISPARO ESPECIAL
- EL PIN RC2/CCP1 DEBE ESTAR CONFIGURADO COMO SALIDA EN TRISC<2>
- EL TIMER1 DEBE ESTAR EN MODO TEMPORIZADOR O EN MODO CONTADOR SINCRÓNICO
- EL EVENTO DISPARO ESPECIAL LA SALIDA DE CCP1 RESETEA EL TIMER1 QUEDANDO CCPR1 COMO UN REGISTRO DE 16 BITS PROGRAMABLE
- EL EVENTO DISPARO ESPECIAL LA SALIDA DE CCP2 RESETEA EL TIMER1 Y ARRANCA LA CONVERSIÓN A/D SI ESTÁ HABILITADO. (ESTOS EVENTOS NO PONDRÁN CCPXIF =1)

DIAGRAMA EN BLOQUES



11.3 MODO PWM

CARACTERÍSTICAS

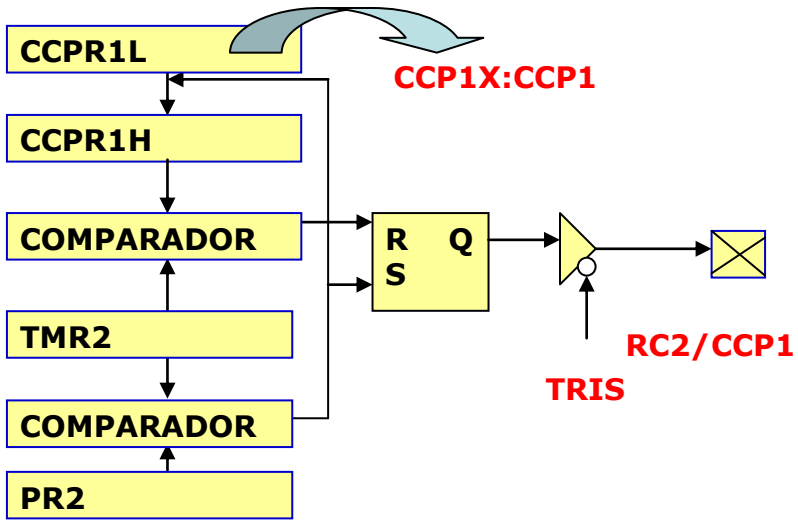
- EL PIN RC2/CCP1 (CONFIGURADO COMO SALIDA) PRODUCE UNA SALIDA PWM (PULSE WIDTH MODULATION) DE 10 BITS DE RESOLUCIÓN.
- LA SALIDA PWM TIENE UNA BASE DE TIEMPO (PERIODO) Y UN TIEMPO EN QUE LA SALIDA PERMANECE ALTA (CICLO DUTY).
- EL PERIODO DEPENDE DEL VALOR DE PR2 (8 BITS), DE Tosc DEL CRISTAL Y DE LA PRESCALA DE TMR2 SEGÚN:

PERIODO = [(PR2)+1]*4*Tosc*(PRESCALA TMR2)

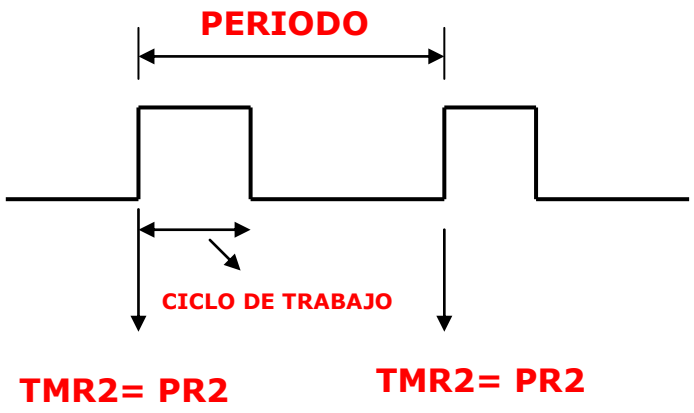
- CUANDO TMR2 = PR2, OCURRE LO SIGUIENTE:
  - TMR2 ES BORRADO
  - RC2/CCP1 = 1
  - EL VALOR DE CCPR1L SE CARGA EN CCPR1H
- LOS 10 BITS DE RESOLUCIÓN SE CARGAN EN CCPR1L (8 MSB) Y EN CCP1X:CCP1Y (CCP1CON)
- EL CICLO DE TRABAJO SE CALCULA CON LA ECUACIÓN:

DUTY = (CCPR1L:CCP1CON<5:4)*Tosc*(PRESCALA TMR2)

DIAGRAMA EN BLOQUES



SALIDA PWM



PASOS PARA CONFIGURAR MODO PWM:

1. PONER PERIODO PROGRAMADO EN PR2
2. PONER CICLO DE TRABAJO ESCRIBIENDO EN CCPR1L Y EN LOS BITS *CCP1X:CCP1Y* DEL REGISTRO *CCP1CON*
3. COLOCAR PIN *RC2/CCP1* COMO SALIDA (TRISC)
4. PONER PRESCALA DE TMR2 Y HABILITAR TMR2 EN *T2CON*
5. CONFIGURAR MÓDULO CCP1 PARA PWM

12. CONVERSOR ANÁLOGO/DIGITAL

12.1 CARACTERÍSTICAS

- TIENE 5 ENTRADAS PARA LA SERIE 16F873/876 Y 8 ENTRADAS PARA LOS 16F874/877.
- EL CONVERSOR GENERA UN RESULTADO DIGITAL DE LA ENTRADA ANÁLOGA VÍA APROXIMACIONES SUCESIVAS
- EL RESULTADO ES UN NÚMERO DIGITAL DE 10 BITS
- LOS VOLTAJE REFERENCIALES SON SELECCIONADOS POR SOFTWARE: *VDD, VSS, RA2 O RA3*
- TIENE CUATRO REGISTROS*: ADRESH, ADRESL* DONDE SE GUARDA EL RESULTADO DIGITAL DE LA CONVERSIÓN, *ADCON0 Y ADCON1* QUE SON LOS REGISTROS DE CONTROL
- LOS PINES DEL PUERTO PUEDEN SER CONFIGURADOS COMO ENTRADAS ANÁLOGAS O COMO E/S DIGITALES
- CUANDO LA CONVERSIÓN SE COMPLETA, EL RESULTADO SE CARGA EN *ADRESH:ADRESL*, EL BIT GO/DONE- ES BORRADO Y EL FLAG *ADIF = 1*

12.2 REGISTRO *ADCON0*:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

ADCS1:ADCS0 SELECCIONA RELOJ DE CONVERSIÓN
- 00 = 2Tosc
- 01 = 8Tosc
- 10 = 32Tosc
- 11 = FRC (RELOJ INTERNO RC)

CHS2:CHS0: SELECCIONA EL CANAL ANÁLOGO
- 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)
- 110 = canal 6, (RE1/AN6)
- 111 = canal 7, (RE2/AN7)

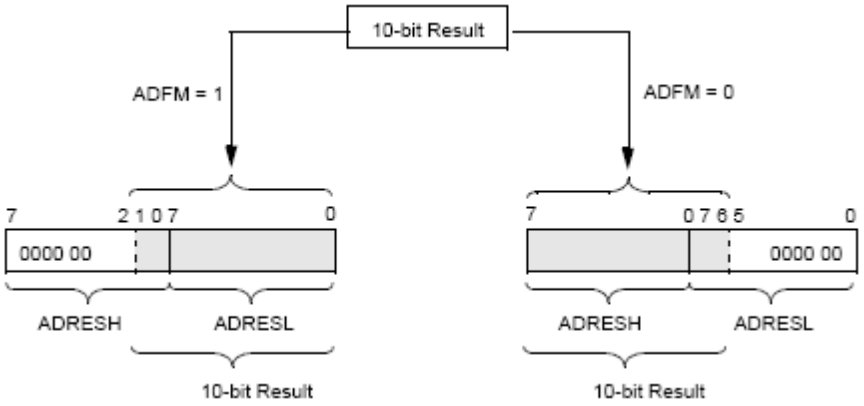
GO/DONE-: STATUS DE CONVERSIÓN. SI ADON = 1, ENTONCES,
- 1 = CONVERSIÓN EN PROGRESO
- 0 = NO HAY CONVERSIÓN

ADON: HABILITA CONVERSOR
- 1 = CONVERSOR OPERANDO
- 0 = CONVERSOR APAGADO

12. 3 REGISTRO *ADCON1*:

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

ADFM: FORMATO DEL RESULTADO
- 1= JUSTIFICACIÓN A LA DERECHA
- 0 = JUSTIFICACIÓN A LA IZQUIERDA



PCFG3:PCFG0: CONFIGURACIÓN DEL PUERTO

PCFG3: PCFG0	AN7[1] RE2	AN6[1] RE1	AN5[1] RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs[2]
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog input D = Digital I/O

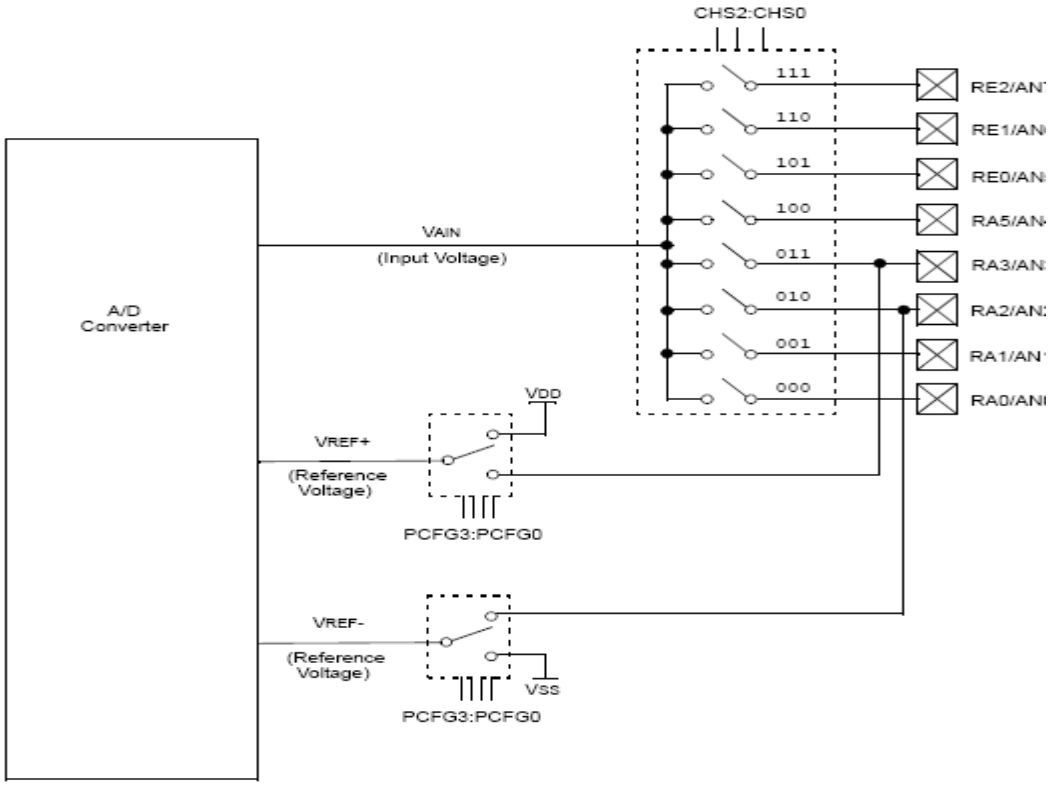
12.4 PASOS DE CONFIGURACIÓN

1. CONFIGURAR LAS ENTRADAS ANÁLOGAS, VOLTAJES DE REFERENCIA Y DIGITALES EN EL REGISTRO *ADCON1*, SELECCIONAR EL CANAL DE ENTRADA EN *ADCON0*, EL RELOJ DE CONVERSIÓN Y ACTIVAR EL A/D EN *ADCON0*
2. SI SE DESEA INTERRUPCIÓN CONFIGURARLA: ADIF = 0, ADIE = 1, PEIE = 1, GIE = 1.
3. ESPERAR TIEMPO REQUERIDO DE ADQUISICIÓN
4. ARRANCAR CONVERSIÓN GO/DONE- = 1
5. ESPERAR QUE SE COMPLETE LA INTERRUPCIÓN POR POLLING DE GO/DONE- = 0, O ESPERAR INTERRUPCIÓN
6. LEER EL RESULTADO EN ADRESH:ADRESL, BORRAR ADIF = 0
7. ESPERAR PRÓXIMA CONVERSIÓN

AD Clock Source (TAD)		Maximum Device Frequency
Operation	ADCS1:ADCS0	Max.
2TOSC	00	1.25 MHz
8TOSC	01	5 MHz
32TOSC	10	20 MHz
RC[1, 2, 3]	11	(Note 1)

Note 1: The RC source has a typical TAD time of 4 µs, but can vary between 2-6 µs.
2: When the device frequencies are greater than 1 MHz, the RC A/D conversion clock source is only recommended for SLEEP operation.
3: For extended voltage devices (LC), please refer to the Electrical Characteristics (Sections 15.1 and 15.2).

12.5 DIAGRAMA EN BLOQUES



12. EJEMPLOS

CONTROL DE UN MOTOR

MACROS.INC

```
;-------------------------------------------------------------------------------------------------
; macros de suma y resta de un valor de 8 bits a una palabra de 16 bits
;-------------------------------------------------------------------------------------------------
; Argumentos: v =valor a sumar/restar
;                MSB: LSB = direcciones de la palabra de 16 BITS
;                d =destino (w/f)

; Esta macro SUMA W al par de registros de 16 bits BINH: BINL
SUMA  MACRO v, LSB, MSB, d
            movlw   v
            addwf    LSB,d          ; LSB = LSB + w
            ADDCF   MSB,d       ; MSB++ si hay CARRY (directiva de MPLAB)
      ENDM

; Esta macro RESTA W al par de registros de 16 bits BINH:BINL
RESTA MACRO v, LSB, MSB, d
```

```
            movlw   v           ; Carga Argumento en W
            subwf   LSB,d       ; LSB = LSB - w
            btfss   STATUS,C    ; Hay BORROW?
            decf    MSB,d       ; MSB-- si hay BORROW
    ENDM
```

BCD-BINARIO.INC

```
;-------------------------------------------------------------------------------
;           RUTINA PARA CONVERSIÓN  DE BCD A BINARIO
;-------------------------------------------------------------------------------
; Convierte los registros de UNIDADES, DECENAS, CENTENAS y MILES (codificados en BCD)
; Recibe la cantidad binaria en los registros MILES, CENTENAS, DECENAS y UNIDADES.
; Devuelve la cantidad codificada en binario, en dos registros: BINH y BINL.
; Su capacidad es de 0000d a 9999d.

;REGISTROS DE TRABAJO NECESARIOS: (definidos en el archivo principal)
;       BINL:     LSB de la cantidad en binario
;       BINH:     MSB de la cantidad en binario
;       MILES:    Unidades de Mil
;       CENTENAS:  ...
;       DECENAS:   ...
;       UNIDADES:  ....

BCD_A_BIN
        movf    UNIDADES,w      ;Carga las UNIDADES en W,
        clrf    UNIDADES        ;las borra del registro
        movwf   BINL            ;y las guarda en el LSB
        clrf    BINH            ;Limpia el MSB

SUMAR_DEC
        clrw
        xorwf   DECENAS,w       ;Compara las DECENAS con 0 y deja resul. en w
        btfsc   STATUS,Z        ;Salta una linea si las decenas no son cero
        goto    SUMAR_CENT      ;Pasa a si no hay DECENAS (son 0)
        movlw   .10             ;Carga 10 -decimal- en w
        addwf   BINL,f          ;suma 10 al LSB
        decfsz  DECENAS,F       ;Decrementa decenas, salta si  se sumaron todas
        goto    $-2             ;Vuelve para sumar la siguiente decena

SUMAR_CENT
        clrw
        xorwf   CENTENAS,w      ;Compara las CENTENAS con 0
        btfsc   STATUS,Z        ;Salta una linea si las CENTENAS no son cero
        goto    SUMAR_MILES     ;Pasa si no hay CENTENAS (son 0)
```

```
SUMAR100
        SUMA    .100,BINL,BINH,f ;Suma 100 al par BINH:BINL
         decfsz  CENTENAS,F
         goto    SUMAR100                    ;Vuelve para la siguiente centena

SUMAR_MILES
        clrw
         xorwf   MILES,w             ;Compara los MILES con 0 y guarda resul. en w
         btfsc   STATUS,Z            ;Salta una línea si los MILES no son cero
         retlw   0                   ;Retorna si no hay MILES (son 0)
SUMAR1000
         SUMA    .250,BINL,BINH,f;
         SUMA    .250,BINL,BINH,f        ;Para sumar 1000, sumo 250x 4 veces
         SUMA    .250,BINL,BINH,f;
         SUMA    .250,BINL,BINH,f;
         decfsz        MILES,F           ; decrementa MILES
         goto    SUMAR1000           ;Vuelve para sumar la siguiente
         retlw   0                   ;Retorna luego de sumar todos los miles

;-------------------------------------------------------------------------------
;         RUTINA PARA CONVERSION DE BINARIO A BCD
;-------------------------------------------------------------------------------
;Recibe la cantidad Binaria BINH:BINL y entrega los dígitos BCD en UNIDADES,
DECENAS, ;CENTENAS, MILES. Su capacidad es de 0d a 9999d

BIN_A_BCD
         clrf    MILESM              ;borra registros de trabajo
         clrf    CENTENASM
         clrf    DECENASM
         clrf    UNIDADESM

RESTAR_MIL
        movf    BINHM,w
         movwf   TEMP                        ;Guarda MSB en TEMP
         RESTA   .250,BINLM,BINHM,f
         RESTA   .250,BINLM,BINHM,f  ;Para restar 1000, resto 250
         RESTA   .250,BINLM,BINHM,f  ;4 veces a BINHM:BINLM
         RESTA   .250,BINLM,BINHM,f  ;
         movf    BINHM,w
         subwf   TEMP,f               ;Hace TEMP = TEMP - W = TEMP - BINHM.
                                     ;Si al restar 1k hicimos BINHM<0, TEMP<0.
         btfss   STATUS,C            ;Salta si no hubo BORROW al restar esto
         goto    FIN_1000S           ;Hubo borrow, se restaron 1000 de más.
         incf    MILESM,f            ;No hubo borrow, incremente MILESM.
         goto    RESTAR_MIL          ;Sube a volver a restar 1000 y revisar...

FIN_1000S
```

```
        SUMA    .250,BINLM,BINHM,f  ;Sumamos 1000 para compensar
        SUMA    .250,BINLM,BINHM,f  ;que se restaron.
        SUMA    .250,BINLM,BINHM,f  ;
        SUMA    .250,BINLM,BINHM,f  ;

RESTAR_CIEN
        movf    BINHM,w
        movwf   TEMP                    ;Guarda LSB en TEMP
        RESTA   .100,BINLM,BINHM,f  ;Resta 100 a la palabra BINHM:BINLM
        movf    BINHM,w
        subwf   TEMP,f                  ;Hace TEMP=TEMP- W = TEMP - BINLM.
        ;Si al restar 100 hicimos BINLM<0, TEMP<0.
        btfss   STATUS,C                ;Salta si no hubo BORROW al restar esto
        goto    FIN_100S                ;Hubo borrow, se restaron 100 de más.
        incf    CENTENASM,f         ;No hubo borrow, incremente CENTENASM.
        goto    RESTAR_CIEN         ;Sube a volver a restar 100 y revisar...

FIN_100S
        SUMA    .100,BINLM,BINHM,f      ;+100 para compensar los -100 de más

RESTAR_DIEZ
        movlw   .10             ;No se usa la macro, la cantidad de DECENASM
        subwf   BINLM,f         ;máxima (100) nunca provocaría un acarreo.
        btfss   STATUS,C        ;Salta si no hubo BORROW al restar 10
        goto    FIN_10S         ;Hubo borrow, se restaron 10 de más.
        incf    DECENASM,f  ;No hubo borrow, incremente DECENASM.
        goto    RESTAR_DIEZ

FIN_10S
        addwf   BINLM,W         ;+10 para compensar los -10 de más
        movwf   UNIDADESM       ;Lo que queda en w Son las UNIDADESM
        clrf    BINLM           ;Limpia LSB
        return
```

ENTRADA DE RPM.INC

```
;-------------------------------------------------------------------------------
;   SUBRUTINA DE SOLICITUD DE ENTRADA DE RPM
;-------------------------------------------------------------------------------
;Recibe la entrada de usuario de la RPM. Si la entrada está fuera del rango de
;RPM del Motor, la subrutina hace TEMP = 255d, indicando error al retornar al
;programa; siendo este último el encargado de restablecer el valor anterior
;(almacenado en registros de respaldo UNIDADES2...MILES2) en la LCD e ignorar
;el valor erróneo. De no haber error retorna haciendo TEMP=00d, indicando
;operación exitosa.

RPM?    clrf    NTECLAS     ;Reinicia contador de teclas
        movlw   MILES       ;Carga dirección donde irán las unid. de mil
```

```
        movwf   FSR             ;Guarda dirección de direc. indirecto
        movlw   0xC0
        call    LCD_CTRL    ;Pasa el cursor de la LCD a la segunda línea.

;Para comenzar a introducir la RPM 'deseada' el usuario debe presionar OK (0x0A)

ESPERE_OK
        call    TECLADO             ;Espera tecla
        call    ANTIRREBOTE         ;Rutina Antirrebote
        movlw   0x0A        ;
        xorwf   TECLA,w             ;Tecla presionada = OK (0x0A)?
        BZ      ESPERE_MILES        ;Va a 'Esperar Miles' si la tecla es OK.
       ;La directiva BZ de MPLAB bifurca a 'k' si STATUS,Z=1
        goto    ESPERE_OK           ;Si no es OK, vuelve a ESPERE_OK.

ESPERE_MILES
        CCP1_PAUSA 1                ;Macro: Detiene la Captura de RPM
        movlw   0x0D        ;
        call    LCD_CTRL    ;Habilita cursor 'de sobreescritura' (BLINK) de la LCD
ESPERE_TCLA
        call    TECLADO             ;Espera tecla
        call    ANTIRREBOTE ;Antirrebote...
        movlw   0x0A        ;
        xorwf   TECLA,w     ;Tecla presionada = OK (0x0A)?
        BZ      OK_OK       ;Bifurca si la tecla es OK.
        movlw   0x0B        ;No era OK
        xorwf   TECLA,w     ;Tecla presionada = CANCEL (0x0B)?
        BZ      CANCEL      ;Bifurca si la tecla es CANCEL.
        movlw   0x04        ;La tecla es un número entre 0 y 9.
        xorwf   NTECLAS,w   ;NTECLAS=4? (ya digitó Miles, Cent.s, dec.s y unid.s?)
        BZ      ESPERE_TCLA ;Si es así, salte, ignorando el número digitado.
        incf    NTECLAS,f   ;NO, aún no termina, incremente el contador de teclas,
        movf    TECLA,w     ;Cargue el dígito y
        movwf   INDF        ;Almacénelo en el registro apuntado por FSR
        incf    FSR,f       ;Incremente apuntador de direccionamiento indirecto
        addlw   0x30        ;Sumamos 30 al dígito para convertirlo a ASCII
        call    LCD_DATO    ;Y poder sacarlo por la LCD mediante la subrutina.
        goto    ESPERE_TCLA ;volvemos arriba para esperar la siguiente tecla.


;OK_OK: Si NTECLAS=0, no pasa nada, sino, sale de la subrutina reportando una entrada
;de RPM exitosa.

OK_OK       clrw
```

```
        xorwf   NTECLAS,w   ;Nº de teclas es cero?
        BZ      ESPERE_TCLA ;salta a ESPERE_CENT si NO se han presionado
dígitos
EXITO   clrf    TEMP        ;Limpia TEMP, lo que indica fin de la operacion.
        retlw   0           ;Retorna luego de introducir la RPM deseada exitosa/te

;CANCEL: Si NTECLAS=0, retorna de la sub. reportando cancelación de usuario
(TEMP=255)
;si NTECLAS no es =0, borra el registro del digito anterior (backspace)
CANCEL  clrw
        xorwf   NTECLAS,w   ;Nº de teclas es cero?
        BZ      ERROR_USR   ;salta a si NO se han presionado dígitos,
CANCELACIÓN.
        decf    NTECLAS,f   ;Decrementa contador de teclas introducidas
        decf    FSR,f       ;Decrementa apuntador de direccionamiento indirecto
        clrf    INDF        ;Borra de mem RAM digito anteriormente introducido
        movlw   0xC0        ;Control para pasar a 2ª línea de la LCD, posición 40h
        addwf   NTECLAS,w   ;Le suma a la palabra el Nº de teclas ya
decrementado
        movwf   TEMP        ;Guarda esta dirección en TEMP
        call    LCD_CTRL    ;Retrocediendo un caracter en la LCD al enviar control
        movlw   " "         ;Carga Espacio en blanco
        call    LCD_DATO    ;Sobreescribiendo caracter anterior Esto incrementa la
        movf    TEMP,w      ;posicion del cursor, luego toca volverlo a mover
        call    LCD_CTRL    ;o lo que es lo mismo, volver a retroceder...
        goto    ESPERE_TCLA ;salta habiendo hecho "backspace" o retroceso en
LCD
ERROR_USR   movlw   .255    ;Se reporta error de usuario o cancelación de
entrada
        movwf   TEMP        ;de RPM al programa haciendo TEMP = 255d.
        retlw   0           ;Retorna al programa principal reportando el error.
```

LCD.INC

```
;=========================Subrutinas de manejo de
LCD==========================
; Subrutinas para manejo de LCD a 4 bits de conexión, donde:
; D<4:7> en la LCD <-> RA<0:3> en el PIC, respectivamente.
; ENABLE (E) en la LCD <-> RE<0> en el PIC
; READ/WRITE (RW) en la LCD <-> RE<1> en el PIC
; Registry Select (RS) en la LCD <-> RE<2> en el PIC
;
; Estas subrutinas retornan en el Banco 0
;
;-------------------------------------------------------------------------------
;REGISTROS DE TRABAJO : (definidos en el archivo principal)

;       TLCD:   Registro temporal
```

```
;       PDel0:  Registro para subrutinas de retardo/reg. temporal
;       PDel1:  Registro para subrutinas de retardo/reg. temporal


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

;               PALABRAS ESPECIALES EMPLEADAS EN EL CONTROL DE LA PANTALLA LCD

#define     ENABLE_LCD  bsf     PORTE,0       ;Habilita la LCD,     E  = 1
#define     DISABLE_LCD bcf     PORTE,0       ;Deshabilita la LCD,  E  = 0
#define     READ_LCD    bsf     PORTE,1       ;Leer de la LCD    ,  WR = 1
#define     WRITE_LCD   bcf     PORTE,1       ;Escribir en la LCD,  WR = 0
#define     REGSEL_DATO bsf     PORTE,2       ;para enviar Dato,    RS = 1
#define     REGSEL_CTRL bcf     PORTE,2       ;enviar byte control, RS = 0


;-------------------------------------------------------------------------------
;                                       ENVIAR BYTE DE CONTROL O CARACTER
;-------------------------------------------------------------------------------
; Primero debe enviarse los 4 bits MSb y Luego los 4 bits LSb. Se elige el re-
; gistro (Datos o control), se preparan los nibbles (medios bytes) y se envían.

LCD_CTRL    REGSEL_CTRL
            goto    $+2
LCD_DATO    REGSEL_DATO
            movwf   TLCD            ;Guarda Caracter/Control a enviar
            movlw   0xF0
            andwf   PORTA,f         ;Limpia Bits de Datos RA<0:3> o D<4:7>
            swapf   TLCD,W          ;W=TLCDL:TLCDH
            andlw   0x0F            ;Limpia TLCDL de w, dejando TLCDH
            ENABLE_LCD
            iorwf   PORTA,f         ;Saca *PARTE ALTA* a Desplegar/Enviar
            call    RETARDO         ;Espera 1 mSeg al enviar orden
            DISABLE_LCD
            call    RETARDO         ;Espera 1 mSeg al enviar órden
            movlw   0xF0
            andwf   PORTA,f         ;Limpia Bits de Datos RA<0:3> o D<4:7>
            movf    TLCD,w          ;Carga Caracter/Control a Enviar
            andlw   0x0F            ;Limpia TLCDH de w, dejando TLCDL
            ENABLE_LCD
            iorwf   PORTA,f         ;Saca *PARTE ALTA* a Desplegar/Enviar
            call    RETARDO         ;Espera 1 mSeg al enviar órden
            DISABLE_LCD
            call    RETARDO
            call    RETARDO         ;Espera 1 mSeg al enviar órden
            retlw   0
```

```
;-------------------------------------------------------------------------------
;                                   CONFIGURACION DE LA PANTALLA LCD
;-------------------------------------------------------------------------------
LCD_CFG     call    mSEGx15         ;15 mSeg de retardo antes de comenzar...
            movlw   0x02            ;inicia display a 4 bits
            call    LCD_CTRL
            movlw   0x28            ;display a 4 bits y 2 líneas
            call    LCD_CTRL
            call    RETARDO         ;Espera 1 mSeg al enviar órden
            movlw   0x0C            ;activa display
            call    LCD_CTRL
            movlw   0x06            ;mensaje fijo
            call    LCD_CTRL
CLEAN_LCD   movlw   0x01
            call    LCD_CTRL        ;Limpia la LCD
            return                  ;Lista la configuración; retorne.


;-------------------------------------------------------------------------------
;                                   RETARDO (ESPERA A QUE LA LCD ASIMILE
DATO/CONTROL)
;-------------------------------------------------------------------------------
;Retardo de aproximadamente 0.5mSeg, asegura que el LCD asmile el envío.
RETARDO     movlw   .248        ; 1 set numero de repeticion
            movwf   PDel0       ; 1 |
            clrwdt              ; 1 clear watchdog
            decfsz  PDel0, 1    ; 1 + (1) es el tiempo 0  ?
            goto    $-2         ; 2 no, loop
            goto    $+1         ; 2 ciclos delay
            clrwdt              ; 1 ciclo delay
            return              ; 2+2 Fin.

;-------------------------------------------------------------------------------
;                                   RETARDO DE 15 miliSEGUNDOS a 4MHZ
;-------------------------------------------------------------------------------
;Necesarios para la inicialización de la LCD y para limpiar (aclarar) la LCD
;luego de inicializada.

mSEGx15     movlw   .21         ;1 set numero de repeticion  (B)
            movwf   PDel0       ;1 |
            movlw   .142        ;1 set numero de repeticion  (A)
            movwf   PDel1       ;1 |
            clrwdt              ;1 clear watchdog
            clrwdt              ;1 ciclo delay
            decfsz  PDel1,f     ;1 + (1) es el tiempo 0  ? (A)
            goto    $-3         ;2 no, loop
```

```
        decfsz  PDel0,f     ;1 + (1) es el tiempo 0  ? (B)
        goto    $-7         ;2 no, loop
        clrwdt              ;1 ciclo delay
        return              ;2+2 Fin.
```

MATRIZ3X4.INC

```
;==============================Subrutina
TECLADO==============================
;Espera a que se presione alguna tecla de un teclado matricial 4x4 convencional
;Que en este caso operara con sólo 3 de sus cuatro filas.
;El teclado está conectado al puerto B, con las filas F0,F1 y F2 conectadas,
;respectivamente, a los pines RB1,RB2,RB3. Las columnas C0,C1,C2,C3, van
;conectadas a los pines RB4,RB5,RB6,RB7. El Bit RB0 no va conectado.
;Antes de llamarla por vez primera se debe limpiar el puerto del teclado (por
;defecto PORTB<7:0>)
; Una vez retorna al programa principal, el valor de la tecla presionada está
; entre 00h y 0Bh según la tecla presionada, y almacenado tanto en el registro
; w como en el registro TECLA

;REGISTROS DE TRABAJO : (definidos en el archivo principal)

;       ROTA:  Rota la fila a examinar
;       TMAT:  Fila y columna 'conectados' al oprimir tecla
;       TECLA: Última Tecla activada x el usuario,


TECLADO call    _FILAX          ;rotacion en entre los bits RB<1:3> (filas)
        movlw 0xF0          ;Carga Mascara en w, w=F0
        andwf PORTB,w           ;Hace que W<0..3> = 0 (enmascara los
bits de fila)
        xorlw  0x00         ;Compara con 00
        btfsc  STATUS,Z     ;Salta si el Flag de Cero = 0 (se presiono tecla)
        goto   TECLADO          ;Z=1, no presionó tecla. Vuelva a cambiar
de fila
        call   _REVISA          ;Z=0, se presionó tecla. Llama subrutina
REVISA.
        return          ;Retorna al Programa con la Tecla presionada

;================================SUBRUTINA
FILAX============================
;Rota la fila a examinar, entre los bits RB<1:3>
_FILAX      rlf             ROTA,f
        btfsc  ROTA,4           ;Salte si el bit ROTA<4> es 0 (Todo Bien)
        goto   _FILA0           ;Si ROTA<4>=1, va a FILA0 para reiniciar
rotación.
        movf   ROTA,w           ;Cargue ROTA en w
```

```
            movwf PORTB             ;Saque ROTA por el puerto B
            return

;_FILA0 se encarga de inicializar o reinicializar la rotación de filas.
_FILA0      movlw 0xF0
            andwf PORTB,f           ;Limpia Puerto B
            bsf   PORTB,1           ;Activa la fila 0 del Teclado (RB<1>)
            movlw b'00000010'  ;Carga valor correspondiente a la fila 0
            movwf ROTA         ;en el registro ROTA
            return

;================================SUBRUTINA
REVISA============================
;Averigua el valor de la tecla oprimida.
_REVISA     movf    PORTB,w    ;Carga PORTB en w
            movwf   TMAT       ;Lo guarda en TMAT
            call _COLUMNA ;Llama para averiguar la columna <0:3> activada
            btfsc   ROTA,1          ;Revisamos la fila 0? salta si NO. Si
es la
            goto    _SACA       ;fila 0, Nº de la columna = tecla; vaya a
SACA
            btfsc   ROTA,2          ;Revisamos la fila 1? salta si NO.
            goto    _F1
_F2         addlw   0x08        ;Suma al valor de la columna el valor
            goto    _SACA      ;inicial de la fila 2 y salta a SACA.
_F1         addlw   0x04        ;Suma al valor de la columna el valor
                ;inicial de la fila 1.
_SACA       movwf   TECLA           ;Almacena el vlr. de la tecla.
            return

;==========================SUB-SUBRUTINA
COLUMNA========================== :)
;Averigua la columna activada, entregando su Nº en el registro W
_COLUMNA  btfsc TMAT,4     ;Revisa si se activó la columna 0.
            retlw   0               ;Si sí, retorna con W = 0.
            btfsc   TMAT,5     ;Si no, Revisa si se activó la columna 1.
            retlw   1               ;Si sí, retorna con W = 1.
            btfsc   TMAT,6     ;Si no, Revisa si se activó la columna 1.
            retlw   2               ;Si sí, retorna con W = 2.
            retlw   3               ;Si no, retorna con W = 3,
                ;pues se activó la columna 3.


;=====RUTINA ANTIRREBOTE=========================
;Espera a que el usuario deje de presionar la tecla actual, evitando que el pro
;grama asuma una misma tecla como dos o más dígitos Consecutivos cuando no
son
```

```
ANTIRREBOTE ;call    mSEGx15     ;15mSeg de Retardo (OJO, SUBRUTINA DE "LCD.INC")
        movf    TMAT,w      ;Fila-columna inmediata/te anteriores en W
        xorwf   PORTB,w     ;Lo compara con el valor actual del puerto B y
        btfsc   STATUS,Z    ;Salta si PORTB ya es diferente de DATO_PUERTO
        goto    ANTIRREBOTE ;Sino, Vuelve a ANTIRREBOTE a ESPERAR.
        clrf    TMAT        ;Limpia registro temporal de teclado matricial
        retlw   0           ;Retorna si el usuario soltó la tecla.
```

MCDU Y RPM.INC

```
;-------------------------------------------------------------------------------
;                                RUTINA DE REVISIÓN DEL VALOR DE RPM
;-------------------------------------------------------------------------------
;**¿ESTÁ LA RPM INTRODUCIDA POR EL USUARIO DENTRO DEL RANGO DE OPERACIÓN DEL MOTOR?**
; El motor opera hasta unas 4100 RPM normalmente, para facilitar los cálculos lo
; dejaremos operar sólo hasta 4000 RPM.  Si la RPM introducida excede este valor, se
; sobreescribirá el valor de RPM introducido con el valor máximo permitido: 4000 RPM.
REVISAR     movlw   .4          ;
        subwf   MILES,w     ;Se desea más de 4000 RPM?
        BC      MAYOR_A_4000 ;Salte Si MSB de 4000 < MSB de la RPM deseada
        retlw   0           ;Sino, retorne.

MAYOR_A_4000                ;Como la RPM deseada > 4000, sobreescribe la
        movlw   .4          ;entrada de usuario con 4000 que es el valor de RPM
        movwf   MILES       ;máximo en el que se operará el motor.
        movlw   .0
        movwf   CENTENAS
        movlw   .0
        movwf   DECENAS
        movlw   .0
        movwf   UNIDADES
        retlw   0

;-------------------------------------------------------------------------------
;                                RUTINA DE DESPLIEGUE DE RPM DE USUARIO X LCD
;-------------------------------------------------------------------------------
;Despliega RPM de usuario o del motor almacenada en MILES, CENTENAS, DECENAS y UNIDADES
;si es la de usuario y en MILES_R, CENTENAS_R, DECENAS_R y UNIDADES_R si es la de motor
RPM_OUT     movf    TEMP,w      ;Carga posición del cursor en la LCD:
                                ;0xC0 para la de USUARIO, 0xCC para la del MOTOR
```

```
        call    LCD_CTRL    ;Pasa el cursor de la LCD a la posición correspondient.
        movf    TEMP,w      ;Carga posición del cursor en la LCD:
        xorlw   0xC0        ;Es TEMP = C0h? (se quiere sacar RPM de USUARIO)
        BNZ     OUT_MOTOR   ;NO, salte para sacar RPM del Usuario
        movlw   MILES;      ;Posición del primer caracter a sacar (USUARIO)
        movwf   FSR         ;se guarda en el apuntador
        goto    SACAR_RPM   ;Salte a sacar RPM
OUT_MOTOR   movlw   MILESM      ;Posición del primer caracter a sacar (MOTOR)
        movwf   FSR         ;se guarda en el apuntador

SACAR_RPM
   VARIABLE i
      i=0
      WHILE i<4             ;4 ciclos (ahorra líneas, NO instrucciones en el PIC)
        movf    INDF,w      ;Carga dato apuntado,
        addlw   0x30        ;Le suma 30h (BCD -> ASCII) y
        call    LCD_DATO    ;lo saca por la LCD (M,C,D, o U)
        incf    FSR,f       ;Incrementa el Apuntador
      i++
      ENDW
        clrf    FSR         ;Limpia puntero direccionamiento indirecto
        retlw   0

;-------------------------------------------------------------------------------
;                       RUTINA DE CONVERSION DE RPM (BINARIA) A CICLO DE TRABAJO PWM
;-------------------------------------------------------------------------------
;Convierte RPM a Duty cycle y lo ajusta al formato del PIC de la siguiente forma: Si
;la RPMmax = 4000 = 00001111 101000xxb, y disponemos de 10 bits para expresarla,
;rotaremos a la izquierda el par de registros 4 veces para dejar los 8 bits MSb
;en BINH y los 2 bits LSb en BINL (tal y como los 'reconoce' el PIC).  Los 2
;bits LSb de la RPM deseada se ignorarán (los marcados arriba como xx)
BIN_A_PWM
        movf    BINL,w
        movwf   0x36
        movf    BINH,w
        movwf   0x37
   variable i
   i=0
      WHILE i<4             ;Repetimos las 2 siguientes lineas 4 veces:
        rlf     BINL,f      ;Rotamos MSB a la izquierda (puede generar Acarreo)
        rlf     BINH,f      ;Rotamos MSB a la izq. (Introduce Acarreo si hubo)
      i++
      ENDW
```

```
        return


;-------------------------------------------------------------------------------
;    RUTINA DE CONFIGURACION DEL CICLO DE TRABAJO DE PWM
;-------------------------------------------------------------------------------
;Configura la PWM con el ciclo de trabajo en los registros BINL<0:1> (2 bits LSb) y
;el BINH (8 bits MSb) Copiándolos a CCP2CON<5,4> (2 bits LSb) y a CCPR2L (8 bits MSb),
; Respectivamente. La subrutina se llama y retorna en el banco 0.
NUEVO_CICLOT
        btfss   BINL,6      ;LSb, bit menos significativo del Duty Cycle.
        bcf     CCP2CON,4   ;como CICLOT_L<6> = 0, CCP2CON<4> = 0
        btfsc   BINL,6      ;
        bsf     CCP2CON,4   ;como CICLOT_L<6> = 1, CCP2CON<4> = 0
        btfss   BINL,7      ;2º bit menos significante del Duty Cycle.
        bcf     CCP2CON,5   ;como CICLOT_L<7> = 0, CCP2CON<5> = 0
        btfsc   BINL,7      ;
        bsf     CCP2CON,5   ;como CICLOT_L<7> = 1, CCP2CON<5> = 0
        movf    BINH,w      ;Carga 8 bits MSb del ciclo de trabajo
        movwf   CCPR2L      ;y los guarda en el registro correspondiente: CCPR2L
        retlw   0           ;Ajustado el ciclo de trabajo, retorna
```

MENSAJE 1.INC

```
;TABLA MENSAJE EN LCD: VELOCIDAD DE USUARIO - VELOCIDAD REAL

MENSAJE     addwf   PCL,f
        dt "V.USER -- V.REAL*"
;el asterisco(*) es el indicador de fin de mensaje
        retlw   0

; DIRECTIVA dt
; Genera una serie de instrucciones RETLW, cada caracter en la cadena es
; almacenado en su propia instrucción RETLW
```

;PROGRAMA PRINCIPAL
CONTROLMOTOR.ASM

```
;Usando un motor DC, un microcontrolador; L293, Encoder, diseñar un control por
;lazo abierto de velocidad.  El Setpoint debe ser por teclado y el despliegue por
;LCD.

   LIST       P=16F877
   __config    _XT_OSC & _WDT_OFF & _LVP_OFF       ;XTAL = 20 MHz; No hay
LVP ni WDT
   RADIX      HEX
   INCLUDE    "P16F877.INC"
```

```
   INCLUDE    "MACROS.INC"
   TITLE      "CONTROL DE UN MOTOR X PWM"

   errorlevel -302 ; Desactiva mensaje "Register in operand not in bank 0...."
   errorlevel -207 ; Desactiva advertencia "Found label after column 1."
   noexpand        ; Desactiva la expansión de macros en el archivo .LST

;-------------------------------------------------------------------------------
;                  Declaración de registros de Usuario y Constantes
;-------------------------------------------------------------------------------

   CBLOCK      0x20
      TEMP        ;20h Registro temporal
      NTECLAS     ;21h # de teclas digitadas x el USUARIO/reg. temporal
      MILES       ;MILES,
      CENTENAS    ;CENTENAS,
      DECENAS     ;DECENAS
      UNIDADES        ;Y UNIDADES de la velocidad en RPM deseada por el
USUARIO
      TLCD        ;\
      PDel0       ; >-Registros empleados en las subrutinas de la LCD: LCD.inc
      PDel1       ;/
      BINL        ;\_Registros en donde va la RPM en Binario del USUARIO
      BINH        ;/
      MILESM      ;_
      CENTENASM   ; \
      DECENASM    ;  Digitos BCD de la velocidad del MOTOR en RPM
      UNIDADESM   ;_/
      BINLM           ;\_ Registros en donde se almacena la RPM del MOTOR en
binario
      BINHM       ;/
   ENDC

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

RESET       ORG     0x00        ;Vector de RESET
            goto    INICIO      ;Salta al inicio del programa
            ORG     0x04        ;Vector de Interrupciones

;-------------------------------------------------------------------------------
;                     ATENCIÓN A LA INTERRUPCIÓN
;-------------------------------------------------------------------------------
;En este sistema ocurre dos tipos de interrupciones.  Una es por flanco de subida
;en;RC2/CCP1 (captura, cada 1.2"). La otra interrupción es provocada externa/te
;en ;RB0/INT, la cual le indica al PIC que se van a enviar los 4 dígitos de la
;velocidad deseada por el usuario, en su orden: MILES, CENTENAS, DECENAS y
;UNIDADES.
```

```
INTERRUMPE  bcf     INTCON,GIE  ;Deshabilita todas las int. hasta terminar de
                                ;antender.
        btfsc   INTCON,INTF ;La interrupción es externa? (Entrada de RPM
                            ;usuario)
        goto    RECIBIR_RPM     ;Salta a Recibir RPM de usuario.

CAPTURA     bcf     T1CON,TMR1ON;APAGA EL TIMER1
        clrf    TMR1H           ;Reinicia MSB de la cuenta
        clrf    TMR1L           ;Reinicia LSB de la cuenta
        bsf     T1CON,TMR1ON    ;Enciende el TIMER1, comienza una nueva
                                ;cuenta.
        movf    CCPR1L,w        ;Carga parte baja de Cuenta Capturada
        movwf   BINLM           ;La mueve a BINLM
        movf    CCPR1H,w        ;Carga parte alta de Cuenta Capturada
        movwf   BINHM           ;La mueve a BINHM
        call    BIN_A_BCD       ;Convierte BINHM:BINLM en BCD: MILESM ...
                                ;UNIDADESM.
        movlw   0xCC
        movwf   TEMP        ;Carga posición del cursor de la LCD para que
        call    RPM_OUT     ;Saque el valor de RPM DEL MOTOR.
        bcf     PIR1,CCP1IF ;Borra el flag de Interrupción del Módulo captura CCP1
        bsf     INTCON,GIE  ;Rehabilita todas las int. hasta terminar de antender.
        retfie          ;Retorna al salto anidado

RECIBIR_RPM movlw   MILES   ;Carga la dirección 1er registro a recibir: MILES.
        movwf   FSR         ;Lo guarda en el puntero del direccionamiento indirecto
        clrf    NTECLAS     ;Reinicia contador de teclas recibidas.
        call    RETARDO     ;Llama retardo de 1mSeg (Está en el archivo LCD.inc)
                            ;Para asegurar el dato estabilizado en el Puerto

LEER        bcf     STATUS,C    ;Borra Carrier para preparar rotación
        rrf     PORTB,w         ;Elimina el bit RB0 de interrupción y deja los 4 bits
                                ;del dígito BCD en W.
        movwf   INDF            ;Guarda dígito en la dirección apuntada por FSR
        incf    NTECLAS,f       ;Incrementamos contador de teclas recibidas
        incf    FSR,f           ;Incrementa el puntero de direccionamiento.
        call    RETARDO         ;retardo de 1mSeg (Está en el archivo LCD.inc)
        call    RETARDO     ;retardo de 1mSeg (2mSEG de retardo entre dígitos)
        btfss   NTECLAS,2   ;Es el bit 2 NTECLAS NTECLAS<2>= 1 (NTECLAS =
                            ;4)?
        goto    LEER        ;No, vuelva a leer para recibir el siguiente dígito
        clrf    FSR         ;Limpia puntero de direccionamiento indirecto
ACEPTAR     call    REVISAR     ;Revisa rango de RPM, si es >, sobreescribe con
                                ;4000.
        movlw   0xC0
        movwf   TEMP        ;Ajusta Cursor al comienzo de la 2a línea
```

```
        call    RPM_OUT     ;Saca la nueva RPM de usuario por la LCD
        call    BCD_A_BIN   ;subrutina de conversión de BCD a binario la cual
                            ;entregará en BINH:BINL el valor de RPM binario.
        call      BIN_A_PWM     ;Halla el Duty Cycle partiendo de la RPM en
                                ;BINH:BINL
        call    NUEVO_CICLOT;Configure el Duty Cycle en el módulo CCP2(PWM)
        bsf     INTCON,GIE  ;Rehabilita todas las int. hasta terminar de antender.
        bcf     INTCON,INTF ;Luego de antender la interrupción, apagamos el flag.
        retfie              ;Retorno desde subrutina de interrupción

    Include "LCD.INC"           ;Rutinas de manejo de la LCD:
                                ;LCD_CONTROL: Enviar dato de control
                                ;LCD_DATO   : Enviar caracter a desplegar
                                ;LCD_CFG    : Configurar-Reiniciar la LCD
    Include "Mensaje 1.inc"   ;Contiene tabla MENSAJE con el mensaje:
                                ;  V.USER -- V.REAL
                                ;a mostrarse en la primera línea de la LCD
    Include "BCD - binario.inc" ;Rutina de conversión BCD <-> binario:
                        ;BCD_A_BIN  : recibe UNIDADES, DECENAS, CENT.
                        ; Y MILES, y entrega la cantidad
                        ; binaria en los reg's BINH:BINL
                        ;BIN_A_BCD  : recibe los reg's BINH:BINL y entrega
                        ; su equivalente BCD en UNIDADES_R,
                        ; DECENAS_R, CENTENAS_R, MILES_R.
    Include "MCDU y RPM.inc"    ;Rutinas de manejo de los registros de
                        ;MILES, CENTENTAS, DECENAS y UNIDADES (MCDU) y
                        ;RPM en Binario (BINH:BINL)
                        ;REVISAR    : Revisa si la RPM solicitada < 4000
                        ; (límite de RPM del motor), si es >,
                        ; la sobreescribe con 4000.
                        ;RPM_OUT    : Despliega la RPM de Usuario o motor
                        ;BIN_A_PWM  : Pasa de RPM a ciclo de trabajo
                        ;NUEVO_CICLOT:Configura Nuevo Ciclo de trabajo PWM

;=============================================================;
P R O G R A M A   P R I N C I P A L
;=============================================================
;       PORTA: LCD. PORTB: TECLADO. PORTE: LCD (CONTROLES)    _
INICIO    bsf     STATUS,RP0  ;Banco 1                        |
        movlw   0x06        ;                               | P
        movwf   ADCON1      ;PORTA<x>: E/S Digitales        | U
        clrf    TRISA       ;RA<x>: Salidas.                | E
        movlw   b'00011111' ;                               | R
        movwf   TRISB       ;RB<0:4> Entradas, RB<5:7> Salidas  | T
        movlw   b'11111111' ;                               | O
        movwf   TRISC       ;RC<x>:Entradas. (TEMPORALMENTE)    | S
        clrf    TRISE       ;RE<2:0>: Salidas.              _|
```

```
;                                                    _
        movlw   b'00000100' ;configura interrupción únicamente  | I C
        movwf   PIE1        ;por el módulo CCP1. Ninguno de los | N I
        clrf    PIE2        ;otros módulos pueden interrumpir.  | T O
        movlw   b'11000000' ;                                   | E N
        clrf    INTCON      ;Desactiva interrupciones generales | R E
        bsf     INTCON,INTE ;Interrupción externa en RB0/INT    | R S
        bsf     INTCON,PEIE ;Habilita interrupciones x módulos  | U
        bsf     OPTION_REG,INTEDG ;RB0 = flanco de subida.      | P
        BANKSEL T1CON       ;BANCO 0                        _| -

;-------------------------------------------------------------------------------
;                   INICIALIZACION LCD y TECLADO. Mensaje la línea1 de la LCD
;-------------------------------------------------------------------------------
        call    LCD_CFG     ;Configura la LCD
        clrf    TEMP            ;TEMP=contador caracteres mensaje para LCD en
TABLA.
        movf    TEMP,w      ;Carga contador de caracteres desplegados. A partir de
        call    MENSAJE     ;ese dato, la subrutina devuelve el siguiente caracter
        movwf   NTECLAS     ;Temporalmente, almacena caracter del mensaje
        xorlw   "*"         ; * es el caracter que determina final de mensaje.
        btfsc   STATUS,Z    ;Compara el caracter recibido con *
        goto    $+5         ;Si ya llego al caracter *, terminó el mensaje. Sale
        movf    NTECLAS,w   ;Restaura caracter del mensaje
        call    LCD_DATO    ;saca caracter por la LCD
        incf    TEMP,f      ;Incrementa contador de caracteres en el mensaje
        goto    $-9         ;Vuelve a subir a por el siguiente caracter

;-------------------------------------------------------------------------------
;    INICIALIZACIÓN DEL MÓDULO CCP2 COMO PWM - CONFIGURACION DEL
;    TIMER 2
;-------------------------------------------------------------------------------
;Para tener una frecuencia de pwm Fpwm=1kHz a una fXTAL de 4MHz, PIR2 =
0xF9 y la
;Preescala del Timer2 PSTMR2 = 4. A 1kHz de Fpwm se dispone de los 10 bits de
;resolución para especificar el Duty Cycle (D.C.).  El sistema se inicializará
;A LA MITAD DE SU VELOCIDAD MÁXIMA (50% Ciclo de trabajo, unas 2000
RPM).
;Un D.C. = 100% = 4000 RPM aproximadamente.

        movlw   0xF9        ;1. Ajustar el período de la PWM escribiendo el valor
        BANKSEL PR2     ;BK1      adecuado en el registro PR2. (0xF9)
        movwf   PR2
        BANKSEL PORTA ;BK0  ;2. Ajusta el ciclo de trabajo (Duty Cycle) de la
                                ;PWM
        movlw   b'11010000' ;al valor incial = 50% (2000RPM), Escribiendo el
```

```
        movwf   BINL            ;registro CCPR1L y los bits CCP1CON<5:4> (LSb)
                                ;mediante
        movlw   b'00000111' ;las rutinas BIN_A_PWM -que pasa RPM a binario- y
        movwf   BINH        ;'NUEVO_CICLOT'-que ajusta a los registros BINH:BINL
        call    BIN_A_PWM   ;al formato del PIC : BINH<0:7>:BINL<7,6> = BINH(8
                            ;MSb)
        call    NUEVO_CICLOT;BINL<7,6> (2 LSb) y los configura en el PIC-.
        BANKSEL TRISC ;BK1   3. Hacer del pin RC1/CCP2 una salida
        bcf     TRISC,1     ;bit TRISC<1>.
        BANKSEL T2CON ;BK0   4. Ajustar la preescala del TMR2=4 y habilitarlo
        movlw   b'00000101' ;modificando el registro T2CON.
        movwf   T2CON       ;
        movlw    b'00001111' ;5. Configurar el módulo CCP2 para operación en
                            ;PWM.
        movwf   CCP2CON     ;
        movlw   .2          ;
        movwf   MILES       ;Valores Iniciales de los registros de RPM deseada
        clrf    CENTENAS    ;
        clrf    DECENAS     ;           RPM inicial =  2 0 0 0
        clrf    UNIDADES    ;
        movlw   0xC0        ;
        movwf   TEMP        ;Configura posición en la LCD para
        call    RPM_OUT     ;Desplegar en la LCD este valor de RPM

;-------------------------------------------------------------------------------
;  CONFIGURACION DEL TIMER1 y EL MÓDULO CCP1 COMO CAPTURE
;-------------------------------------------------------------------------------
;El TIMER 1 contará las muestras generadas por un optoacoplador ranurado que
;monitorea una rueda de 50 muescas acoplada al eje del motor.  El módulo CCP1
;debe capturar esta ;cuenta ante un flanco de subida en el pin RC2/CCP1, que se
;enviará cada 1.2" desde un ;un reloj externo de f = 1/1.2 = 0.8333 Hz conectado a
;este pin.

        movlw   b'00000010' ;Control del TIMER1: Sin preescala, sincroniza señal
        movwf   T1CON       ;externa, apagado.
        clrf    CCP1CON     ;Se aclara el registro antes de escribirlo, para no
        movlw   b'00000101' ;generar alguna interrupción falsa (más información
        movwf     CCP1CON;en la datasheet del PIC). Configura CCP1 como
CAPTURE
        clrf    TMR1H           ;Reinicia MSB de la cuenta
        clrf    TMR1L           ;Reinicia LSB de la cuenta
        bsf       T1CON,TMR1ON    ;ENCIENDE EL TIMER1, inicia la cuenta de
                                  ;pulsos.

        bsf     INTCON,GIE  ;Habilita las INTERRUPCIONES

USUARIO     goto    USUARIO     ;Salto anidado.  Espera a que ocurra:
```

```
                    ;Que en el pin RB0/INT haya un alto, indicando al PIC
                    ;que durante los 4 mSeg siguientes se va a mandar la
                    ;RPM de usuario (RPM deseada) o que haya interrupcion
                    ;por flanco de subida en CCP1/RC2 (captura).
End
```

13. CONFIGURACIÓN DE MPLAB PARA USAR CCS (Lenguaje C)

Pasos a seguir:

13.1. Preparación de la instalación.

- Asegurarse que CCS esté instalado en el PC y que funciona correctamente.
- Comprobar que MPLAB 6.xx/7.xx esté instalado.
- Instalar el plugin para que MPLAB pueda usar CCS. Si no lo has descargado, puedes hacerlo desde la web de CCS: CCS MPLAB 6.xx/7.xx plugin

13.2. Creación del proyecto.

- Añadir los archivos fuente .c al proyecto.
- En la pantalla donde el asistente nos pregunta que "Toolsuite" deseamos, seleccionar ***"CCS C Compiler for PIC12/14/16/18"***. Si no puedes ver esta opción es que el plugin CCS MPLAB 6.xx/7.xx plugin no se instaló correctamente.
- No añadir más de un fichero .C en el directorio "Source Files" de la ventana de proyecto. Si se añade más de uno, cuando se compile el proyecto, MPLAB compilará cada uno individualmente y luego intentará enlazarlos. Como CCS no incluye un enlazador ("linker") siempre nos dará un error.

13.3. Compilación.

- Compilar el proyecto, seleccionando el menú "Project → Compile" (vemos que el plugin de CCS ha cambiado la etiqueta "Build" por "Compile"
- Después de compilar, MPLAB es capaz de reconocer las dependencias entre los ficheros, es decir que #include estamos utilizando. Automáticamente añadirá a nuestra ventana de proyecto los archivos utilizados en el #include. Si bien nosotros podemos hacer esto manualmente, como vemos no es necesario. Es conveniente mencionarlo para que nadie se extrañe de donde han salido esos nuevos archivos en el proyecto. Por ejemplo, #include "16F877A.h" este archivo se añade en el directorio de "Header files"

13.4. Vistas

Por último en el apartado de los "watch", mientras se depura se pueden añadir las variables en C y ver sus valores. En este caso, se selecciona de la lista al lado del botón "Add Symbol" la variable (símbolo) deseada y se añade con dicho botón.

El próximo curso sobre Microcontroladores será presentado con el Micro 16F887 con ejemplos en Lenguaje C