EVOLUPIC
Bootloader 16F88
Manual del Usuario

Indice :

1. Introducción :

La filosofía de diseño de este módulo se centró en maximizar el número de funciones disponibles al usuario a un costo mínimo. **EVOLUPIC *Bootloader* 16F88** conserva la compatibilidad con 16F628 y el 16F84, realizando solo cambios mínimos en los programas.

Gracias a la posibilidad del microcontrolador 16F88 de autoprogramar su memoria FLASH, el **EVOLUPIC *Bootloader* 16F88** no necesita del hardware de programación ICSP, usado por su antecesor el EVOLUPIC 16F628 lo cual simplifica considerablemente su electrónica y permite, a través del firmware ***Bootloader*** (precargado de fábrica), programar el 16F88, vía un cable serial DB9-DB9, ó cualquier cable de interfaz USB-DB9, mediante un software que se ejecuta en la PC.

Le recomendamos ir directamente **al capítulo 17, página 22** : "**Puesta en Marcha**", para tener el sistema operando rápidamente. Este manual se acompaña de la especificación técnica y diagramas electrónicos completos del módulo, así como de diagramas y explicación detallada de cada uno de sus interfaces.

Por favor llame al tel. 56 53 58 01 para aclarar cualquier duda sobre **EVOLUPIC *Bootloader* 16F88**. Puede también enviar sus emails a : **atencionaclientes@puntoflotante.net**

Juan Martínez, Punto Flotante, S.A., enero de 2011

2. Descripción General.

El sistema **EVOLUPIC *Bootloader*** fue diseñado para su uso en una amplia gama de aplicaciones : como kit para el aprendizaje del microcontrolador 16F88, como módulo central en equipos de instrumentación ; en aplicaciones de robótica, comunicaciones y redes de control de acceso, o incorporado en sistemas de control industrial ó sistemas de seguridad. El diagrama general de la tarjeta EVOLUPIC ***Bootloader*** se muestra en la siguiente figura:

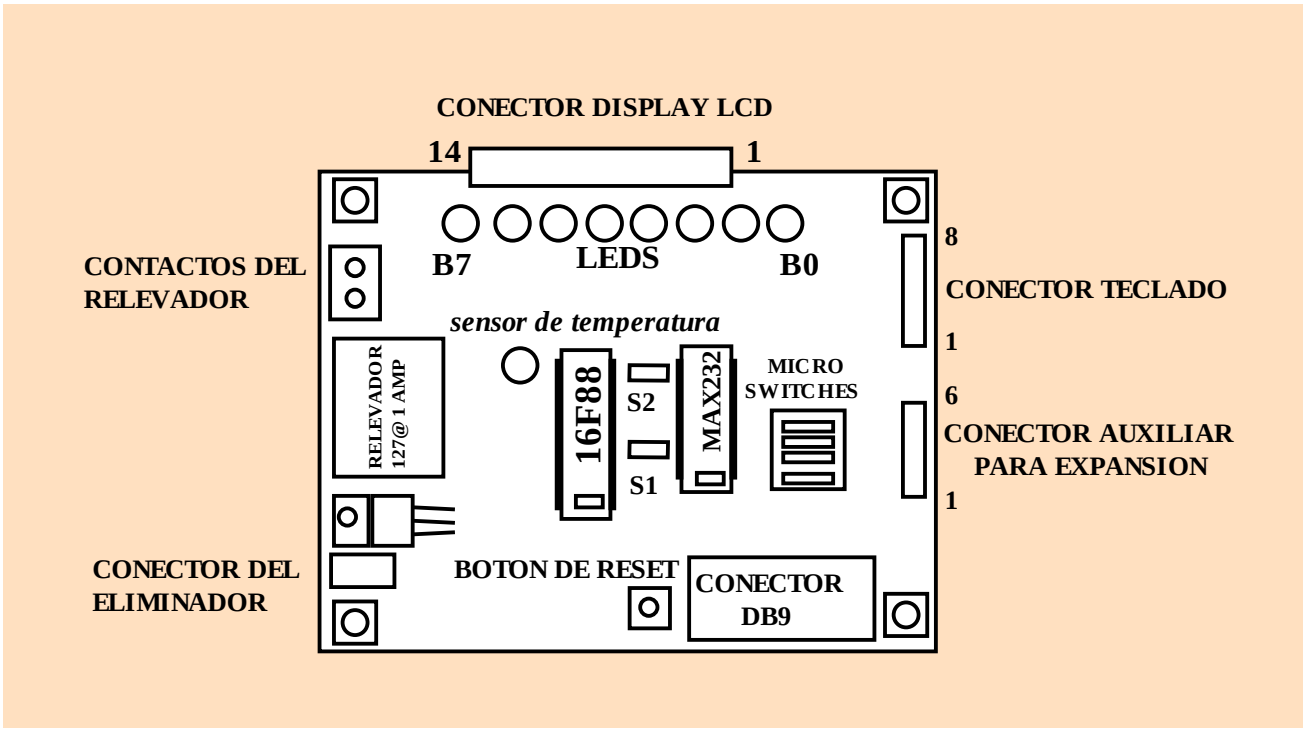


FIGURA 1

Se hace uso de la versión de 18 pines del microcontrolador 16F88, con encapsulado "dual in line", el cual contiene, 4096 bytes de memoria FLASH, 368 bytes de RAM y 256 bytes de EEPROM, y. El diseño hace un uso extensivo de los puertos e interfaces ofrecidas por el microcontrolador para ofrecer al usuario un sistema con un máximo de opciones disponibles.

- Microcontrolador **16F88,** funcionando con un cristal externo de **4 Mhz.**
- ***Programador Bootloader*** *de la memoria FLASH del 16F88* por vía puerto serial RS232.

- **4K bytes de memoria FLASH, 256 bytes de EEPROM, 368 bytes de RAM.**
- Tecnología CMOS con **muy bajo consumo,** en funcionamiento normal, <1 miliampere @ 5 volts.
- **Arquitectura Harvard**, con un set **RISC**, de solamente **35 instrucciones.**
- Puerto de salida de **8 bits con leds** conectados como testigos para facilitar pruebas por parte del usuario..
- Entradas para **4 señales digitales con microswitches** para pruebas y emulación de alarmas.
- Un total de **16 bits programables** como entradas o salidas digitales.
- **Relevador** de 127VAC@ 1A, integrado a la tarjeta, para la activación de dispositivos externos.
- **Sensor digital de temperatura DS18B20**, integrado al módulo.
- **3 Temporizadores de 8/16 bits** para la generación de retrasos, reloj de tiempo real ó contadores de eventos.
- **Puerto serial USART** para comunicación asíncrona estándar **RS232,** con salida de conector DB9.
- **SSP (Syncronous Serial Port) Puerto serial síncrono,** con 2 modos de operación: **SPI** (Serial Peripheral Interface, modos Master/Slave) e **I2C** (Integrated, Integrated Circuit. Modo Slave)
- **1 Voltaje de referencia analógico de salida**
- **2 comparadores analógicos.**
- **1 módulo de captura/comparación digital.**
- **1** salida especial para generar **PWM** (pulse wide modulation), con 10 bits de resolución.
- **7 canales de conversión ADC** con 10 bits de resolución.
- Conector de 14 pines para conexión **a display LCD** de 16 x 1 ó 16 x 2.
- Conector de 8 pines para **teclado matricial** de 16 teclas.
- Conector **Header auxiliar de 6x** para conexión a interfaces o aplicaciones externas.
- Circuito vigilante **Watch Dog** programable para evitar que el microcontrolador se salga de operación.
- Circuito de protección **Brown Out Reset,** el cual genera un reset automático al detectar picos en el voltaje de 5 v.
- Modo de operación de bajo consumo **SLEEP**, con un consumo virtual de 0 (<1 ua).
- Opción de protección de código **CODE PROTECTION** para evitar posible copia del firmware del circuito.
- Sistema de **interrupciones**, generadas desde varios dispositivos, entre ellos, las señales en los puertos, el temporizador y el USART, la escritura en la EEPROM.
- En cuanto a su alimentación, EVOLUPIC puede activarse mediante un eliminador de baterías externo, el cual alimenta a un regulador de 5 volts integrado a la tarjeta, o bien puede funcionar en forma autónoma por medio de una batería estándar “cuadrada” de 9 volts.

SOFTWARE :

- Se proporciona junto con el sistema, un disco CD. Se incluye en dicho disco: el programa **MPLAB IDE**(ejecutable desde una PC con cualquier plataforma WINDOWS), el cual incluye funciones de editor, macroensamblador, simulador y compilador de C.
- También se incluye el software Free Open Source llamado “TINY”, para la programación de la memoria FLASH del 16F88 a través de su puerto serial RS232 y un firmware precargado llamado Bootloader TINY.
- Decenas de programas ejemplo para prueba u desarrollo de los periféricos y cada una de las funciones del sistema.

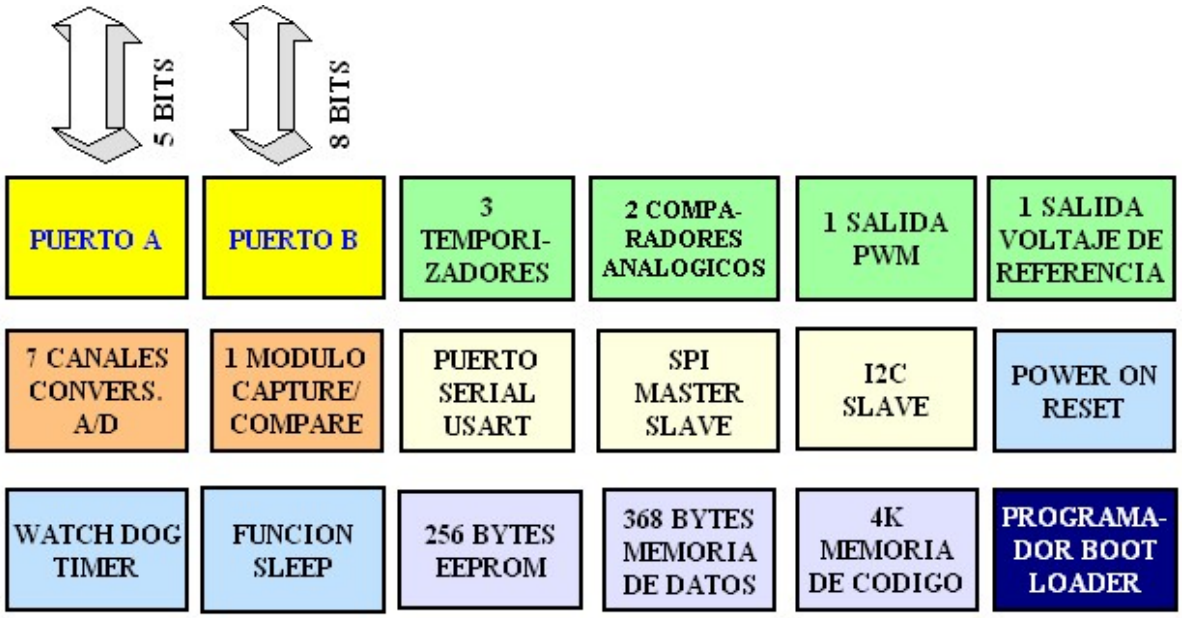


DIAGRAMA DE BLOQUES DEL MICROCONTROLADOR 16F88

3. Definición de los pines en el 16F88.

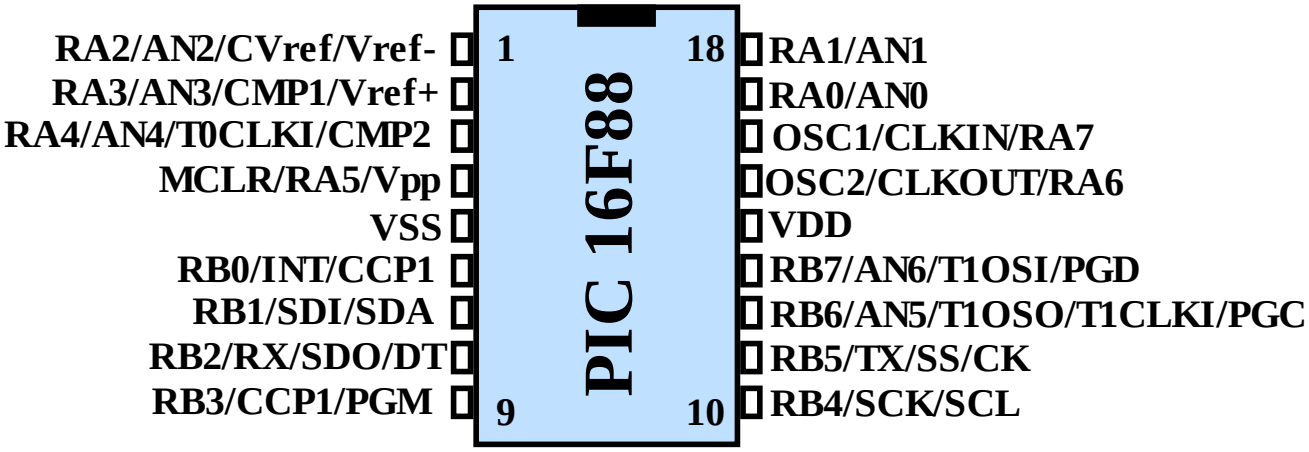


FIGURA 3

Casi todos los pines tienen varias funciones, en la tabla que sigue se describe cada una de ellas.

Pin	Nombre	Tipo	Funciones
1	RA2/ AN2/ CVref / Vref-		RA2 ENTRADA/SALIDA PUERTO A AN2 ENTRADA ANALOGICA 2 Vref SALIDA VOLTAJE DE REFERENCIA PARA COMPARADOR Vref- VOLTAJE BAJO DE REFERENCIA PARA CONVERTIDOR A/D.
2	RA3/ AN3/ Vref+/ C1out		RA3 ENTRADA/SALIDA PUERTO A AN3 ENTRADA ANALOGICA 3 Vref+ VOLTAJE ALTO DE REFERENCIA PARA EL CONVERTIDOR A/D C1out SALIDA DEL COMPARADOR ANALÓGICO 1
3	RA4/ AN4/ T0CKI/ C2out		RA4 ENTRADA/SALIDA PUERTO A. SALIDA ES OPEN DRAIN. AN4 ENTRADA ANALÓGICA 4 T0CKI ENTRADA DE RELOJ PARA TIMER 0 C2out SALIDA DEL COMPARADOR ANALÓGICO 2

4	MCLR/ RA5/ Vpp		MCLR RESET GENERAL AL CONTROLADOR RA5 ENTRADA PUERTO A Vpp VOLTAJE DE PROGRAMACION (ver estándar ISCP)
5	VSS		TIERRA DEL 16F88
6	RB0/ INT / CCP1		RB0 ENTRADA/SALIDA PUERTO B INT INTERRUPCION EXTERNA CCP1 ENTRADA MODULO CAPTURA, SALIDA MODULO DE CAPTURA, SALIDA DE PWM
7	RB1/ SDI / SDA		RB1 ENTRADA/SALIDA PUERTO B SDI DATOS DE ENTRADA DEL SPI SDA DATOS DEL TRASMISOR I2C
8	RB2/ SDO/ RX /DT		RB2 ENTRADA/SALIDA PUERTO B SD0 DATOS DE SALIDA DEL SPI RX DATOS DE RECEPCION ASINCRONA DEL USART DT DETECCIÓN SINCRONÍA DEL USART
9	RB3/ PGM/ CCP1		RB3 ENTRADA/SALIDA PUERTO B PGM VOLTAJE DE PROGRAMACIÓN BAJO DEL ICSP CCP1 ENTRADA MODULO CAPTURA, SALIDA MODULO DE CAPTURA, SALIDA DE PWM
10	RB4/ SCK/ SCL		RB4 ENTRADA/SALIDA PUERTO B SCK ENTRADA/ SALIDA SINCRONA DE RELOJ PARA SPI SCL ENTRADA SINCRONA DE RELOJ PARA I2C
11	RB5/ SS/ TX/ CK		RB5 ENTRADA/SALIDA PUERTO B SS SELECTOR DE MODO ESCLAVO PARA SPI TX DATOS DE TRASMISIÓN ASINCRONA DEL USART CK RELOJ SINCRONO DEL SPI
12	RB6/ AN5/ PGC/ T1OSO/ T1CKI		RB6 ENTRADA/SALIDA PUERTO B AN5 ENTRADA ANALÓGICA 5 PGC ENTRADA DE PROGRAMACION CLOCK ISCP T1OSO SALIDA DE OSCILADOR DEL TIMER 1 T1CKI ENTRADA DE RELOJ EXTERNO DEL TIMER 1
13	RB7/ T1OSI/ PGD/ AN6		RB7 ENTRADA/SALIDA PUERTO B T1OSCI ENTRADA OSCILADOR TIMER 1 PGD ENTRADA DE DATOS DE PROGRAMACION ISCP AN6 ENTRADA ANALÓGICA 6
14	VDD		VOLTAJE 5 VOLTS DEL 16F88
15	OSC2/ CLKO/ RA6		OSC2 ENTRADA OSCILADOR CRISTAL 4 MHZ CLKO SI HAY OSCILADOR RC EXTERNO, SALIDA ¼ DE FRECUENCIA RA6 ENTRADA/SALIDA BIDIRECCIONAL
16	OSC1/ CLKI/ RA7		OSC2 ENTRADA OSCILADOR EXTERNO CRISTAL 4 MHZ CLKIN ENTRADA OSCILADOR EXTERNO RA7 ENTRADA/SALIDA PUERTO A
17	RA0/ AN0		RA0 ENTRADA/SALIDA PUERTO A AN0 ENTRADA ANALOGICA 0
18	RA1/ AN1		RA1 ENTRADA/SALIDA PUERTO A AN1 ENTRADA ANALOGICA 1

4. Programador ***Bootloader*** de la memoria FLASH:

La tarjeta **EVOLUPIC *Bootloader*** cuenta con un programador para su memoria FLASH, basado en la capacidad de autoprogramación del 16F88. El denominado “Bootloader TINY” es un firmware precargado de fábrica en la parte alta de la memoria de código del 16F88. A través del ***Bootloader***, y de un software llamado “tinybldWin.exe”, se programa en forma rápida y sencilla la memoria FLASH, sin necesidad de usar un programador ICSP, ni de circuitería electrónica adicional.

La programación se realiza desde la computadora PC, a través de un cable serial DB9-DB9 ó bien por medio de un cable de interfaz USB-DB9, cuyo driver de instalación crea un puerto virtual COM.

5. El ciclo de máquina del 16F88

El 16F88 puede funcionar con un oscilador interno ó también puede usarse un cristal externo para aplicaciones que requieran una base de tiempo precisa. En el caso de la tarjeta **EVOLUPIC**, un cristal externo de 4 MHz se encuentra ya instalado. El oscilador principal es dividido entre 4 para formar los pulsos Q1, Q2, Q3, y Q4, estos 4 pulsos hacen un ciclo de máquina. En el siguiente diagrama se muestra el diagrama básico de operación del reloj, en donde se divide el oscilador principal en 4 ciclos, para cada ciclo de máquina.

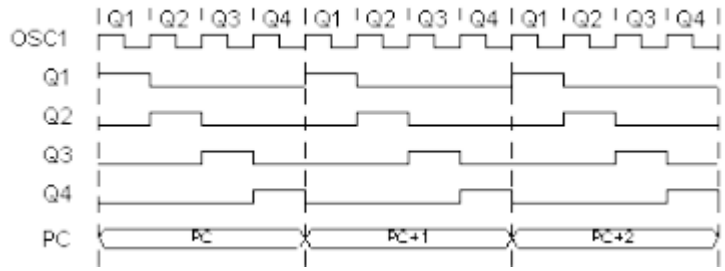


FIGURA 4

6. Arquitectura del microcontrolador PIC 16F88

El PIC16F88 pertenece al tipo de procesadores con arquitectura Harvard, es decir, la memoria de datos y de código están separadas, El microcontrolador cuenta con los siguientes elementos: memoria de programa de 4K del tipo FLASH, programable y borrable eléctricamente, 256 bytes de memoria EEPROM para el almacenamiento de parámetros, direcciones ó claves, según la aplicación, 368 bytes de RAM., dos puertos de entrada-salida , el puerto A con 8 señales y el B con 8 señales, en total 16 señales de entrada salida. Adicionalmente, el microcontrolador cuenta con 3 temporizadores.

Gracias a un set de instrucciones RISC (Reduced Instruction Set Computer), el CPU procesa solo 35 instrucciones. Todas las instrucciones tienen una longitud de palabra de 14 bits y se ejecutan en un ciclo de instrucción, con excepción de las instrucciones que modifican el contenido del contador del programa: JUMP, BRANCH, CALL, RETURN, RETFIE, RETLW. Lo anterior es debido al esquema de “pipeline” usado en arquitecturas HARVARD y que permiten al procesador realizar el FETCH y el EXECUTE simultáneamente con excepción de las instrucciones de salto referidas. En el siguiente diagrama se muestra la ejecución del programa de ejemplo con un sistema tipo “pipeline” . Obsérvese que en todos los ciclos de reloj, se hace el fecth y execute simultáneamente, con excepción del ciclo TCY4, en donde se deshecha (flush) la instrucción número 4 y se continúa con la instrucción 5, llamada por la subrutina..

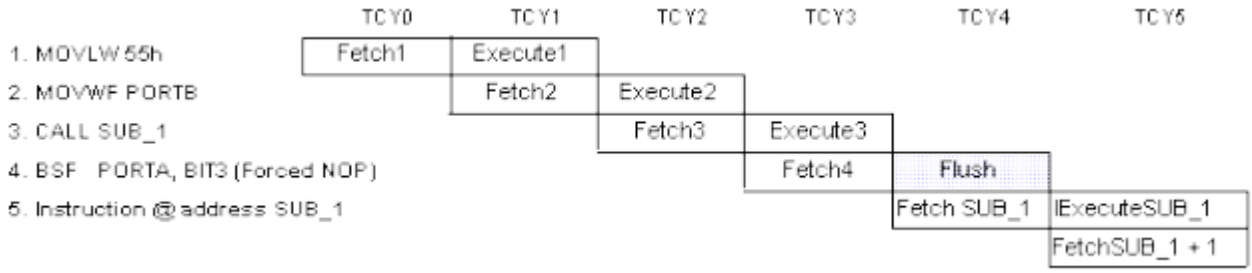


FIGURA 5

El microcontrolador 16F88 contiene los siguientes registros principales: el **registro W**, de 8 bits, que es el único acumulador del procesador, y el **registro PC** (program counter) que es un apuntador de 12 bits y que direcciona a la siguiente localidad de memoria de código que habrá de leerse y ejecutarse. La **PILA ó STACK**, que se usa para el manejo de las instrucciones de GOTO, CALL, RETURN, RETFIE, RETLW. Es una pila de 8 niveles que se encuentra en una memoria independiente de la memoria de programa y código, y allí se almacenan y recuperan las direcciones de retorno después de los llamados a subrutina. Es importante señalar que, dado que se trata de una pila de solo 8 localidades, solo pueden anidarse hasta 8 llamados a subrutinas o interrupciones dentro del programa.

7. Memoria

La arquitectura HARVARD permite el uso de dos buses de datos separados para la memoria de **programa y de datos**. La memoria de programa es del tipo FLASH, con capacidad de programar y borrar hasta 10,000 veces, y cuenta con 4096 localidades de 14 bits, con direcciones de la 000H a la FFFH.

Las direcciones 0 y la 4 de esta memoria son usadas para los vectores de RESET e interrupción respectivamente, es decir, después de RESET, al contador del programa apunta hacia la dirección 0 y después de la ocurrencia de una interrupción (que se encuentre habilitada), el contador del programa apuntará a la dirección 4.

La memoria de datos es de localidades de 8 bits, e incluye 368 localidades de RAM, y 256 localidades de EEPROM (con capacidad de programar y borrar hasta 10 millones de veces). La memoria de datos RAM **está organizada en 4 bancos** y se encuentra debajo de los registros especiales (SFR) del 16F88. Las direcciones disponibles de RAM son: BANCO 0: 20H A 7FH (96 bytes), BANCO 1: A0H a EFH (80 bytes), BANCO 2: 110H a 16FH (96 bytes) y BANCO 3: 190 a 1EF (96 bytes). Las localidades pueden ser accesadas con direccionamiento directo o indirecto.

En el siguiente diagrama, se muestran los dos tipos de arquitectura usados en computadoras: la arquitectura Harvard y la Von Neumann.

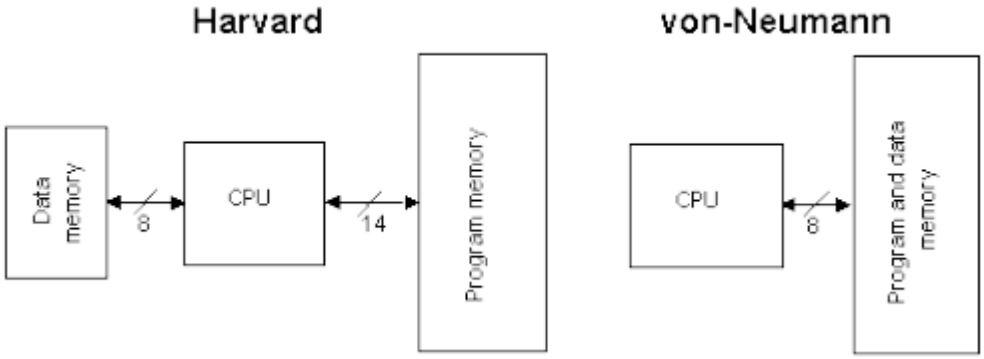


FIGURA 6

8. Registros especiales. SFR

Los denominados SFR (Special Function Registers), permiten al programador seleccionar las distintas opciones de las funciones del microcontrolador. En seguida se detalla la función de cada registro de estos 4 bancos de memoria. El banco se selecciona mediante los bits RP0 y RP1 del registro de STATUS. Algunos de los registros se encuentran repetidos en los bancos.

Registros en el 16F88, similares al 16F84:

INDF	REGISTRO USADO, JUNTO CON EL APUNTADOR FSR, PARA DIRECCIONAMIENTO INDIRECTO .
TMR0	REGISTRO QUE CONTIENE EL VALOR DEL CONTADOR/ TEMPORIZADOR (8 BITS)
OPTION REG	REGISTRO QUE PERMITE EL CONTROL DEL CONTADOR/ TEMPORIZADOR 0, DE LA INTERRUPCION EXTERNA Y DE LAS RESISTENCIAS DE PULL UP DEL PUERTO B.
PCL	PARTE BAJA DEL CONTADOR DEL PROGRAMA (8 BITS).
STATUS	GUARDA EL ESTADO DE LAS BANDERAS C (CARRY), DC (HALF CARRY), Z (ZERO), PD (POWER DOWN, TO (TEMPORIZADOR), RP0 (SELECTOR DE BANCO), RP1 (SELECTOR DE BANCO).
FSR	REGISTRO APUNTADOR USADO PARA EL DIRECCIONAMIENTO INDIRECTO DE LA MEMORIA DE DATOS. SE USA JUNTO CON INDF PARA LEER O ESCRIBIR SOBRE UNA LOCALIDAD DE MEMORIA.
PORTA	PUERTO A
TRISA	REGISTRO DE SELECCIÓN DE BITS DE ENTRADA O SALIDA DEL PUERTO A
PORTB	PUERTO B
TRISB	REGISTRO DE SELECCIÓN DE BITS DE ENTRADA O SALIDA DEL PUERTO B.
EEDATA	ALMACENA EL VALOR LEIDO DE LA EEPROM, DE LA LOCALIDAD A DONDE APUNTA EEADDR.
EECON1	REGISTRO DE CONTROL HABILITA LECTURA Y ESCRITURA DE EEPROM.
EEADDR	APUNTADOR QUE ALMACENA LA DIRECCIÓN QUE HABRA DE LEERSE EN LA EEPROM

EECON2 REGISTRO DE CONTROL DE ESCRITURA. PROTEJE CONTRA ALTERACIONES INDESEADAS DEL CONTENIDO DE LA EEPROM.

PCLATH PARTE ALTA DEL CONTADOR DEL PROGRAMA. 4 BITS, QUE JUNTO CON LOS 8 BITS DEL PCL, FORMAN LA DIRECCION COMPLETA CON LA CUAL CUAL PUEDEN DIRECCIONARSE 4096 LOCALIDADES, DE LA 0000H A LA 0FFFH. PCLATH PUEDE TAMBIEN VERSE COMO EL REGISTRO QUE CONTIENE EL NUMERO (0...15) DE LA PAGINA DE 256 BYTES EN DONDE HABRA DE DIRECCIONARSE LA MEMORIA.

Registros nuevos en el circuito 16F88

RCREG REGISTRO DE RECEPCION ASÍNCRONO DEL USART

RCSTA REGISTRO DE STATUS DE RECEPCION ASÍNCRONA DEL USART

TXREG REGISTRO DE TRASMISION ASÍNCRONA DEL USART

TXSTA REGISTRO DE STATUS DE TRASMISION ASINCRONA DEL USART

SPBRG REGISTRO PARA GENERACION DEL BAUD RATE DEL USART

PIR1 (PERIPHERAL INTERRUPT REGISTER) REGISTRO DE CONTROL DE INTERRUPCIONES DEL USART, EL CCPR1, TEMPORIZADOR1, TEMPORIZADOR2, Y EEPROM

PIR2 PERIPHERAL INTERRUPT REQUEST FLAG REGISTER

PIE1 (PERIPHERAL INTERRUPT ENABLE REGISTER) REGISTRO DE HABILITACION DE INTERRUPCIONES DEL USART, EL CCPR1, EL TEMPORIZADOR 1 Y LA EEPROM.

PIE2 PERIPHERAL INTERRUPT ENABLE REGISTER

PCON REGISTRO DE BANDERAS (STATUS) PARA CONOCER LA FRECUENCIA DE OPERACIÓN, Y EL MODO DE RESET DEL CONTROLADOR (POWER ON TIMER, BROWN OUT RESET)

WDTCON REGISTRO DE CONTROL DEL WATCH DOG TIMER

TMR1L TEMPORIZADOR 1, PARTE BAJA

TMR1H TEMPORIZADOR 1, PARTE ALTA

T1CON REGISTRO DE CONTROL DEL TEMPORIZADOR 1

TMR2 REGISTRO TEMPORIZADOR 2. TAMBIEN PUEDE USARSE PARA LA GENERACION DE PWM.

T2CON REGISTRO DE CONTROL DEL TEMPORIZADOR 2

SSPBUF BUFFER DEL PUERTO SERIAL SINCRONO

SSPCON CONFIGURACIÓN DEL PUERTO SERIAL SINCRONO

SSPADD SYNCRONOUS SERIAL PORT (I2C MODE) ADDRESS REGISTER

SSPSTAT SYNCRONOUS SERIAL PORT STATUS REGISTER

PR2 REGISTRO PARA CONTROL DEL PERIODO DEL TEMPORIZADOR 2

CCPR1L REGISTRO PARA EL MODULO CCP CAPTURA/COMPARA/PWM, PARTE BAJA

CCPR1H REGISTRO PARA EL MODULO CCP CAPTURA/COMPARA/PWM, PARTE ALTA

CCP1CON REGISTRO DE CONTROL PARA EL MODULO CCP CAPTURA/COMPARA/PWM

CMCON LOS BITS DEL PUERTO A ESTAN MULTIPLEXADOS CON EL COMPARADOR Y LAS FUNCIONES DEL

CVRCON VOLTAJE DE REFERENCIA. LOS REGISTROS CMCON (COMPARATOR CONTROL) Y CVRCON (VOLTAGE REFERENCE CONTROL) SE USAN PARA SELECCIONAR ESTAS FUNCIONES.

ADRESH PARTE ALTA DEL REGISTRO CON EL RESULTADO DE LA CONVERSION

ADRESL PARTE BAJA DEL REGISTRO CON EL RESULTADO DE LA CONVERSION

ADCON0 REGISTRO DE CONTROL DE CONVERTIDORES A/D

ADCON1 REGISTRO DE CONTROL DE CONVERTIDORES A/D

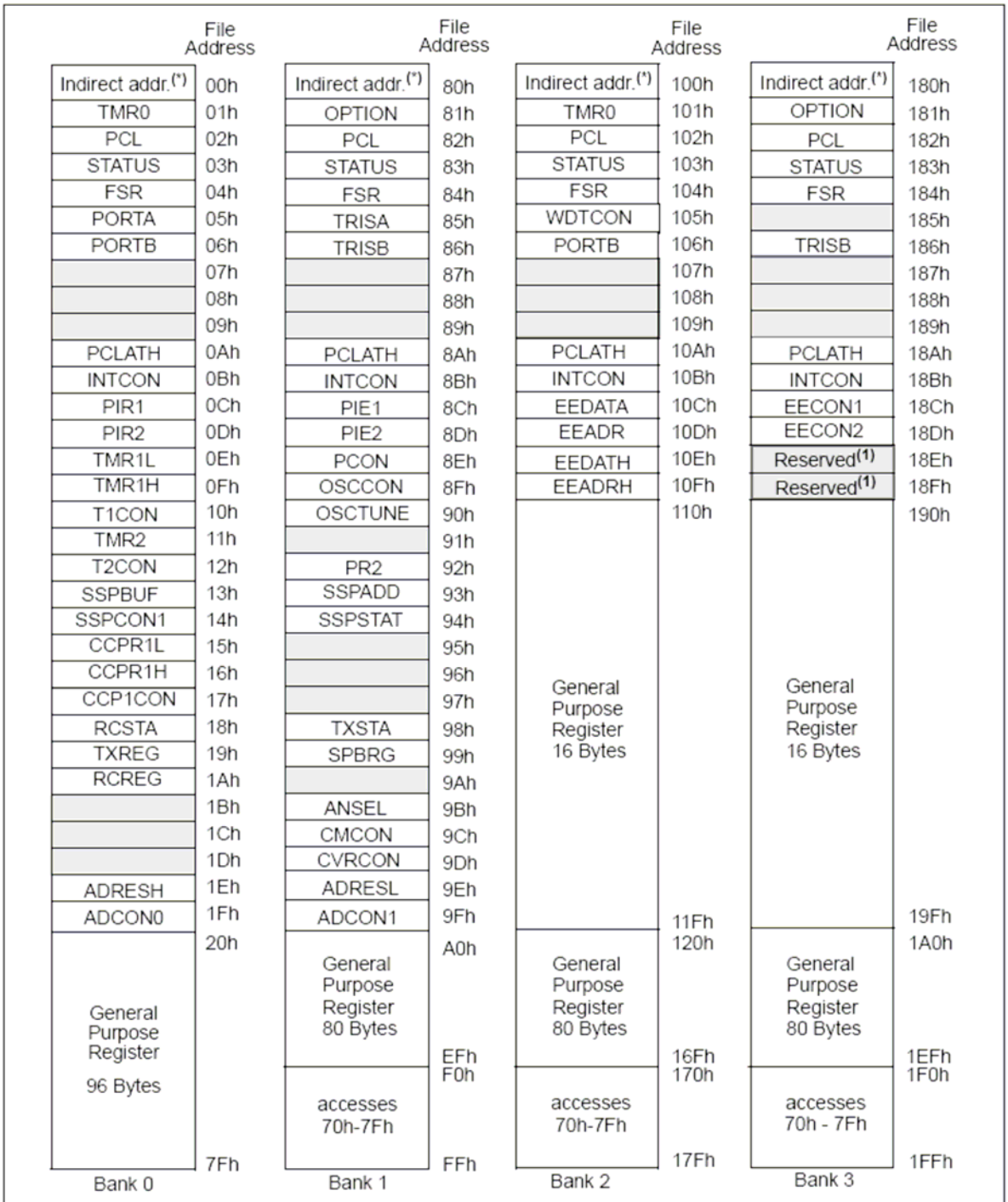
ANSEL REGISTRO DE SELECCIÓN DEL CONVERTIDOR A/D

OSCCON OSCILATOR CONTROL REGISTER (REGISTRO DE CONTROL DEL OSCILADOR)

OSCTUNE OSCILATOR TUNING REGISTER (REGISTRO DE AJUSTE DEL OSCILADOR)

NOTA IMPORTANTE: EN ESTE MANUAL SOLO SE DESCRIBEN CON DETALLE **LOS REGISTROS CON LETRAS RESALTADAS.** FAVOR DE USAR COMO REFERENCIA EL 16F88 DATA SHEET PARA LA INFORMACIÓN COMPLETA DEL RESTO DE LOS REGISTROS.

FIGURE 2-3: PIC16F88 REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: This register is reserved, maintain this register clear.

8.1 Registros PCL y PCLATH:

PCLATH (PC Latch), puede modificarse a través de la instrucción MOVWF, pero su ejecución solo almacena el dato y **no modifica en forma inmediata** la parte alta del contador del programa y por lo tanto no produce ningún salto en el flujo del programa.

El registro **PCL** puede ser accesado por las instrucciones MOVWF ó ADDWF. Su ejecución **modifica directamente la parte baja del contador del programa y carga también el registro PCLATH en la parte alta,** e induce por tanto un salto inmediato a otra localidad. En resumen, al modificar PCL, debe de tenerse cuidado previamente de inicializar también correctamente PCLATH, pues de lo contrario el programa efectuará un salto a una localidad en una página no deseada.

8.2 Registro de Status:

REGISTRO STATUS (DIRECCION 03H)

IRP	RP1	RP0	TO	PD	Z	DC	C
Bit 7							Bit 0

- los bits 0, 1 y 2 son el CARRY, HALF CARRY Y ZERO, son banderas que se activan con un valor igual a 1, cuando el resultado de una operación o instrucción genera un carry, un half carry o un valor igual a cero respectivamente.
- El bit 3 se llama POWER DOWN y su valor es de 1 después de una instrucción CLRWDT (CLEAR WATCH DOG TIMER) ó bien después de encender el sistema (POWER UP). El valor es de 0 después de ejecutar la instrucción SLEEP.
- El bit 4, se llama "TIMER OUT" tendrá un valor de 1 después de POWER UP, CLRWDT ó SLEEP y tendrá un valor de 0 si el WDT (WATCH DOG TIMER) activa su señal de alarma.
- Los bits 5 y 6 RP0, RP1 seleccionan el banco de memoria que habrá de accesarse. Si RP0 =0, RP1=0 se selecciona el banco 0. Si RP0=1, RP1=0 banco 1; RP0=0, RP1=1 banco 2; RP0=1, RP1=1 banco 3.
- El bit IRP se usa para direccionamiento indirecto, junto con el registro FSR, para seleccionar el banco de registros. Si IRP=0, se seleccionan los bancos 0 y 1; si IRP=1,se seleccionan los bancos 2 y 3

8.3 Registro OPTION :

Este registro controla varias **funciones del temporizador** (bits 0..5), **de la interrupción externa** (bit 6), así como las **resistencias de PULL UP del puerto** B (bit 7). En seguida se muestra un diagrama del registro de opción.

REGISTRO OPTION (DIRECCION 81H)

RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
Bit 7							Bit 0

- los bits 0, 1 y 2, toman un valor del 0 al 7 binario, y programan el divisor del temporizador y del WATCH DOG TIMER, de acuerdo a la siguiente tabla:

PS2 PS1 PS0	DIVISOR TMR0	DIVISOR WDT
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	1:128	1:32
110	1:128	1:128
111	1:256	1:128

- el bit 3, determina si el valor anterior se asigna al temporizador o al WDT. Si el valor es de 1, se asigna al WDT, si el valor es de 0, se asigna al temporizador.
- El **bit 4**, determina si el contador del temporizador se incrementa con el flanco ascendente (1) o con el flanco descendente (0) de la señal del pin 3, (RA4/T0CK1) del chip 16F88.
- El **bit 5** determina si la fuente de incremento del temporizador es la transición en el pin RA4/T0CLK1 (1) o el clock interno que maneja el ciclo de instrucción CLKOUT (0).
- El bit 6, determina, cuando su valor es de 1, que la interrupción externa se genera con el flanco ascendente del pin 6 del 16F88 (RB0/INT). Cuando su valor es de 0, entonces la interrupción se genera con el flanco descendente de la misma señal.

- El bit 7 determina, cuando su valor es de 1, que las resistencias de PULL UP en las entradas del puerto B estarán DESHABILITADAS. Si su valor es de 0, entonces dichas resistencias están HABILITADAS.

9. Registro temporizador/contador TMR0:

El registro TMR0 puede operar como un contador de los pulsos provenientes del bit RA4/T0CLK1 o como un temporizador. El modo de funcionamiento se selecciona con el bit 5 del registro de OPTION. El bit 5 de OPTION debe de ponerse en 1 si se selecciona el **modo contador**. Al mismo tiempo, el bit 4 determina, como se explicó arriba, si la cuenta en el registro TMR0 se incrementa con el flanco ascendente o descendente del bit externo RA4/T0CLK1.

Cuando se selecciona el **modo temporizador**, entonces el bit 5 del registro de OPTION debe de ponerse en un 0. En este modo de operación, el registro TMR0 funciona junto con un PREESCALADOR. Este preescalador puede programarse para dividir la cuenta de ciclos de instrucción, entre el valor seleccionado en el registro OPTION (en los bits PS0, PS1 y PS2), de acuerdo a la tabla mostrada en la sección 7.2. En total, se pueden generar períodos de espera de hasta un máximo de 256 x 256 ciclos de instrucción ó 65,536 microsegundos = 65.5 milisegundos (operando a 4 Mhz).

Si el usuario desea manejar el registro con base en el sistema de interrupciones, la interrupción TMR0 se genera cuando el registro pasa de una valor de FFH a 00H. El mecanismo de operación de las interrupciones, usa los bits 2 y 5 del registro INTCON y se explica en el capítulo siguiente. Debe de tomarse en cuenta que si el procesador se encuentra en el modo SLEEP, entonces la interrupción TMR0 no activará al procesador, ya que es deshabilitada durante ese modo.

10. El puerto serial síncrono/asíncrono USART

El USART (Universal Syncronous-Asyncronous Receiver Transmitter) es un puerto serial que se puede comfigurar en modo asíncrono full dúplex ó bien síncrono en modo half dúplex, pudiendo operar como MASTER o como SLAVE,. Los registros que intervienen en su operación son:, TXSTA, RCSTA, TXREG, RCREG y SPBRG.

TXSTA: es un registro de status para programar el formato y leer banderas de la señal de trasmisión.
RCSTA: es un registro de status para programar el formato y leer banderas de la señal de recepción.
TXREG: es el registro de trasmisión. Allí se carga el dato que habrá de trasmitirse.
RCREG: es el registro de recepción. Allí se carga el dato recibido para ser leído por el CPU.
SPRBG: es el registro de Baud Rate. Allí se carga un dato de 0...255 que define la velocidad de trasmisión-recepción.

TXSTA: D7

CSRC	TX9	TXEN	SYNC	-----	BRGH	TRMT	TX9D

TX9D: solo se usa cuando se selecciona formato de 9 bits, es el noveno bit de la palabra trasmitida.
TRMT: es el bit de status del registro de trasmisión. 1=registro vacío; 0=registro lleno
BRGH: bit de selección de velocidad de baud rate en modo asíncrono. 1=alta velocidad; 0=baja velocidad.
SYNC: modo de trasmisión del USART. 1= modo síncrono; 0=modo asíncrono.
TXEN: bit de habilitación de trasmisión. 1=trasmisión habilitada; 0=trasmisión deshabilitada.
TX9: bit de selección de formato. 1=formato de 9 bits; 0=formato de 8 bits
CSRC: se usa solo en modo síncrono. Selecciona la fuente de la señal de reloj. 1=reloj generado internamente (modo MASTER). 0=reloj generado en forma externa (modo SLAVE)

RCSTA:

SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D

RX9D: solo se usa cuando se selecciona formato de 9 bits, es el noveno bit de la palabra recibida.
OERR: bandera de error de overrun.1=ocurrió un error de overrun. 0=no hay error de overrun.
FERR: bander ade error de formato. 1=ocurrió un error de formato. 0=no hay error de formato.
ADEN:bit para habilitar la detección de dirección en redes de comunicación con varios dispositivos. Solo se usa en modo asíncrono, y con formato de 9 bits. 1=detección de dirección habilitada; 0=detección deshabilitada
CREN: bit de habilitación de recepción continua. 1=habilita recepción continua; 0=deshabilita recepción continua.
SREN: SINGLE RECEIVE ENABLE BIT. solo se usa en modo síncrono-MASTER. 1=habilita modo de recepción; 0=deshabilita.
RX9: bit de selección de formato de recepción. 1=selecciona formato de 9 bits; 0=seleccionaformato de 8 bits.
SPEN: configura los bits RB1 y RB2 en el circuito 16F88 para funcionar como señales del USART. 1=habilita USART; 0=deshabilita USART.

PROGRAMACION DEL BAUD RATE: el baud rate de operación del USART se programa cargando un valor en el registro SPRBG y seleccionando el bit BRGH. A 4 Mhz que es la frecuencia de operación de EVOLUPIC, deben de seguirse los datos de la siguiente tabla, para operar en modo asíncrono.

BAUD RATE bps	REGISTRO SPRBG (decimal)	BIT BRGH en registro TXSTA
1200	51	0
2400	25	0
9600	25	1
19200	12	1

RUTINA DE INICIALIZACION DEL USART, PARA TRASMISION ASINCRONA, 8 BITS, 2400 BPS (cristal 4 Mhz).

```
BSF STATUS,RP0          ; selecciona Banco 1
MOVLW D'25'             ; baud rate 2400
MOVWF SPBRG             ; carga en el registro de baud rate
MOVLW 0x20              ; habilitar trasmisión, 8 bits
MOVWF TXSTA             ; modo asíncrono, baja velocidad
BCF STATUS,RP0          ; selecciona Banco 0
MOVLW 0x90              ; receptor habilitado, 8 bits
MOVWF RCSTA             ; puerto serial habilitado
```

RUTINAS PARA LA TRASMISION/RECEPCION DE UN CARÁCTER ASCII

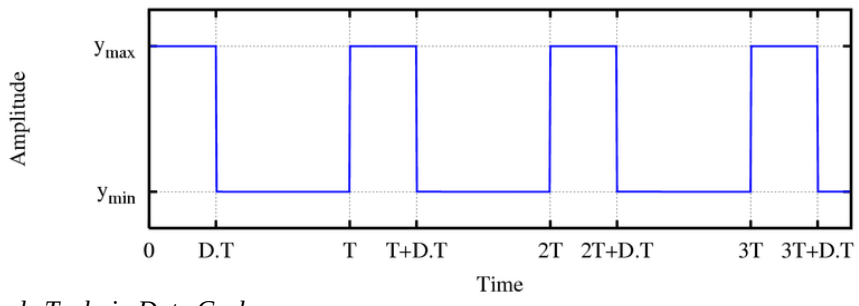
```
putchar:                ;caracter contenido en w
    btfss PIR1, TXIF
    goto putchar
    movwf TXREG         ;envía el contenido de w
    return

getchar:
    btfss  PIR1, RCIF   ;¿dato listo?
    goto   getchar      ;loop de espera hasta que el dato esté listo
    movfw  RCREG        ;caracter recibido en w
    return
```

11. PWM Pulse Wide Modulation

11.1 Aplicaciones de PWM

La modulación de señales por ancho del pulso es una técnica que se ha usado extensivamente durante décadas en los sistemas de control. Se trata de una señal digital que permite implementar un control de tipo analógico en sistemas lineales. Su aplicación más típica es el control proporcional de velocidad en motores de DC, razón por la cual es muy usado en aplicaciones de robótica. También se emplea en la implementación de "dimmers", es decir controles proporcionales de intensidad en lámparas incandescentes, y en general para la generación de señales en frecuencias de audio. El microcontrolador 16F88 puede generar señales PWM con una resolución de 10 bits, y a una frecuencia de operación programable de acuerdo a las necesidades del usuario.



11.2 Ciclo de Trabajo Duty Cycle

El pin CCP1 genera la señal PWM. Este pin se encuentra multiplexado con el bit RB3, razón por la cual, el bit 3 del registro TRISB debe ser previamente puesto en cero para habilitar CCP1. Hay dos parámetros que definen a la señal PWM: la

frecuencia de operación y el ciclo de trabajo (duty cycle). El período de operación de la señal, así como el ciclo de trabajo se programan usando las siguientes fórmulas (Frec. Xtal= 4 Mhz).

[frecuencia de operación PWM (khz)] = 1000/ [(valor decimal en PR2+1)* (PS)]

En donde PS es el valor de preescalamiento (1, 4, ó 16), programable con los bits del registro [T2CON<1,0>] cuyos nemotécnicos son: T2CLKPS1 y T2CLKPS0. Si estos bits valen 00, PS=1, si valen 01, PS=4, si valen 10, PS=16.

[ciclo de trabajo en %] = [CCPR1L:CCP1CON<5,4>]* (PS)* (25)

En donde [CCPR1L:CCP1CON<5,4>] es un valor de 10 bits, con CCP1L aportando los 8 bits más significativos y CCP1CON<5,4> los dos bits menos significativos. Dicho valor debe ser convertido a decimal para ingresarlo en la fórmula.

12. Interrupciones del sistema.

El chip 16F88 cuenta con múltiples fuentes de interrupción asociadas a la ocurrencia de alguno de los siguientes eventos y que permiten implementar un software del tipo multitareas en su aplicación. Algunas de estas fuentes de interrupción son:

- La interrupción externa en el pin RB0/INT del chip, con flanco ascendente o descendente.
- El overflow en el temporizador 0, el temporizador 1 ó el temporizador 2.
- Cuando en el USART, el registro de recepción está lleno o el de trasmisión vacío.
- Cualquier cambio de nivel en los pines RB4...RB7
- Cuando se ha completado la escritura de un dato en la EEPROM.
- Interrupción del módulo CCP, CAPTURA/COMPARA/PWM
- Interrupción del módulo comparador.
- Interrupción generada por el ciclo de escritura en la EEPROM.

El vector de inicio de la subrutina de atención a interrupciones es la dirección 0004H. Después de la ocurrencia de una interrupción que se encuentre habilitada, el programa efectuará automáticamente un llamado a subrutina en esa dirección. En esta **subrutina de atención a interrupciones**, el programador debe primeramente leer los registros de status de las interrupciónes habilitadas para conocer cual está activa en ese momento y efectuar un salto al subprograma correspondiente. Para regresar de la subrutina de interrupción, se ejecuta la instrucción RETFIE, la cual habilita automáticamente el bit de interrupción global GIE

El registro INTCON controla la habilitación y deshabilitación de algunas interrupciones del sistema. Sus bits tiene las funciones que se indican enseguida.

REGISTRO INTCON (DIRECCION 0BH)

GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
Bit 7							Bit 0

- el **BIT 0** es una **bandera** que se pone en valor 1, si alguno de los bits del puerto RB4...RB7 cambió de valor y en 0 si ninguno de estos bits cambió su valor.
- el **BIT 1** es una **bandera** que se pone en 1, si se activa la interrupción externa (señal RB0/INT) del 16F88 y tomará un valor de 0 si no se activa dicha señal. En el registro de OPCION debe programarse si la interrupción se genera con al flanco ascendente o descendente.
- el **BIT 2** es una **bandera** que se pone en 1, si el contador del temporizador 0 del circuito sufre un overflow, es decir, excede su cuenta máxima. Y en 0 si dicho contador no excede su cuenta máxima.
- en el **BIT 3**, deberá escribirse un valor de 1 para **habilitar** la interrupción de los bits RB4...RB7 (ver bit 0) y de 0 para deshabilitar dicha interrupción.
- en el **BIT 4** deberá escribirse un valor de 1 para **habilitar** la interrupción externa (ver bit 1) y de 0 para deshabilitar dicha interrupción.
- en el **BIT 5** deberá escribirse un valor de 1 para **habilitar** la interrupción del temporizador (ver bit 2) y de 0 para deshabilitar dicha interrupción.
- En el **BIT 6** deberá escribirse un valor de 1 para **habilitar** la interrupción “escritura de un dato en la EEPROM completado” y un valor de 0 para deshabilitar dicha interrupción. El bit 4 del registro EECON1 es la bandera correspondiente que maneja la interrupción y se pone en un valor de 1, cuando está activa.
- el **BIT 7** corresponde al **habilitador GLOBAL** de las interrupciones (GIE). Debe de tener un valor de 1 para habilitar todas la interrupciones y de 0 para deshabilitarlas.

Cuando se genera una interupción, el bit GIE es automáticamente puesto en 0 para impedir que se generen nuevas interrupciones. El contador del programa se carga con la dirección 0004H y la dirección de retorno es almacenada en el STACK o pila. Una vez que la subrutina de interrupción está ejecutándose (a partir de la dirección 0004H), la fuente de la interrupción puede

ser determinada a través de un poleo en los bits 0, 1 y 2 del registro INTCON y bit 4 del registro EECON1. Dentro de la subrutina de atención a las interrupciones, debe también de escribirse un 0 en el bit de bandera correspondiente, para impedir que la misma interrupción vuelva a activarse una y otra vez. La instrucción RETFIE, habilita de nuevo la bandera GIE.

13. Programación de la EEPROM:

La memoria EEPROM residente en el chip 16F88, posee 256 bytes. Dichas localidades pueden ser leídas o bien, puede escribirse en ellas durante la ejecución de un programa. Sus direcciones son de la 0 a la FF H. El acceso a esta memoria es a través de **direccionamiento indirecto** utilizando 4 de los registros SFR (special fuction registers) EECON1, EECON2, EEDAT, y EEADR.

EEDAT guarda el dato que habrá de escribirse en la EEPROM, mientras que **EEADR** guarda la dirección. **EECON1** es un registro cuyos bits permiten al usuario habilitar o deshabilitar la lectura y escritura en la EEPROM. **EECON2** es un registro de control usado para evitar escrituras accidentales, de tal manera que deben escribirse en dicho registro los valores 55H y AAH, antes de iniciar un ciclo de escritura. En seguida se muestran los bits de control del registro EECON1:

REGISTRO EECON1 (DIRECCION 9CH)

-	-	-	EEIF	WRERR	WREN	WR	RD
Bit 7							Bit 0

- para leer la EEPROM, debe de escribirse un 1 en el **BIT 0** del registro. Este bit tomará automáticamente el valor de 0 después de haberse efectuado la lectura del dato.
- Para iniciar el ciclo de escritura en la EEPROM, debe escribirse un 1 en el **BIT 1** del registro. Una vez terminado el ciclo de escritura, el bit tomará automáticamente el valor de 0.
- El **BIT 2** habilita o deshabilita el ciclo de escritura. Si vale 1, se habilita y si vale 0, se deshabilita.
- El **BIT 3**, es una bandera que anuncia, después de un ciclo de escritura, que ésta fue completada en forma exitosa. Si el valor es 1, existió un error y si el valor es de 0, entonces la escritura fue completada sin error.
- El **BIT 4** es una **bandera de interrupción** asociada a la escritura en la EEPROM. Si el valor es de 1, el ciclo de escritura terminó. Si el valor es de 0, el ciclo de escritura no ha iniciado ó no ha concluído.
- Los bits 5, 6 y 7, no se usan.

```
;subrutina para leer dato de EEPROM
;para usar esta subrutina, debe escribirse antes la dirección deseada
;en el registro EEADR (BANCO2). La rutina regresa con el dato en w.

EERD:  BANCO3
       BCF      EECON1,EEPGD     ;APUNTA HACIA EEPROM DE DATOS
       BSF      EECON1,RD        ;HABILITA EL BIT 0 (RD) DEL REGISTRO EECON1
       BANCO2
       MOVF     EEDATA,W         ;TRANSFIERE EL DATO EN EEDATA A W,
       RETURN

;subrutina para escribir dato en EEPROM
;para usar esta subrutina, debe escribirse antes la dirección DE LA EEPROM
;en el registro EEADR (BANCO2) y el dato en el registro EEDATA (BANCO2)
;la rutina regresa una vez que el dato fue escrito en la localidad deseada.

EEWR:  BANCO3                   ;HABILITA EL BANCO 3
       BCF      EECON1,EEPGD    ;APUNTA HACIA EEPROM DE DATOS
       BSF      EECON1,WREN     ;HABILITA ESCRITURA EN EEPROM
       BCF      INTCON, GIE     ;DESHABILITA INTERRUPCIONES
       MOVLW    H'55'           ;PREPARA SECUENCIA DE SEGURIDAD
       MOVWF    EECON2          ;ESCRIBE PRIMER DATO DE SECUENCIA
       MOVLW    H'AA'           ;SEGUNDO DATO
       MOVWF    EECON2          ;ESCRIBE SEGUNDO DATO DE SECUENCIA
       BSF      EECON1,WR       ;INICIA CICLO DE ESCRITURA
EW:    BTFSC    EECON1,WR       ;MALLA PARA ESPERAR AL FINAL DEL CICLO
       GOTO     EW              ;SI WR=1, CICLO DE ESCRITURA AUN NO TERMINA
       BCF      EECON1, WREN    ;DESHABILITA ESCRITURA
       BSF      INTCON, GIE     ;HABILITA INTERRUPCIONES
       RETURN
```

14. Funciones especiales:

14.1REGISTRO DE CONFIGURACION:

El 16F88 cuenta con un registro de configuración de 14 bits, que solamente puede accesarse durante el ciclo de programación del chip. Su dirección es la 2007H. El valor de los bits del registro de configuración controlan la operación de diversas funciones especiales, como son: la frecuencia del oscilador, el WATCH DOG, el POWER ON TIMER, el MASTER CLEAR, el BROWN OUT RESET, la programación LOW VOLTAGE PROGRAMMING y la función CODE PROTECT para memoria de datos (EEPROM) y para memoria de código. El código usualmente empleado es el 3F69H. Por favor lea el documento:

"Guía General para la programación del sistema EVOLUPIC *Bootloader* 16F88.pdf"

```
ORG 0x2007
DATA 3F69H     ;para EVOLUPIC Bootloader 4 MHZ EXTERNO
```

REGISTER 15-1: CONFIG1: CONFIGURATION WORD 1 REGISTER (ADDRESS 2007h)

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
CP	CCPMX	DEBUG	WRT1	WRT0	CPD	LVP	BOREN	MCLRE	FOSC2	$\overline{\text{PWRTEN}}$	WDTEN	FOSC1	FOSC0
bit 13													bit 0

14.2 POWER UP TIMER ENABLE, PWRTE

Al seleccionar en el registro de configuración la opción power up timer, y con el objeto de permitir la estabilización del oscilador, se mantiene el pulso de reset activo hasta después de 72 ms después de haber conectado la energía. En el caso que se esté usando un oscilador de cristal, se genera automáticamente un retraso adicional de 2048 pulsos de reloj, antes de que el pulso de reset termine. Estos retrasos permiten la estabilización del cristal antes de que el microcontrolador inicie su operación.

14.3 BROWN OUT RESET

El 16F88 integra un novedoso circuito de protección automático, el cual genera un RESET al detectar picos de voltaje en la fuente de alimentación Vdd de 5v. Estos picos son generalmente inducidos a través del eliminador de baterías, por efecto de variaciones bruscas del voltaje de alimentación 127 VCA ó bien por ruido inducido a través de los cables que conectan las entradas y salidas digitales del microcontrolador, (cuando éstas no se encuentran adecuadamente aisladas) a sensores o actuadores remotos. La función es especialmente útil en ambientes industriales y garantiza la operación continua del microcontrolador. Para activar esta función especial se usa el comando **_BOREN_ON_.** El bit 0 (BOR) del registro especial PCON es una bandera que indica: 0=ocurrió un reset BROWN OUT RESET y, 1=no ocurrió un BOR.

14.4 WATCH DOG TIMER

El WDT, es un circuito de vigilancia que permite generar un pulso de **reset automático** en caso de que el 16F88 se salga de operación por alguna inestabilidad en el voltaje de alimentación en su fuente de poder ó alguna falla en la ejecución del programa. La función es sumamente importante para **evitar que el sistema necesite intervención manual** externa para dar reset al procesador. El WDT funciona como un contador de eventos cada 18 ms, el cual genera un reset al sistema cuando la cuenta llega a un máximo y genere un **TIMEOUT.**

La activación del WDT, **debe de hacerse desde el registro de configuración**,. La dirección del registro es la 2007H. Debe recordarse que el registro de configuración no puede accesarse desde el programa ejecutable del microcontrolador, sino directamente debe programarse en el programa fuente. (ver ejemplo en 13.1)

Además, desde el programa ejecutable, el bit 3 del registro OPTION, debe de programarse como PSA=1, para asignar el valor del preescalador al WDT. Adicionalmente, en los bits PS0, PS1, PS2 del registro OPTION debe escribirse, desde el programa, un valor entero del 0 al 7. Cualquier valor diferente a 0, eleva el período de activación del WDT a 18 milisegundos, multiplicado por 2 elevado a ese valor, de acuerdo a la tabla mostrada en la figura. Por ejemplo, si el valor de los bits PS0, PS1 y PS2 es de 5, el período de TIMEOUT será de 18ms x 32 = 576 ms. El TIMEOUT máximo para el WDT es de 2.3 segundos.

Una vez que el WDT está activado, a través de la instrucción CLRWDT, se reinicia desde 0 su período de activación. Entonces dicha instrucción debe de ejecutarse regularmente dentro de la malla principal en el programa, con un período que debe de ser MENOR al TIMEOUT programado para el WDT. Cuando por alguna causa de malfuncionamiento del 16F88 el programa se sale de su operación normal y por consecuencia la instrucción CLRWDT no se ejecuta, entonces, al llegar a un máximo la cuenta en el WDT (TIMEOUT), el circuito genera automáticamente un RESET que reinicia la operación del 16F88.

14.5 SLEEP

El 16F88 cuenta con una función que le permite operar en un modo de muy bajo consumo, por ejemplo en el caso de un sistema con alimentación de energía solar ó pilas. Si se tiene una aplicación en la cual el microcontrolador no desempeña ninguna función útil hasta la ocurrencia de alguna interrupción, puede abatirse el consumo promedio del circuito a niveles cercanos a 0 ma (1 uA). La función de SLEEP se habilita con la instrucción del mismo nombre. A partir de su ejecución, los circuitos del oscilador maestro cesan de funcionar, siendo de esta forma el consumo de corriente de casi cero. Solamente la ocurrencia de alguna interrupción externa en el pin RB0/INT, la interrupción por algún cambio en los niveles de las entradas en el puerto B, la interrupción proveniente de la EEPROM, ó bien un reset en el pin MCLR del 16F88 puede restaurar la operación normal del circuito. Antes de entrar al estado de SLEEP, debe de inhibirse la operación del WDT para evitar que éste reactive al circuito a través de su reset automático.

14.6 CODE PROTECT

El microcontrolador 16F88 cuenta con esta opción para evitar, de ser necesario, que alguna persona pueda copiar el código del programa contenido en la memoria FLASH del chip. Si usted desea proteger su programa entonces deberá añadir en la línea de configuración el comando **_CP_ON_**. Sin embargo, debe de tenerse cuidado de no manipular indebidamente este bit, ya que, una vez habilitado el modo "CODE PROTECT" será imposible accesar de nuevo el código almacenado en la memoria FLASH. También es importante señalar que un chip que ha sido protegido, no puede ser leído, pero sí puede ser borrado y reprogramado. Si desea proteger únicamente los datos de la memoria EEPROM, entonces se usa el comando **_CPD_ON_**

15. Puertos digitales :

El sistema 16F88 cuenta con dos puertos digitales, el puerto A, con 8 bits y el puerto B con 8 bits disponibles. Ambos puertos son bidireccionales, ésto es, pueden programarse como entradas o como salidas, de acuerdo a los registros de dirección de datos, llamados "TRIS", en el caso del puerto A es "TRISA" y del puerto B es "TRISB". En la tarjeta EVOLUPIC, le han sido conectados entradas con 4 microswitches para el puerto A y salidas de 8 LEDS para el puerto B, así como un relevador conectado al pin RA0. La asignación de funciones en cada uno de los bits, se muestra en la siguiente tabla. Por favor tome nota de que cada entrada y salida se encuentran disponibles en varios conectores (LCD, TECLADO y AUX) en la tarjeta, de tal manera que el usuario pueda conectar dispositivos externos.

PUERTO	PIN EN HEADER 16x	FUNCION
PUERTO A		
RA0	PIN 1	ACTIVA/DESACTIVA RELEVADOR
RA1	PIN 3	MICROSWITCH A1, CONTROL LCD
RA2	PIN 5	MICROSWITCH A2, CONTROL LCD
RA3	PIN 7	MICROSWITCH A3, SENSOR DE TEMPERATURA
RA4	PIN 9	MICROSWITCH A4
PUERTO B		
RB0	PIN 2	LED B0, TECLADO Y1
RB1	PIN 4	LED B1, TECLADO Y2,
RB2	PIN 6	LED B2, TECLADO Y3, RECEPCION SERIAL RX
RB3	PIN 8	LED B3, TECLADO Y4
RB4	PIN 10	LED B4, TECLADO X1
RB5	PIN 12	LED B5, TECLADO X2, TRASMISION SERIAL TX
RB6	PIN 14	LED B6, TECLADO X3
RB7	PIN 16	LED B7, TECLADO X4

Antes de poder escribir y leer de los puertos, es necesario primero programar qué bits serán entradas y salidas, usando las siguientes instrucciones:

```
;MACROS PARA SELECCIONAR EL BANCO

BANCO0      MACRO
            BCF         STATUS,RP0
            BCF         STATUS,RP1
            ENDM
BANCO1      MACRO
            BSF         STATUS,RP0
            BCF         STATUS,RP1
            ENDM

;PARA INCIALIZAR LOS PUERTOS A Y B
;RA0= SALIDA, RA1...RA4= ENTRADAS
;RB0...RB7= SALIDAS

            BANCO1
            MOVLW       H'1E'
            MOVWF       TRISA
            CLRF        TRISB
            BANCO0
```

15.1LEDS Y MICROSWITCHES:

Una vez inicializados los puertos de la forma mostrada, se puede desde el programa escribirse en los LEDS o leer desde los microswitches, considerando los diagramas electrónicos que se muestran enseguida. El objetivo de los LEDS y los microswitches es dar al usuario la posibilidad de realizar emulaciones de sensores digitales y salidas para la activación de actuadores. Todas las señales de los puertos están disponibles en en conector header de 14x, para su conexión a interfaces externas.

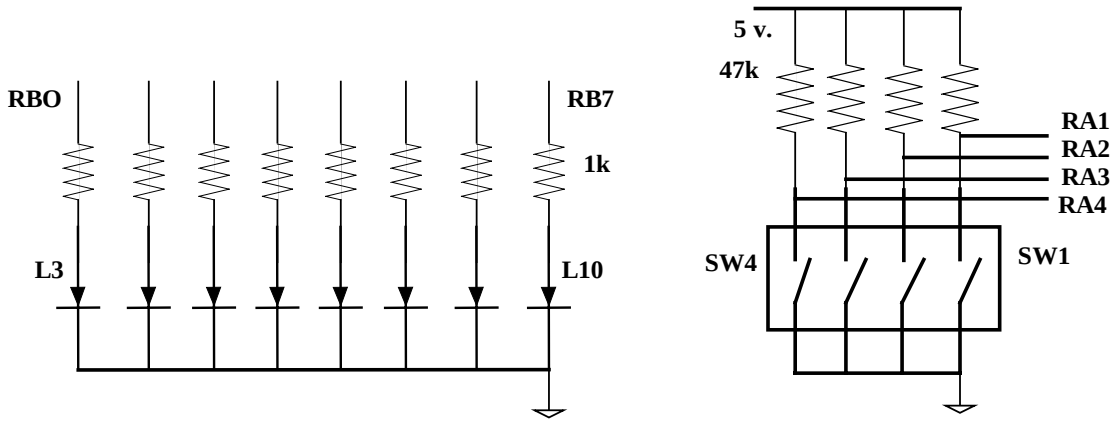


FIGURA 8: DIAGRAMA DE LOS LEDS Y LOS MICROSWITCHES

15.2 RELEVADOR

Mediante el manejo del bit RA0 del puerto A, el procesador 16F88 puede activar un relevador integrado en la tarjeta EVOLUPIC. Los datos nominales de este relevador son : un polo un tiro, activación con 9 volts DC y contactos de 127 VCA @ 10 Amperes. Este relevador puede ser usado ya sea como un sensor digital ABIERTO- CERRADO para alertar a otros dispositivos del estado de alguna alarma, o bien como actuador para activar dispositivos externos como focos, válvulas, solenoides, motores, etc.

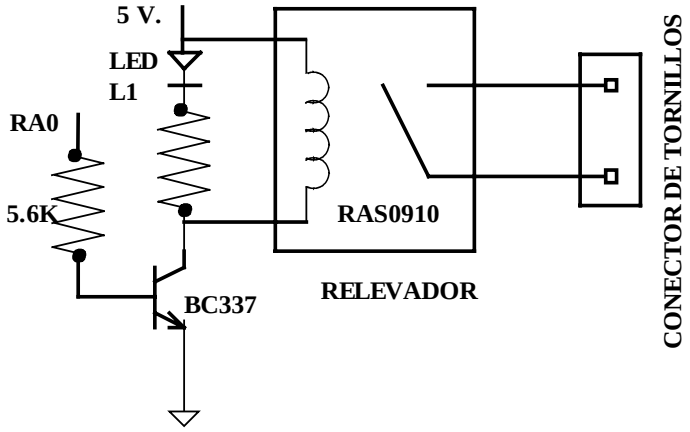


FIGURA 9: CONEXIÓN DEL RELEVADOR

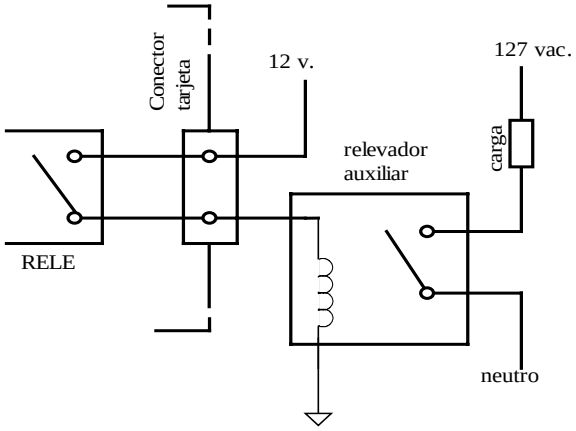


FIGURA 10: CONEXIÓN DE UN RELEVADOR AUXILIAR

15.3 CONECTORES PARA EL TECLADO Y HEADER AUXILIAR.

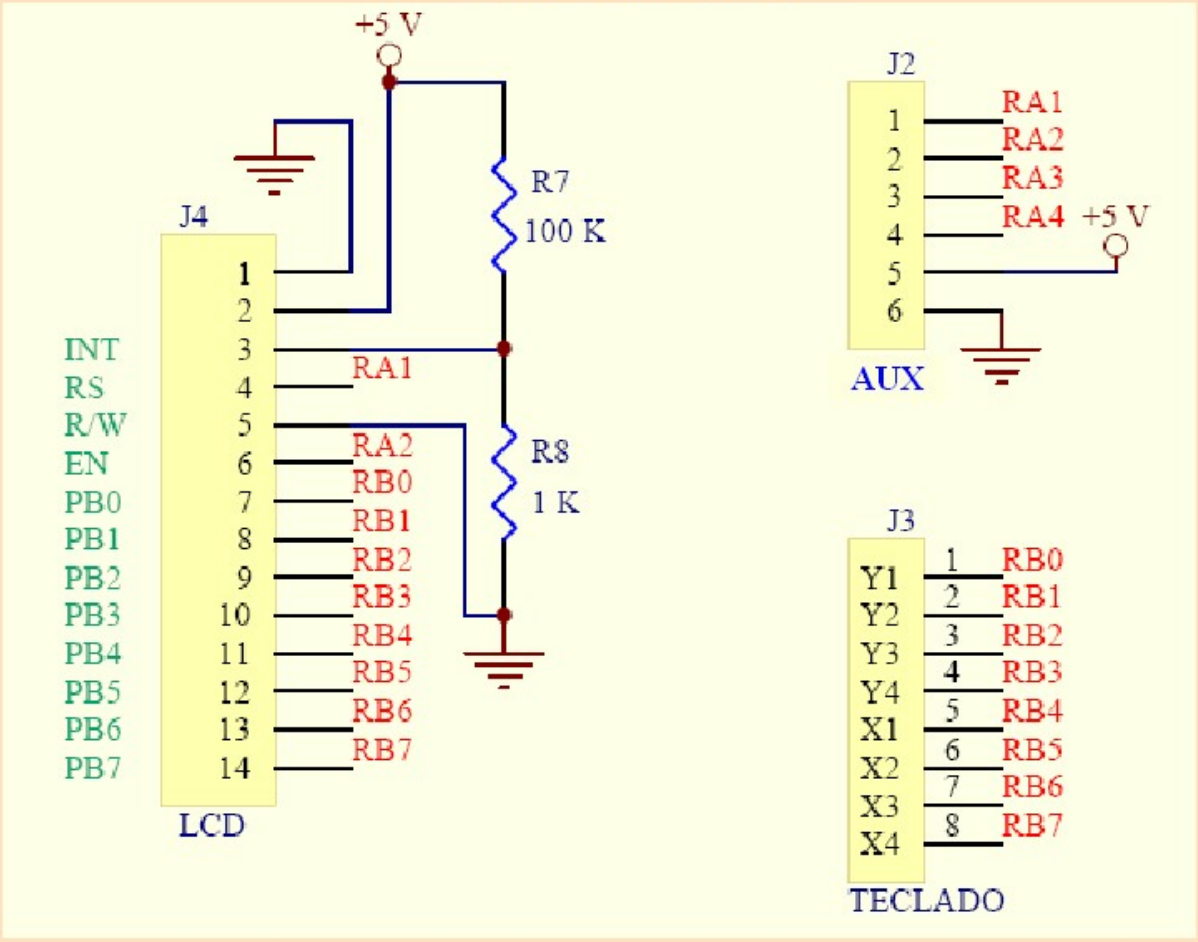
El conector para el teclado tiene 8 señales marcadas con los números del 1 al 8, correspondientes a RB0...RB7 respectivamente, para insertar el teclado matricial. Existe también un header auxiliar con 6 señales, marcadas con los números del 1 al 6. Las señales son respectivamente RA1...RA4, 5 volts Y GND. Favor de ver el diagrama en la siguiente hoja.

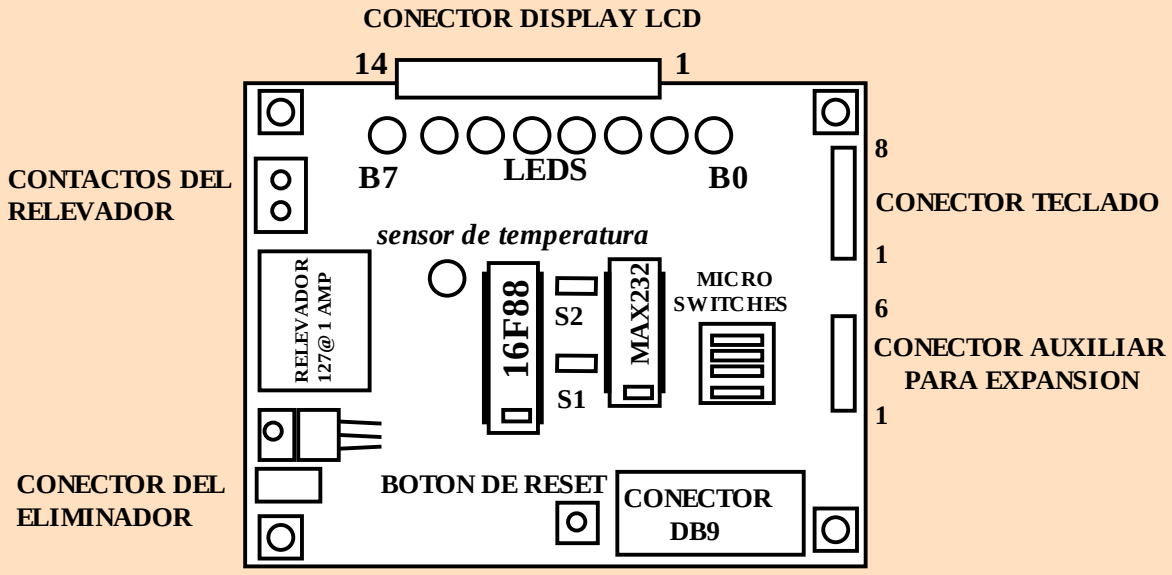
15.4. CONECTOR A LCD:

Este conector dispone de 14 señales, mostradas en la tabla de abajo.

LCD	EVOLUPIC	FUNCION	LCD	EVOLUPIC	FUNCION
1		Tierra	8 DB1	RB1	DATOS
2		5 volts.	9 DB2	RB2	DATOS
3 INT		Control de Intensidad	10 DB3	RB3	DATOS
4 RS	RA1	0=comando 1=datos	11 DB4	RB4	DATOS
5 R/W	TIERRA	0=escribir en LCD 1=leer	12 DB5	RB5	DATOS
6 EN	RA2	Enable modo pulso	13 DB6	RB6	DATOS
7 DB0	RB0	DATOS	14 DB7	RB7	DATOS

ASIGNACION DE SEÑALES DE LOS PUERTOS A Y B EN LOS CONECTORES DE EVOLUPIC Bootloader 16F88





16. El set de instrucciones.

Existen un total de 35 instrucciones. Todas las instrucciones son palabras de 14 bits, divididas en dos partes: el código de operación y el operando. Los operandos, pueden ser bytes o bits de memoria o registros. De esta forma se puede hablar de instrucciones “orientadas a bytes” u “orientadas a bits”.

Todas las instrucciones, con excepción de las que modifican el contenido del contador del programa (como son los saltos y llamados a subrutina) se ejecutan en un ciclo de instrucción, es decir, 4 ciclos de reloj. Para un sistema funcionando a 4 Mhz, cada instrucción se ejecuta en 1 microsegundo. Si la instrucción modifica el contenido del contador del programa, entonces el tiempo de ejecución es de 2 ciclos de instrucción ó 2 microsegundos para el ejemplo a 4 Mhz. El grupo de 35 instrucciones es el siguiente:

MOVE GROUP

movf	**f,d**	**move f**
movwf	**f**	**move w to f**
movlw	**k**	**move literal to w**
clrf	**f**	**clear f**
clrw		**clear w**
swapf	**f,d**	**swap nibbles in f**

ARITHMETIC GROUP

addwf	**f,d**	**add w and f**
addlw	**k**	**add literal to w**
subwf	**f,d**	**substract w from f**
sublw	**k**	**substract w from literal**
indf	**f,d**	**increment f**
incfsz	**f**	**increment f, skip if 0**
decf	**f,d**	**decrement f**
decfsz	**f**	**decrement f, skip if 0**

LOGIC GROUP

andwf	**f,d**	**and w and f**
andlw	**k**	**and literal to w**
iorwf	**f,d**	**inclusive or w and f**
iorlw	**k**	**inclusive or literal to w**
xorwf	**f,d**	**exclusive or w and f**
xorlw	**k**	**exclusive or literal to w**
comf	**f,d**	**complement f**
rlf	**f,d**	**rotate left f, through carry**
rrf	**f,d**	**rotate right f, through carry**

BIT GROUP

bcf	**f,b**	**bit clear in f**
bsf	**f,b**	**bit set in f**
btfsc	**f,b**	**bit test in f, skip if clear**
btfss	**f,b**	**bit test in f, skip if set**

CONTROL GROUP

clrwdt		**clear watchdog timer**
sleep		**go into sleep mode**
nop		**no operation**

BRANCH GROUP

goto	**k**	**goto address**
call	**k**	**call soubrutine**
return		**return from subrutine**
retlw	**k**	**return with literal in w**
retfie		**return from interrupt**
incfsz	**f**	**increment f, skip if 0**
decfsz	**f**	**decrement f, skip if 0**
btfsc	**f,b**	**bit test in f, skip if clear**
btfss	**f,b**	**bit test in f, skip if set**

16.1 OPERANDOS:

Los operandos pueden asignarse con las letras F, W, B, K, D. Cada letra tiene el significado siguiente:

F: designa alguna localidad de memoria (file register), de alguno de los 4 bancos de los llamados “registros especiales”, o bien, alguna de las 368 localidades de memoria RAM.

W. designa el acumulador del 16F88.

B. designa alguno de los 8 bits del registro especial o localidad de memoria elegido.

K. designa una constante ó una dirección.

D. designa el destino de la operación. Si D=0, el destino es el registro W. Si D=1, entonces el destino es el registro ó localidad de memoria F.

Directivas del programa ensamblador:

Dentro del archivo del programa fuente, es decir del programa escrito en lenguaje ensamblador, se puede, con ayuda de la directiva EQU, definir previamente algunos valores para la facilidad de su identificación. Algunas de las definiciones más utilizadas son las siguientes:

W	EQU	H'0000'	PORTA	EQU	H’0005’
F	EQU	H'0001'	PORTB	EQU	H’0006’
STATUS	EQU	H'0003'	TRISA	EQU	H’0085’
RP1	EQU	H'0006'	TRISB	EQU	H’0086’
RP0	EQU	H'0005'	PCLATH	EQU	H’000A’

Estas declaraciones, junto con las del resto de los registros, así como los bits individuales de cada registro, se almacenan en un solo archivo que se denomina 16F88.inc y que es parte de las librerías ya incluídas en el programa MPLAB, del cual se habla más adelante. Entonces, es suficiente escribir, dentro del programa fuente en lenguaje ensamblador, la directiva: **include <p16F88.inc>**

Una vez establecidas estas equivalencias, podemos poner algunos ejemplos con instrucciones. Observe que, en todas ellas es posible usar las equivalencias o bien escribir directamente el valor numérico. Por ejemplo:

Etiqueta	**Instrucción**	**Operando**	**Forma general:**	
EJEMPLO1	BCF	STATUS,RP0	BCF	F,D

Resultado: Bit Clear F. Pon en cero el bit RP0 del registro STATUS.

EJEMPLO2 BCF 3,5

Resultado: mismo que en el ejemplo anterior, pero usando ahora las constantes directamente al escribir la instrucción. Observe como el hecho de escribir directamente palabras como “STATUS” en lugar del número 3, facilitan mucho la comprensión.

EJEMPLO3 BTFSS STATUS,RP1 BTFSS F,B

Resultado: Bit Test F, Skip if Set. Si el bit RP1 del registro designado es cero, ejecuta la siguiente instrucción, si el bit es 1, entonces no ejecuta la siguiente instrucción, pero sí la que sigue a ésta.

EJEMPLO4 ADDLW 3AH ADDLW K

Resultado: Add literal to W. Suma el registro W con la constante 3AH. El resultado lo pone en W.

EJEMPLO5 DECFSZ 20H,W DECFSZ F,D

Resultado: Decrement F, Skip if Zero. Decrementa el valor de la localidad 20H. El resultado lo almacena en el registro W. Si el resultado es 0, no ejecuta la siguiente instrucción, sino la inmediata después de ésta. Si el resultado es diferente de 0, entonces ejecuta la siguiente instrucción.

EJEMPLO6 DEC 2AH,W DEC F,D

Resultado: Decrementa la localidad 2AH, el resultado lo pone en el registro W.

EJEMPLO7 ANDWF 2B,F ANDWF F,D

Resultado: hace la operación lógica AND entre la localidad 2B y el registro W. El resultado lo pone en la localidad 2B.

EJEMPLO8 BTFSC 35H,3 BTFSC F,B

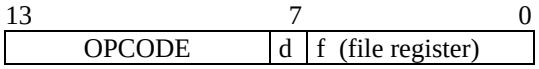
Resultado: Bit Test F, Skip if Clear. Hace una prueba sobre el bit 3 de la localidad 35H. Si el valor es 0, no ejecuta la siguiente instrucción, pero sí la siguiente. Si el valor es 1, entonces ejecuta la siguiente instrucción.

16.2 FORMATO DE LAS INSTRUCCIONES:

Todas las instrucciones llevan alguno de los siguientes formatos, dependiendo de la función que desempeñen: Instrucciones orientadas a byte. Instrucciones orientadas a bit. Instrucciones de manejo de constantes (en la literatura de Microchip, las constantes se denominan "literals") y, finalmente instrucciones de salto CALL y GOTO.

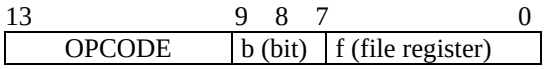
Enseguida, se muestra el formato para cada tipo de instrucción. Las palabras son de 14 bits. El trabajo de decodificación de cada formato es realizado por el programa ensamblador, razón por la cual el programador no requiere de decodificar manualmente cada instrucción..

INSTRUCCIONES ORIENTADAS A BYTES:



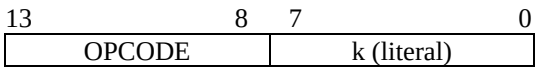
Si d=0, destino es W, si d=1, destino es f

INSTRUCCIONES ORIENTADAS A BITS:



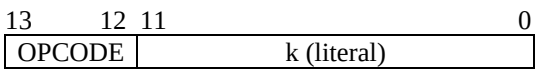
b selecciona el bit del registro f, (valor de 0 a 7)

INSTRUCCIONES MANEJO DE CONSTANTES:



k es la constante en la instrucción.

INSTRUCCIONES CALL Y GOTO:



k es la dirección inmediata en 12 bits.

16.3 MANEJO DE TABLAS:

Es importante recordar que el microcontrolador 16F88 opera con una arquitectura HARVARD. Lo anterior hace que el manejo de tablas sea distinto a procesadores con arquitectura VON NEUMANN, en donde la memoria de código y datos es compartida, y entonces, a través de direccionamiento indirecto es posible recuperar los datos de la tabla. En el caso de la arquitectura Harvard, la tabla está en la memoria de código y debe entonces de manejarse como parte del programa ejecutable. La forma de resolverlo es a través instrucciones RETLW. La tabla forma parte de una subrutina que en este ejemplo lleva la etiqueta TABLA.

Supongamos que deseamos crear una tabla de 4 datos. Y que estos 4 datos son los caracteres ASCII de las letras H,O,L y A. Dentro del programa principal, existirá una instrucción de "call" que llama a la etiqueta que está al inicio de la tabla, como se muestra en el listado de abajo.

Una vez que el el contador del programa queda posicionado al inicio de la tabla, usamos la instrucción addwf, para sumar a la parte baja del contador del programa, PCL, un número del 1 al 4, que debe de estar almacenado en el registro W desde antes de la ejecución de la instrucción de "call". Al ejecutarse dicha instrucción (addwf), el contador del programa queda posicionado en la localidad deseada de la tabla.

Ahora, mediante la instrucción "retlw" (return from subroutine with literal in w), regresamos al programa principal, pero el registro W, contiene ahora el dato que aparece a la derecha de la instrucción (alguno de los códigos ASCII de la H,O,L ó A), y los cuales forman parte de la tabla.

Es decir que el registro W contiene, antes de la instrucción de "call", el desplazamiento (offset) deseado sobre la dirección de inicio de la tabla, (que en el ejemplo, puede ser un valor del 1 a 4). Aquí es muy importante señalar que en la instrucción (addwf PCL), se está afectando la parte baja PCL (8 bits) y también la alta (3 bits) del contador del programa, la cual se carga con el contenido de PCLATH. Es necesario entonces inicializar también el registro PCLATH para que contenga el valor de la página en donde se encuentra la tabla, que pueden ser los valores 0 a 7.

```
...main...
movlw   1               ;en este ejemplo, la tabla se encuentra en la página 1.
movw,f  PCLATH          ;carga el número de página en PCLATH
.............           ;en esta zona del programa, debe de inicializarse w con el desplazamiento.
```

```
        movlw   H'1'            ;en este ejemplo w=1
        call    TABLA
        .............           ;en ésta línea regresa la subrutina TABLA con el dato de la tabla en w.
        .............
        org     H'100'          ;origen de la tabla en la página 1. (100H ... 1FFH)
TABLA   addwf   PCL             ;ésta instrucción suma a PCL el contenido de w, y carga PCLATH en la parte al-
                                ;ta del contador del programa.
        retlw   'H'             ;regresa de la subrutina con el código ASCII de la "H" en w.
        retlw   'O'
        retlw   'L'
        retlw   'A'
```

17. <u>Puesta en marcha</u>:

PASO 1: REVISION DE COMPONENTES Y PRUEBA INICIAL DE LA TARJETA EVOLUPIC *Bootloader* 16F88:

Revise por favor que el kit incluya lo siguiente: tarjeta **EVOLUPIC *Bootloader* 16F88**, eliminador de baterías, cable serial, teclado hexadecimal, display LCD, el sensor de temperatura DS18B20 y disco CD de aplicación. Para poder usar el disco de aplicación, usted necesita una computadora con Windows XP o Vista con una unidad lectora de CD, y un puerto serial con conector DB9 ó bien una salida USB y un cable de interfaz serial USB-DB9.

S1 y S2 deben estar en la posición "RS232". Los 4 microswitches deben estar en la posición "OFF". Inserte el display LCD al conector de 14 pines. Posicione el selector de voltaje del eliminador de baterías a 6.0 ó 7.5 volts y conéctelo a la tarjeta **EVOLUPIC *Bootloader* 16F88.** Una vez alimentada la tarjeta, el programa cargado de fábrica en la memoria FLASH del 16F88, funciona en forma inmediata, mostrando un mensaje en el LCD si SW4=OFF ó mostrando la temperatura ambiente si SW4=ON.

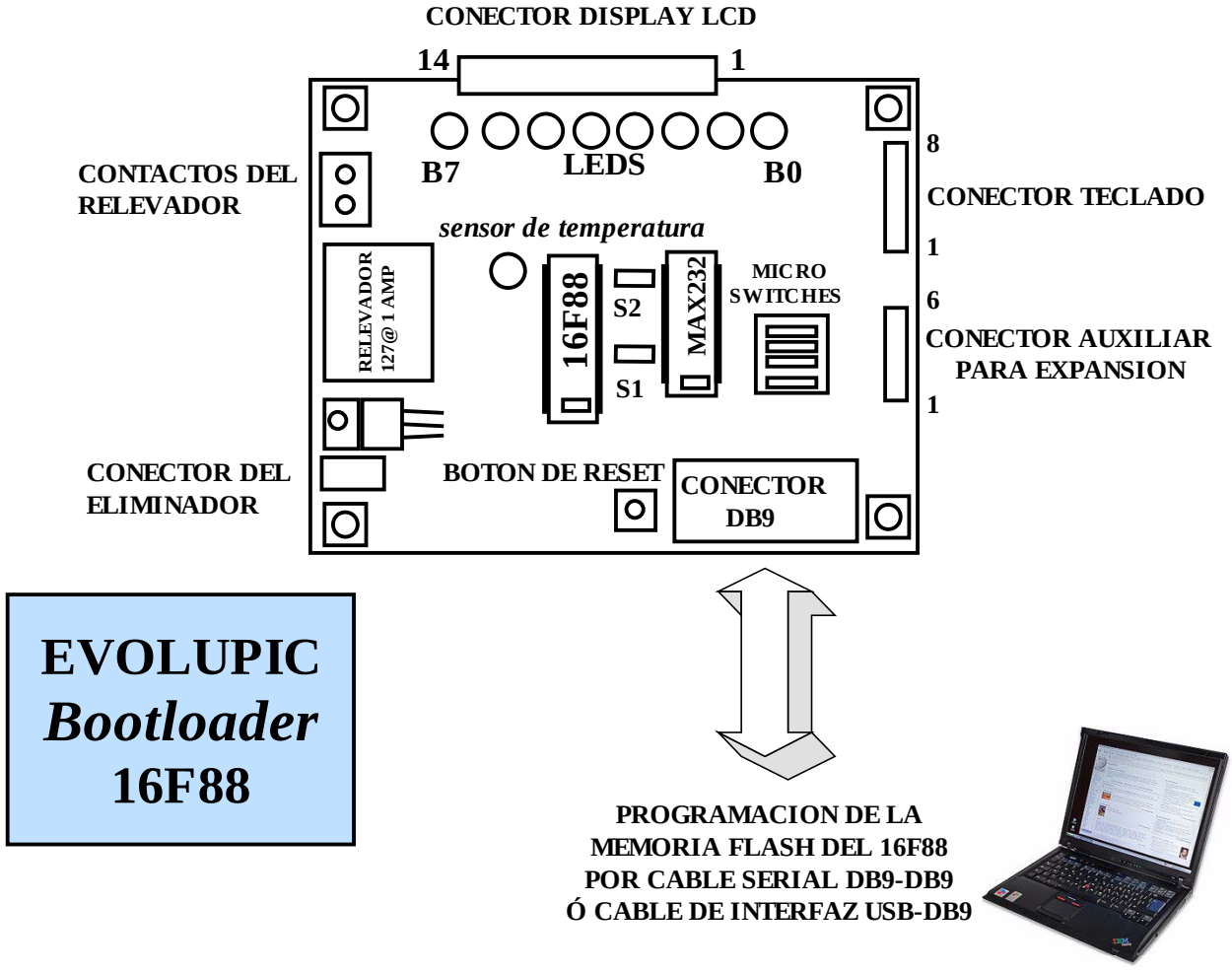


FIGURA 12

PASO 2: COPIA DEL DISCO CD, CONEXIÓN DE LA TARJETA A LA COMPUTADORA Y PROGRAMACIÓN DE LA MEMORIA FLASH DEL 16F88.

Inserte el disco de soporte en la unidad lectora de CD de la computadora. Abra el contenido del disco y copie la carpeta completa "EVOLUPIC Bootloader 16F88" hacia su computadora. Para la programación de la memoria FLASH del 16F88, por favor consulte el siguiente documento, incluido en el disco CD en la carpeta "MANUALES EVOLUPIC Bootloader 16F88".

"Guía General para la programación del sistema EVOLUPIC *Bootloader* 16F88 TINY.pdf"

PASO 3: INSTALACION DE MPLAB IDE:

En la carpeta "MPLAB IDE v 8.50", siga las instrucciones de la siguiente guía para la instalación de MPLAB IDE:

"Guía para la instalación de MPLAB IDE v8.50.pdf"

PASO 4: SI ES NECESARIO, QUITAR LA PROTECCION DE SOLO LECTURA A LOS ARCHIVOS.

En ocasiones –no siempre- es necesario quitar la protección de solo lectura de los archivos de prueba que se encuentran en el subdirectorio **"ARCHIVOS EVOLUPIC Bootloader 16F88 TINY".** Estos archivos traen a veces esa protección debido a que fueron copiados directamente del CD de solo lectura.

Abra el subdirectorio y quite la protección de solo lectura en los archivos mediante el siguiente procedimiento: seleccione "edit" y luego "seleccionar todo". Una vez que todos los elementos aparezcan sombreados, dé click en la parte derecha del mouse sobre cualquiera de los íconos y seleccione la opción "propiedades". Quite allí la selección de la opción "solo lectura", dejando el cuadrito en blanco. Luego dé click en "aplicar" y "cerrar". De esta forma todos los archivos quedarán ahora disponibles para su edición en MPLAB. Si omite este paso, MPLAB podría enviar mensaje de error al intentar editar ó ensamblar algún archivo

Los distintos tipos de archivo son: **Terminación .ASM** que son los archivos fuente. Estos archivos son de texto y están escritos en lenguaje ensamblador para el microcontrolador 16F88. Los archivos con **terminación .HEX** son los archivos ejecutables listos para su transferencia a la tarjeta EVOLUPIC Bootloader 16F88 a través del programa TINY.

PASO 5: CONFIGURAR MPLAB. EDITAR, ENSAMBLAR Y SIMULAR UN PROGRAMA EN LA PC:

MPLAB es un programa que integra numerosas funciones adicionales a las que se cubren en este manual. En esta sección se verá como cargar y ensamblar un programa.

Abra desde el escritorio el software MPLAB IDE. Para configurarlo, elija la opción "configure", "select device". Allí elija la opción "16F88". Ahora señale la opción "file", "open" y después elija el path:
C:\EVOLUPIC Bootloader 16F88\archivos EVOLUPIC Bootloader 16F88 TINY\MANEJO DE LEDS, MICROSWITCHES Y RELEVADOR TINY\demoF88TINY.asm

```
processor 16f88
include    <p16f88.inc>

BANCO0          MACRO
                BCF     STATUS,RP0
                BCF     STATUS,RP1
                ENDM
BANCO1          MACRO
                BSF     STATUS,RP0
                BCF     STATUS,RP1
                ENDM
J       equ     H'20'
K       equ     H'21'
                org 0
                clrf     PCLATH
                goto    WEB
                org     4
WEB             BANCO0
```

```
                bcf     RCSTA,SPEN      ;DESHABILITA UART
                BANCO1
                movlw   7
                movwf   CMCON
                clrf    CVRCON
                clrf    ANSEL           ;DESHABILITA CONVERTIDORES A/D
                clrf    TRISA
                clrf    TRISB
                BANCO0
inicio:         clrf    PORTB
                clrf    PORTA
                movlw   B'00000001'
                movwf   PORTB           ; port B =1
                bcf     STATUS,C        ;carry=0
mloop:          rlf     PORTB,f         ;rotate left PORT B (incluye carry)
                movlw   D'200'          ;retraso antes de seguir rotación
                movwf   J               ; J := w
jloop:          movwf   K               ; K := w
kloop:          decfsz  K,f             ; K = K-1, skip next if zero
                goto    kloop
                decfsz  J,f             ; J = J-1, skip next if zero
                goto    jloop
                goto    mloop

                end
```

La directiva **include <p16F88.inc>** contiene equivalencias entre los nombres y las direcciones de los registros y los bits usados por el 16F88.

Observe la línea en donde se encuentra la instrucción de carga al registro W que se encuentra con comentarios en ***letra inclinada***. Con el objeto de escalar la velocidad de simulación, que es muy lenta en comparación al tiempo real, modifique el valor "200" y ponga un "4". Ahora, elija la opción "project" y después "quickbuild". Esta opción realiza el ensamblado del programa y produce como salidas, entre otros, el archivo demof88TINY.hex..El archivo .HEX, contiene solamente el código ejecutable que habrá de almacenarse en el 16F88.

FORMATO DEL ARCHIVO DEMOF88TINY.HEX:

```
:020000040000FA
:040000008A01042845
:0800080083120313981383160 1
:10001000031307309C009D019B018501860183121B
:10002000031386018501013086000310860DCA3056
:0E003000A000A100A10B1A28A00B1928162869
:00000001FF
```

Es importante, antes de pasar a explicar el formato del archivo, aclarar que la longitud de palabra de las instrucciones en el 16F88 es de 14 bits, es decir, que cada localidad de memoria ocupa 2 bytes de almacenamiento en un archivo. Originalmente, el formato .HEX fue diseñado para computadoras con localidades de memoria de 8 bits de longitud, de tal forma que el número total de bytes en el archivo .HEX es el doble para el 16F88 que para otros microcontroladores de 8 bits.

El primer byte de cada línea, es un valor que corresponde al número de bytes (en hexadecimal) de información existentes en dicha línea. Por ejemplo, en la primera línea, hay un 02, lo cual significa que habrá 2 bytes de información. En la segunda línea hay un 04, lo que significa que habrá 4 bytes de información. En la cuarta un 10, ó 16 bytes de información en esa línea.

En seguida sigue la dirección de memoria inicial del bloque en donde habrán de almacenarse dichos bytes. En este ejemplo, vemos un 0000 en la primera línea, y un 0008 en la tercera que corresponde a la dirección número 4 del sistema 16F88 por lo que se explicó en el primer párrafo.

El siguiente byte es un 00 en todas las líneas, pero sin información útil. Los siguientes bytes en cada línea corresponden a la información que habrá de almacenarse en la memoria FLASH. Al final de cada línea está un byte check sum que sirve para

verificación, y corresponde al byte menos significativo de la suma binaria de todos los bytes anteriores en esa línea, obteniendo luego el complemento a 100H

Elija la opción "debugger" y "select tool" y luego "MPLAB SIM", de esta manera se cargará automáticamente la herramienta de simulación del procesador, con la cual usted podrá observar la ejecución del programa paso a paso. En cada instrucción usted puede revisar el contenido de registros y memoria mediante la opción "View", la cual le permite ver los registros o localidades de memoria seleccionados.

Dentro de la opción "View", elija "file registers" para que aparezca una ventana con los registros, e inicie la simulación oprimiendo F6. Avance la simulación oprimiendo repetidamente la tecla F7 y observe el contenido del registro 06, (que es el puerto B) que es en donde están conectados los LEDS. Con la tecla F6, avance paso a paso. El registro pasará del valor hexadecimal 01 al 02, 04, 08, 10, 20,40, 80 que es el patrón de corrimiento de los leds. Observe asimismo el contenido de las localidades 20H y 21H que se usan como contadores para los retardos.

ESCALAMIENTO DEL TIEMPO EN EL SIMULADOR CON RESPECTO AL TIEMPO REAL:

La velocidad de simulación es mucho menor a la velocidad del programa corriendo en tiempo real en la tarjeta EVOLUPIC *Bootloader*. Por esta razón es que en el programa cargado en MPLAB, en la subrutina de retraso debe de modificarse el valor decimal de 200 en el registro W, poniendo un 4 en su lugar. De lo contrario tomaría mucho tiempo hacer la simulación de una secuencia completa de corrimientos. Una vez concluída la simulación, este valor deberá reemplazarse de nuevo por un valor de 200 para cargar el programa en la tarjeta, y obtener un retraso de 200 ms aprox. entre cada corrimiento en tiempo real.

Para finalizar este punto, reemplace, como se explicó, el valor '4' por '200' y vuelva a ensamblar el programa mediante los comandos "project" y "quickbuild". Una vez ensamblado el programa sin errores, puede transferir el archivo ejecutable a la tarjeta EVOLUPIC *Bootloader* 16F88 mediante el software TINY.

18. Información técnica:

18.1 CARACTERISTICAS GENERALES:

Procesador: microcontrolador PIC16F88, cristal de 4 Mhz, con tecnología CMOS de bajo consumo.
Arquitectura: Harvard, con la memoria de código (14 bits) y de datos (8 bits) separadas. Procesamiento "pipeline".
Tecnología: RISC (reduced instruction set computer), con 35 instrucciones, con 14 bits de longitud de palabra.
Memoria: 4K localidades (14 bits) de FLASH, 368 localidades (8 bits) de RAM, 128 localidades (8 bits) de EEPROM.
Bootloader: firmware precargado de fábrica con capacidad de autoprogramación de la memoria FLASH.
Capacidad de lectura /escritura: hasta 10,000 ciclos en la memoria flash y hasta 10,000,000 en la EEPROM.
Puertos digitales: puerto A de 5 bits, puerto B de 8 bits, un total de 13 bits programables como entradas o como salidas.
Resistencias de pull up: disponibles en puerto B, cuando está programado como entradas.
Salidas digitales a LEDS : el puerto B con 8 bits conectados a LEDS.
Capacidad de salidas: cada bit de salida puede tomar ("sink"), ó generar ("source"), hasta 25 miliamperes.
Entradas digitales a microswitches: un total de 4, en el puerto A.
Salida a relevador: relevador modelo RAS-0910, contactos de 127 V @ 1 A. Salida a conector de tornillos.
Header auxiliar: de 6 contactos, señales RA1...RA4 disponibles, tierra y 5 v. Puede extenderse junto con el conector del teclado para hacer un solo header de expansión de 14 señales.
Conector para teclado: 8 señales RB0..RB7 incluídas en el header 8x, para teclado matricial de 4 x 4.
Conector para LCD: de 14 contactos, estándar y listo para la conexión de un LCD u otras interfaces.
Puertos seriales: USART compatible RS232, con su propio puerto DB9. SSP Puerto serial síncrono.
Convertidores AD: 7 canales, 10 bits de resolución.
Funciones adicionales: power-on reset, power up timer, watch dog, code protection , sleep (bajo consumo).
Temporizadores/contador de eventos: 3 temporizadores. Un generador de PWM
Interrupciones: fuentes de interrupción: externa del pin RBO/INT, overflow del temporizador, cambio en los niveles de las entradas RB4...RB7. Por escritura completa en la EEPROM. Del USART.
Dimensiones: 9.5 cms x 7.9 cms, tarjeta de fibra de vidrio, thru hole.
Consumo: normal < 2 ma @ 5v. y 4 Mhz., en SLEEP mode solo 15 uA.
Fuente de poder: eliminador de baterías de 300 ma. @ 6 v. ó 7.5 v.
Programador del 16F88: integrado en la tarjeta, la programación se realiza desde una computadora PC por puerto serial, empleando el firmware ***Bootloader.***
Software para desarrollo: MPLAB, que incluye editor, ensamblador, simulador y compilador. Software "TINY", programador de la memoria flash del 16F88.

18.2. LAY OUT

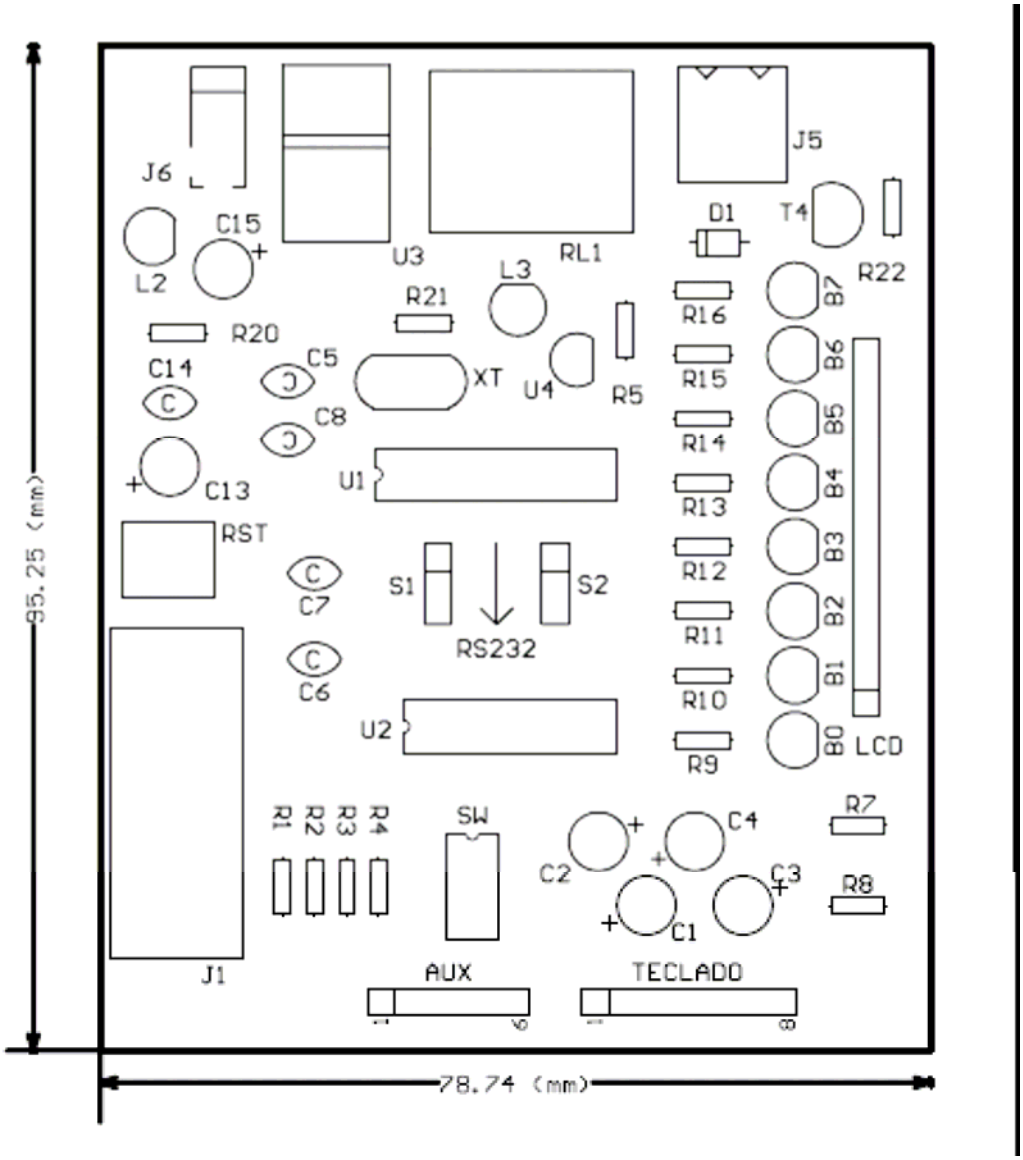
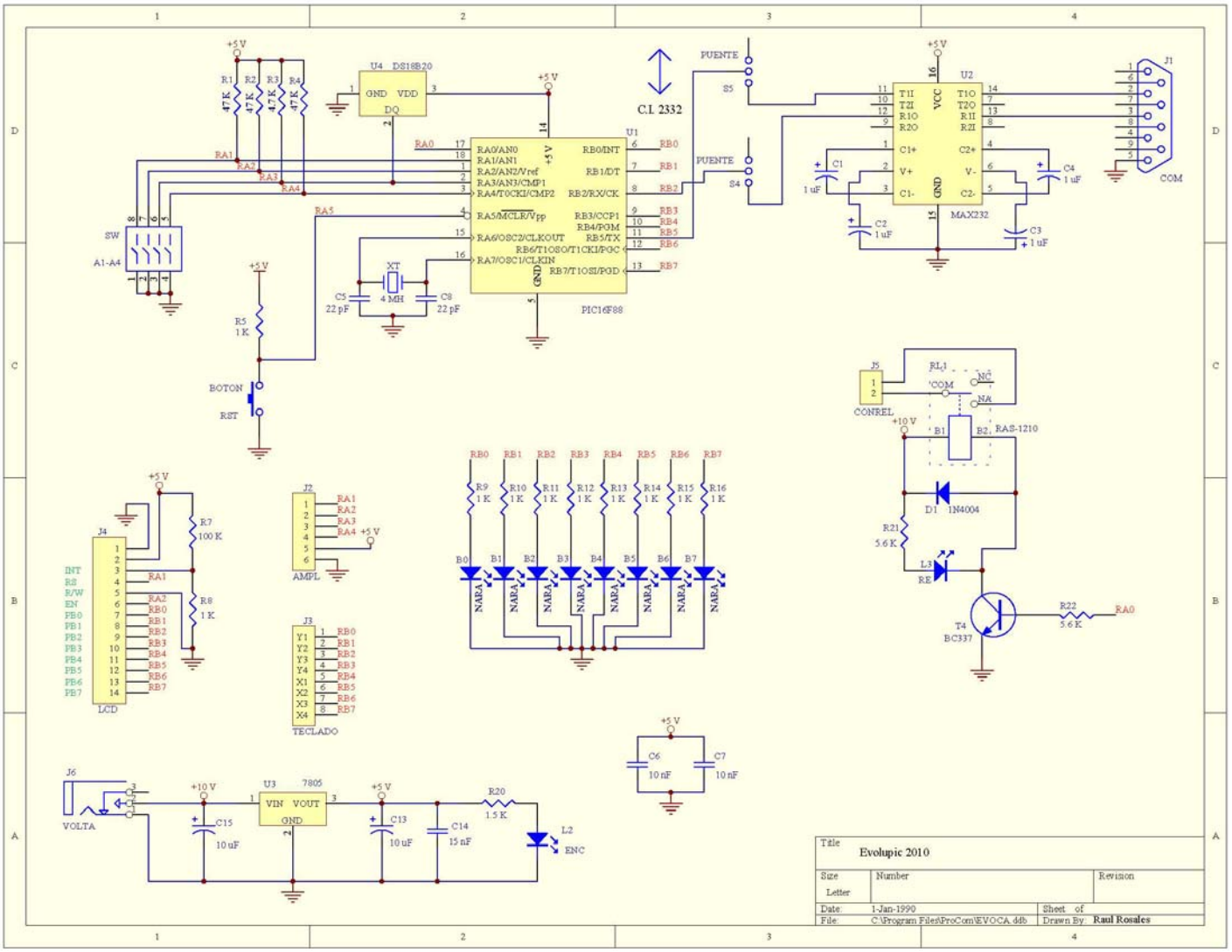


FIGURA 13

18.3 DIAGRAMA ELECTRONICO:



18.4 LISTA DE COMPONENTES

MATERIAL PARA EVOLUPIC Bootloader 16F88

	Referencia	Descripcion	Tipo	Huella	Cant.
1	B0...B7, L2,L3	leds rojos 5 mm.	NARA	LUZ	10
2	C13,C15	Capacitor Elec.	10 uF	CELCH	2
3	C6,C7,C14	Capacitor Cer.	10 nF	CAPI	3
4	C1...C4	Capacitor Elec.	1 uf		4
5	C5,C8	Capacitores Cer.	22 pF	CAPI	2
6	D1	Diodos	1N4148	DINU	1
7	J1	Conector DB9 para impreso	500-020	DB9H	1
8	J6	Conector para eliminador	ALIM	COCO	1
9	J5	Conector de tornillos 2X	TRT-02	TORA	1
10	J3	Conector ángulo 8X (teclado)	HEADER		1
11	J2	Conector ángulo 6X	HEADER		1
12	J4	Conector 14 pines	TIPO CAJA	CP14	1
13	R7	resistencia 1/2 w	100K	RES	1

14	R8...R16,R20,R5	resistencias 1/2 w	1 K	RES	11
15	R21,R22	resistencias 1/2 w	5.6K	RES	2
16	R1...R4	resistencias 1/2 w	47 K	RES	4
17	RL1	relevador	RAS-0910	REL	1
18	RST	botón de reset	AU-101	BOT	1
19	S1, S2	conector jumper 3 pines	PUENTE	CP3	2
20	puentes	jumpers	GMJ-2		2
21	SW	microswitches 4P	DIP-4P	BAS8	1
22	T4	Transistores NPN	BC337	TR92	1
23	U4	Sensor de temperatura	DS18B20		1
24	U2	Interfaz serial	MAX232		1
25	U1	Microcontrolador	PIC16F88	BAS18	1
26	base	para circuito integrado	16 patas		1
27	base	tubular para DS18B20	3 patas		1
28	base	para circuito integrado	18 patas		1
29	U3	Regulador de voltaje + 5 V	7805	REG	1
30	XT	Cristal miniatura	4 MHz	XTAL	1
31	tarjeta	circuito impreso	EVOLUPIC Bootloader		1
32	fuente	eliminador de baterias	300 ma.ELI-030		1
33	DISCO CD	disco CD y etiqueta			1
34	cable	cable serial macho-hembra	DB9 A DB9		1
35	empaque	caja de empaque			1
36	bolsa				1
37	ensamblado de tarjeta				1
38	Módulo LCD 16 x 1	1 renglón			1
39	teclado	hexadecimal			1

AVISO IMPORTANTE

EL 16F88 DEL SISTEMA EVOLUPIC *Bootloader* TIENE PRECARGADO DE FABRICA UN FIRMWARE BOOTLOADER EN LA ZONA ALTA DE SU MEMORIA FLASH, QUE ES EL QUE LE PERMITE AUTOPROGRAMAR EL RESTO DE LA MEMORIA FLASH A TRAVES DEL PUERTO SERIAL.

EL MICROCONTROLADOR 16F88 CUENTA CON UN MECANISMO DE SEGURIDAD PARA EVITAR ESCRITURAS ACCIDENTALES EN LA ZONA DE MEMORIA FLASH EN DONDE SE ENCUENTRA RESIDENTE EL FIRMWARE BOOTLOADER MENCIONADO. ESTE MECANISMO ES SIMILAR AL UTILIZADO PARA PROTEGER EL CONTENIDO DE SU MEMORIA EEPROM.

SIN EMBARGO, SI BIEN ES UN EVENTO DE MUY BAJA PROBABILIDAD, PUDIESE OCURRIR QUE EL SISTEMA REALICE UNA ESCRITURA ACCIDENTAL EN ESTA ZONA, LO QUE HARIA NECESARIO REPROGRAMAR EL 16F88 CON EL FIRMWARE BOOTLOADER ORIGINAL.

ES NECESARIO EN ESTE CASO USAR UN PROGRAMADOR ICSP, NO INCLUIDO CON EL KIT EVOLUPIC Bootloader 16F88. LAS INSTRUCCIONES PARA HACERLO SE ENCUENTRAN EN LA CARPETA "MANUALES EVOLUPIC Bootloader 16F88", EN EL ARCHIVO:

"GUIA GENERAL PARA LA PROGRAMACIÓN DEL SISTEMA EVOLUPIC *Bootloader* 16F88"