Prefacio

El presente documento forma parte del programa de estudios de la materia Microcontroladores II ofrecida a sus estudiantes en el Instituto Universitario de Tecnología para la informática – Iutepi. Sirve de apoyo complementario bajo la modalidad de autoaprendizaje publicado en su campus virtual, a todos los alumnos que cursan la materia.

Introducción

Desde la invención del circuito integrado, el desarrollo constante de la electrónica digital ha dado lugar a dispositivos cada vez más complejos. Entre ellos los microprocesadores y los microcontroladores, los cuales son básicos en las carreras de ingeniería electrónica. Un microcontrolador es un computador completo, aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado y se destina a gobernar una sola tarea. El número de productos que funcionan en base a uno o varios microcontroladores aumenta de forma exponencial. La industria Informática acapara gran parte de los microcontroladores que se fabrican. Casi todos los periféricos del computador, desde el ratón o el teclado hasta la impresora, son regulados por el programa de un microcontrolador. Los electrodomésticos de línea blanca (lavadoras, hornos, lavavajillas, etc.) y de línea marrón (televisores, vídeos, aparatos musicales, etc.) incorporan numerosos microcontroladores. Igualmente, los sistemas de supervisión, vigilancia y alarma en los edificios utilizan estos chips. También se emplean para optimizar el rendimiento de ascensores, calefacción, aire acondicionado, alarmas de incendio, robo, etc. Las aplicaciones de los microcontroladores son vastas, se puede decir que solo están limitadas por la imaginación del usuario. Es común encontrar microcontroladores en campos como la robótica y el automatismo, en la industria del entretenimiento, en las telecomunicaciones, en la instrumentación, en la industria automotriz, etc. Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado frecuentemente. Hace tres décadas, los controladores electrónicos se construían exclusivamente con componentes de lógica discreta, posteriormente se emplearon los microprocesadores, que se rodeaban con chips de memoria y E/S sobre una tarjeta de circuito impreso. En la actualidad, todos los elementos del controlador se han podido incluir en un solo circuito integrado, el cual recibe el nombre de microcontrolador. Realmente consiste en un sencillo pero completo ordenador contenido en un circuito integrado.

Contenido del programa de estudios

- 1 Microcontroladores PIC de la Gama Media.
- 1.1 Programación.
- 1.2 Herramientas de Trabajo: Editor, Simulador, Compilador, Programador.
- 1.3 Set de Instrucciones en BASIC para PICs.
- 1.4 Uso del Entorno de Programación en BASIC con PROTON.
- 1.5Uso del Simulador PROTEUS Aplicado al PICs de la Gama Media.
- 2 El Cargador de Programas del PIC.
- 3 Manejo de Interruptores y Leds.
- 3.1 Displays.
- 3.2 Potencia (Triacs. Relés. Optoacopladores).
- 4 Módulos:
- 4.1 Conversor A/D.
- 4.2 De Interrupciones.
- 4.3 Otros Módulos.

Microcontroladores PIC de la Gama Medio.

Las gamas de los PIC

La forma de designación de los PIC en general obedece a la siguiente estructura:

PIC nn LLL xxx

Siendo:

nn — un número propio de la gama del PIC.

LLL — código de letras donde la primera indica la tensión de alimentación y las otras dos el tipo de memoria que utiliza. En la tabla 1.1 se puede ver las distintas opciones que se pueden dar.

TABLA 1.1 Nomenclatura de los PIC.

LETRAS	ALIMENTACIÓN	MEMORIA
C	Standard (4.5.6.0 V)	EPROM
CR	Standard (4.5-6.0 V)	ROM
F	Standard (4.5.6.0 V)	FLASH
LC	Extendida (2.5-6.0 V)	EPROM

Instituto Universitario de Tecnología para la Informática – Iutepi

LCR	Extendida (2.5-6.0 V)	ROM
LF	Extendida (2.045.0 V)	FLASH

xxx — número que indica el modelo.

Los PIC se clasifican en distintas gamas atendiendo a los recursos disponibles en cada uno de ellos. Las gamas son:

a) Gama Enana (PICl2Cxxx): La principal característica es que son muy pequeños. con encapsulados de 8 pinas, y un juego de 33 instrucciones de 12 bits.
b) Gama Baja (PICI6C5xx): Los encapsulados son de 18, 20 6 28 pinos. Al igual que en los anteriores el número de instrucciones es de 33 con un ancho de 12 bits. La memoria de programa es de 512 palabras. I K 6 2K. y la de datos está comprendida entre 25 y 73 bytes. No permite interrupciones.
c) Gama Media (PIC16Cxxx): Es la gama más variada y completa de los PIC. con encapsulados desde 18 a 68 pinas. Tiene un conjunto de 35 instrucciones de 14 bits de ancho. Permite además características importantes que no soportaban los anteriores como son: - Interrupciones - Pila de 8 niveles que permite anidamiento de subrutinas. Esta familia a su vez se puede dividir en sub familias en función de los recursos de que se dispongan.
d) Gama Alta (PIC17Cxxx): Tienen unas características muy diferentes a las anteriores, ya que son microcontroladores de arquitectura abierta, es decir, que sacan sus buses al exterior.

El número de instrucciones es de 58 con una anchura de 16 bits. Tienen instrucciones vectorizadas.

e) Gama Mejorada (PICISCxxx): Es la última gama que ha salido. Tiene memoria de programa de hasta IM palabras. La mayoría de las instrucciones son de 16 bits, aunque las hay también de 32 bits. El número total de instrucciones es de 76. Esta gama está diseñada para aplicaciones de control. Todos ellos tienen convertidores A/D y. por ejemplo. están en fase de desarrollo algunos modelos que tienen interface para el bus CAN.

Los PIC de la gama media.

A lo largo de este ebook sólo se hablará de los PIC de la Gama Media, siendo esta la más variada y completa. En la tabla 1.2 se muestran las características que definen a cada modelo de los PIC de la Gama Media. Aparecen sólo los más significantes. ya que la variedad es muy amplia.

TABLA 1.2 Principales características de los PIC de la Gama Media.

	Reloj	Memoria				Periféricos							Características						
		Memoria de programa															Cápsulas		
	Máxima frecuencia de trabajo	EPROM	EEROM	Memoria de datos (bytes)	EEPROM de datos (bytes)	Módulos de temporización	Módulos CCP	Puertos serie	Puerto Paralelo	Convertidor A/D de 8 bits	Comparadores	Tensión interna de referencia	Fuentes de interrupción	Patillas de E/S	Rango de voltaje (voltios)	Detector de fallo en V_{DD} "Brown-out"	Pines DIP	Pines PLCC	Pines QFP
PIC16C61	20	1K	---	36	---	TMR0	--	---	--	--	--	--	3	13	3,0-6,0	---	18	18	--
PIC16C62	20	2K	---	128	---	TMR0,TMR1 ,TMR2	2	SPI/I²C/SCI	--	--	--	--	10	22	2,5-6,0	---	28	28	--
PIC16C63	20	4K	---	192	---	TMR0,TMR1 ,TMR2	2	SPI/I²C/SCI	---	--	--	--	10	22	3,0-6,0	---	28	28	--
PIC16C64	20	2K	---	128	---	TMR0,TMR1 ,TMR2	1	SPI/I²C	Si	--	--	--	8	33	3,0-6,0	---	40	44	44
PIC16C65	20	4K	---	192	---	TMR0,TMR1 ,TMR2	2	SPI/I²C/SCI	Si	--	--	--	11	33	3,0-6,0	---	40	44	44
PIC16C620	20	512	---	80	---	TMR0	--	----	--	--	2	Si	4	13	3,0-6,0	Si	18	18	20
PIC16C621	20	1K	---	80	---	TMR0	--	----	--	--	2	Si	4	13	3,0-6,0	Si	18	18	20
PIC16C622	20	2K	---	128	---	TMR0	--	----	--	--	2	Si	4	13	3,0-6,0	Si	18	18	20
PIC16C71	20	1K	---	36	---	TMR0	--	----	--	4ch	--	--	4	13	3,0-6,0	---	18	18	--
PIC16C73	20	4K	---	192	---	TMR0,TMR1 ,TMR2	2	SPI/I²C/SCI	--	5ch	--	--	11	22	3,0-6,0	---	28	28	--
PIC16C74	20	4K	---	192	---	TMR0,TMR1 ,TMR2	2	SPI/I²C/SCI	Si	8ch	--	--	12	33	3,0-6,0	---	40	44	44
PIC16C76	20	8K	---	368	---	TMR0,TMR1 ,TMR2	2	SPI/I²C/SCI	--	5ch	--	--	11	22	2,5-6,0	Si	28	28	--
PIC16C84	20	--	1K	36	64	TMR0	--	----	--	--	--	--	4	13	3,0-6,0	---	18	18	--

A continuación en la figura 1.1 se muestra algunos de los encapsulados utilizados en los PIC de la Gama Media. Como ya se ha dicho anteriormente, éstos van desde los 18 a 68 pines.

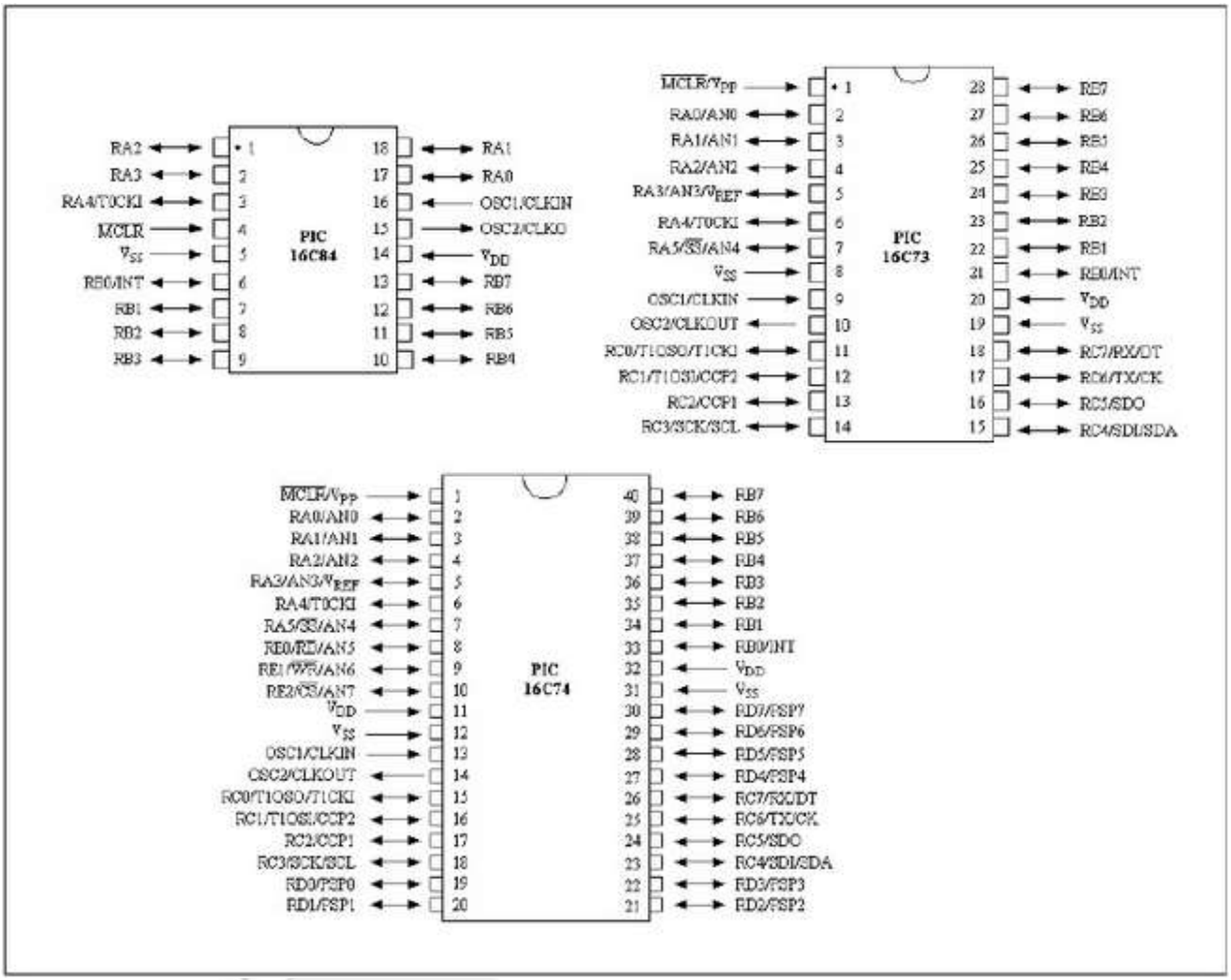


Figura 1.1 Principales diagramas de planilla.

La descripción de las funciones de los diferentes pines del microcontrolador son:

RA0/AN0-RA2/AN2: Líneas de E/S digitales del Puerto A, o entradas analógicas.

RA3/AN3/VREF: E/S digital, analógica o entrada externa de VREF.

RA4/T0CKI: E/S digital o entrada de reloj externo para TMRO.

RA5/AN4/SS : E/S digital, analógica o selección del puerto síncrono.

RB0/INT-RB7: E/S digitales del Puerto B. RBO/INT puede actuar como entrada de interrupción externa. RB4-Rb7 pueden provocar una interrupción cuando cambian de estado.

RC0/T1OSO/T1CKI: E/S digital del Puerto C. Conexión del oscilador externo para el temporizador TMRI o entrada de reloj para el TMRI.

RC1/T1OSI/CCP2: E/S digital. Conexión del oscilador externo para el TMR1 o salida del módulo 2 de captura/comparación.

RC2/CCP1: E/S digital. Salida del módulo 1 de captura/comparación.

RC3/SCK/SCL: E/S digital. E/S de reloj para el Puerto Serie Síncrono (SSP) en los módulos SPI o I^2 C.

RC4/SDI/SDA: E/S digital. Entrada de datos serie en el modo SPI. E/S de datos serie en el modo I^2 C.

RC5/SDO: E/S digital. Salida de datos serie en el modo SPI.

RC6/TX/CK: E/S digital. Transmisión serie asíncrona. Entrada de reloj para comunicación serie síncrona.

RC7/RX/DT: E/S digital. Recepción serie asíncrona. Línea de datos en la comunicación serie síncrona.

RDO/PSP0-RD7/PSP7: E/S digitales del Puerto D. Este puerto puede trabajar como puerto paralelo esclavo para interconexión con un bus de datos de 8 bits de otro microprocesador.

RE0/RD /AN5: E/S digital del Puerto E. Señal de lectura del puerto paralelo esclavo. Entrada analógica.

REU WR /AN6: E/S digital. Señal de escritura del puerto paralelo esclavo. Entrada analógica.

RE2/CS /AN7: E/S digital. Señal de activación del puerto paralelo esclavo. Entrada analógica.

La Arquitectura de los PIC de la Gama Media.

Desde el punto de vista de la arquitectura, la característica más importante de los PIC es que utilizan la Harvard, frente a la Van Neumann que es la habitual.

La arquitectura Harvard tiene la memoria de programa y la memoria de datos separadas y se accede a ellas mediante bases distintos. Esto mejora el ancho de banda sobre la tradicional arquitectura secuencia], en la cual los programas y datos son buscados en la misma memoria, utilizando el mismo bus. En la arquitectura Harvard mientras se accede a la memoria de programa, sobre la memoria de datos se puede estar leyendo o escribiendo, lo que permite ejecutar una instrucción a la vez que se busca la siguiente.

En la figura 1.2 se muestra el esquema de la arquitectura general de los PIC de la Gama Media.

Además de las características antes mencionadas, otras características que poseen los PIC de la Gama Media son:

a) Memoria no volátil de programa de hasta 8K x 14 de tamaño, direccionada por el contador de programa (PC) de 13 bits.

Instituto Universitario de Tecnología para la Informática – Iutepi

b) Memoria RAM de datos de hasta 368 x 8 de tamaño, direccionada por el código de operación o por el registro FSR y parte del registro STATUS.

c) Unidad aritmético-lógica de 8 bits con un registro acumulador de trabajo asociado también de 8 bits. Realiza operaciones aritméticas y lógicas utilizando siempre como operando el registro acumulador y otro dato perteneciente a cualquier registro. Realiza operaciones de suma, resta y desplazamiento. Opera en complemento a 2 (C'2). Dependiendo del resultado, afecta a algunos bits del registro de estado.

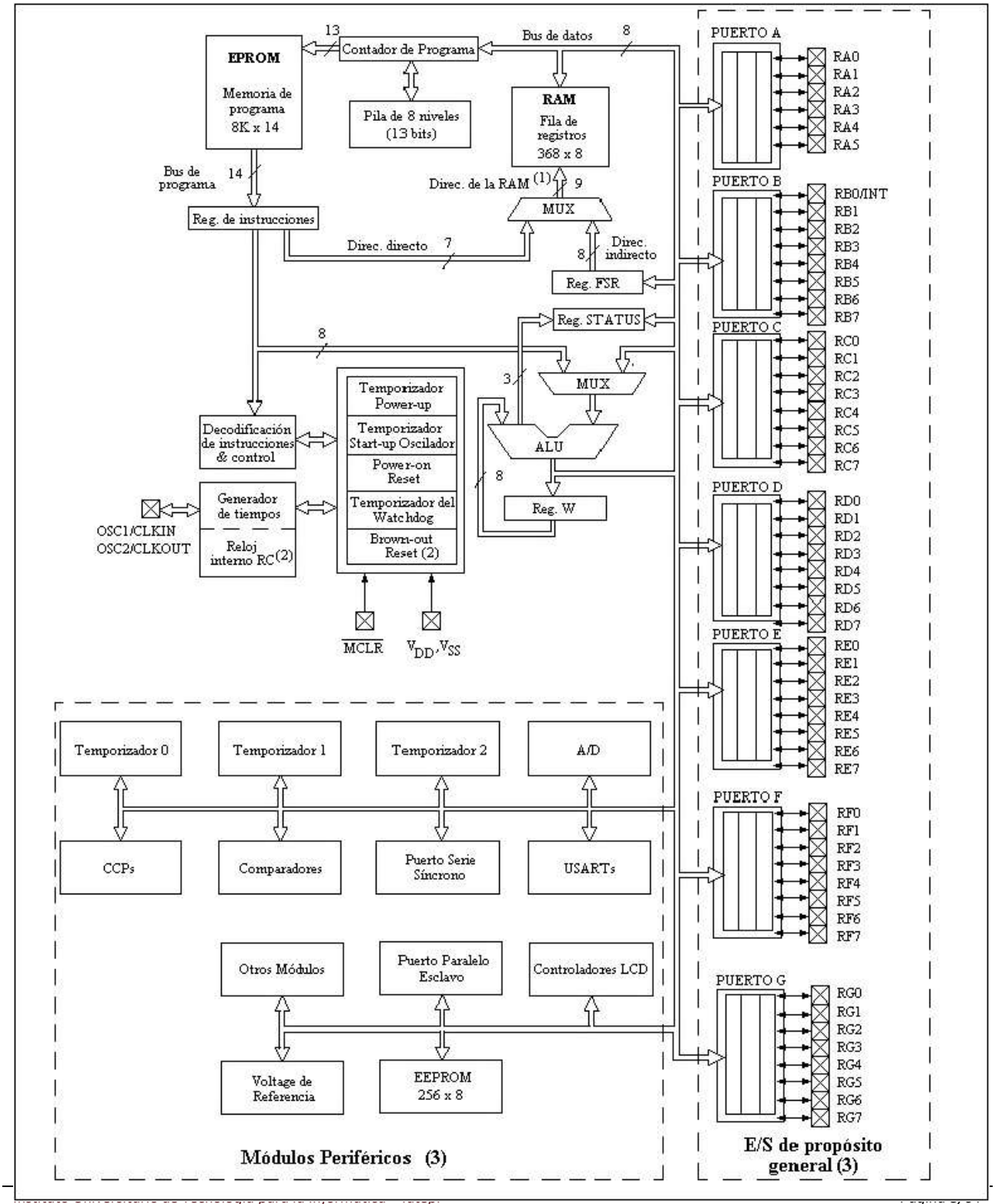


Figura 1.2 Diagrama de bloques general de los PIC de la gama media.

Nota 1: Los bits de mayor orden en el direccionamiento directo de la RAM se obtienen del Registro de Estado.

2: No todos los dispositivos tienen estas características, habrá que consultar las hojas de características.

3: Muchos de los pines de las puertas E/S de propósito general están multiplexadas con una o más funciones de los módulos periféricos. La combinación de las funciones multiplexadas dependen del dispositivo.

d) Oscilador encargado de las generaciones de tiempo del sistema.

e) Circuitos controladores del RESET.

f) Puertos E/S. Existen hasta 7 puertos de E/S, aunque no existe ningún dispositivo que los implemente todos. Además se multiplexan con distintos elementos del sistema.

g) Módulos periféricos. Son la característica que diferencia a los distintos microcontroladores. Éstos facilitan la comunicación con el mundo exterior, tal como las E/S de propósito general, y tareas internas tal como puede ser la generación de las distintas bases de tiempo. Los periféricos sobre los que se está hablando son; E/S de propósito general, hasta tres temporizadores, módulo de captura, comparación, y PWM (CCP), puerto serie síncrono (SSP), USART (SCI), módulo para generación de voltajes de referencia, módulos comparadores analógicos, conversores analógico digital (A/D), módulo para control de Display de cristal líquido (LCD) y puerto paralelo esclavo (PSP).

h) Registros generales del sistema.

Organización de la memoria.

La memoria de los PIC está dividida en dos bloques; la memoria de programa y la memoria de datos. Cada uno de los bloques tiene su propio bus, pudiendo tener acceso a ambos bloques de memoria en el mismo ciclo de reloj.

Organización de la memoria de programa.

En los PIC de la gama media el contador de programa es de 13 bits, con lo que se puede direccionar una capacidad de memoria de 8K x 14 bits. El ancho de la memoria de programa va a ser la longitud de una instrucción que es de 14 bits, por lo se podrán almacenar hasta 8K instrucciones. De esta forma es fácil determinar si el dispositivo tiene suficiente memoria de programa para una aplicación deseada.

Esta memoria de programa a su vez está dividida en cuatro páginas de 2K palabras cada una (Oh — 7FFh, 800h — FFFh, 1000h — 17FFh, y 1800h — 1FFFh). En la figura 1.3 se muestra el mapa de memoria con la pila de 8 niveles. No todos los dispositivos tienen implementados estos cuatro bancos.

Instituto Universitario de Tecnología para la Informática – Iutepi

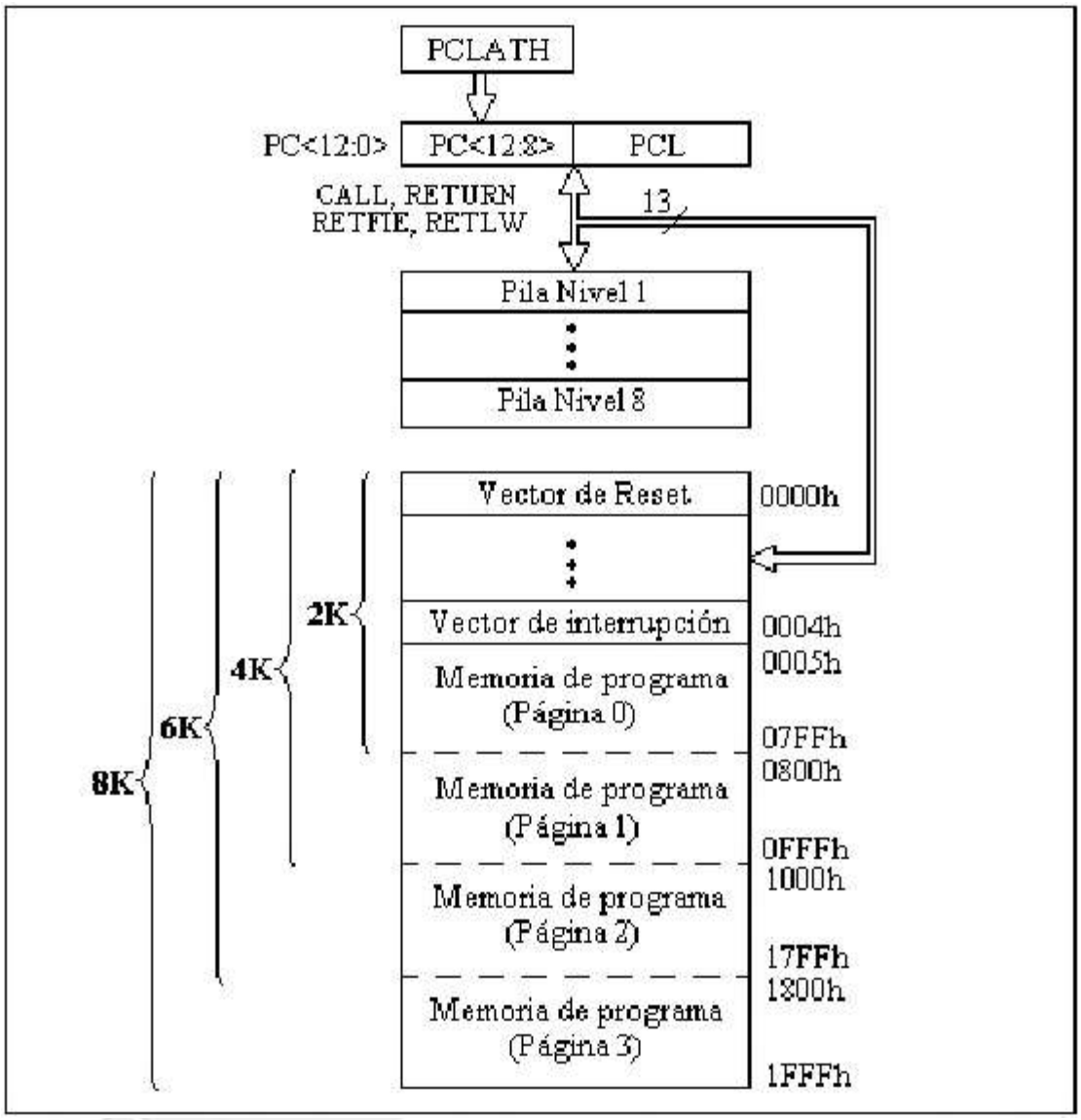


Figura 1.3. Mapa de memoria.

Para realizar los saltos entre las páginas de la memoria de programa se utilizan los dos bits más altos del contador de programa (PC <11:12>), los cuales son implementados físicamente en la memoria de datos en el registro PCLATCH <4:3>. Si el programa se ejecuta secuencialmente no es necesario modificarlo.

Dentro de la memoria de programa existen dos direcciones reservadas:

1. Vector de Reset .- En un dispositivo, un reset pone el contador de programa (PC) a cero. A esta dirección se la denomina "Dirección del Vector de Reset", donde está la dirección de inicio para la ejecución del programa.

Cualquier reses borrará también el contenido del registro PCLATH, de modo que cuando se produzca un resello dirección inicial ha de estar localizada en la página O de la memoria de programa.

2. Vector de Interrupción .- Cuando una interrupción es reconocida el PC es forzado a la dirección 0004h. A esto es a lo que se le llama "Dirección del Vector de Interrupción". Cuando el PC es forzado con la dirección del vector de interrupción, el PCLATH no es modificado. Por eso mismo en la dirección del vector de interrupción, el registro PCLATH debería ser escrito con el valor que especificará la localización deseada en la memoria de programa. Antes de realizar esta operación, el PCLATH debería ser salvado para volverlo a restaurar al terminar el tratamiento de la rutina de interrupción.

El contador de programa (PC) es un registro que especifica la dirección de la instrucción que ha de ser ejecutada. El PC tiene un ancho de 13 bits, correspondiendo el byte más bajo con el registro de lectura-escritura PCL, implementado en la memoria de datos, y el byte más alto con el registro PCH. Este último contiene sólo los bits <12:8> del PC y no se puede leer ni escribir directamente, sino que son cargados a través del PCLATH.

Existen varias instrucciones que modifican el flujo normal del sistema, y por tanto afectan al PIC. Estas instrucciones son:

1. Instrucciones de salto relativo: Son aquellas instrucciones que usan como operando destino al PC. En estas instrucciones se cargan los 8 bits menos significativos desde la UAL. El resto de los bits que coinciden con los del PCLATH, habría que modificarlos manualmente.

2. Instrucción GOTO de salto incondicional: Carga desde el código de operación los 11 bits menos significativos. Los dos que quedan se cargan desde el PCLATH <4:3>. Si se hace un GOTO a una dirección que está en una página distinta, primero hay que modificar estos bits del PCLATH.

3. Instrucción CALL de salto a subrutina: Es igual que la anterior con la diferencia de que antes de modificar el PC, el valor que tuviese se lleva a la pila.

4. Instrucciones RETURN, RETLW y RETFIE de retornos de subrutinas y rutinas de tratamiento de interrupción: Cargan directamente en el PC el valor que se halla guardado anteriormente en la pila. No es necesario modificar el PCLATH.

Los PIC de la Gama Media poseen una pila de 8 niveles con un ancho de 13 bits, la cual nos permite guardar las direcciones de retorno a un programa, cuando en éste se produce un salto a una subrutina. Esto nos da capacidad para anidar hasta 8 subrutinas producidas por programa o mediante interrupciones. Si en la pila ya se han almacenado 8 valores, el nuevo valor se cargará sobre el primer nivel de la pila, de modo que puede dar problemas en el funcionamiento de un programa.

Organización de la memoria de datos.

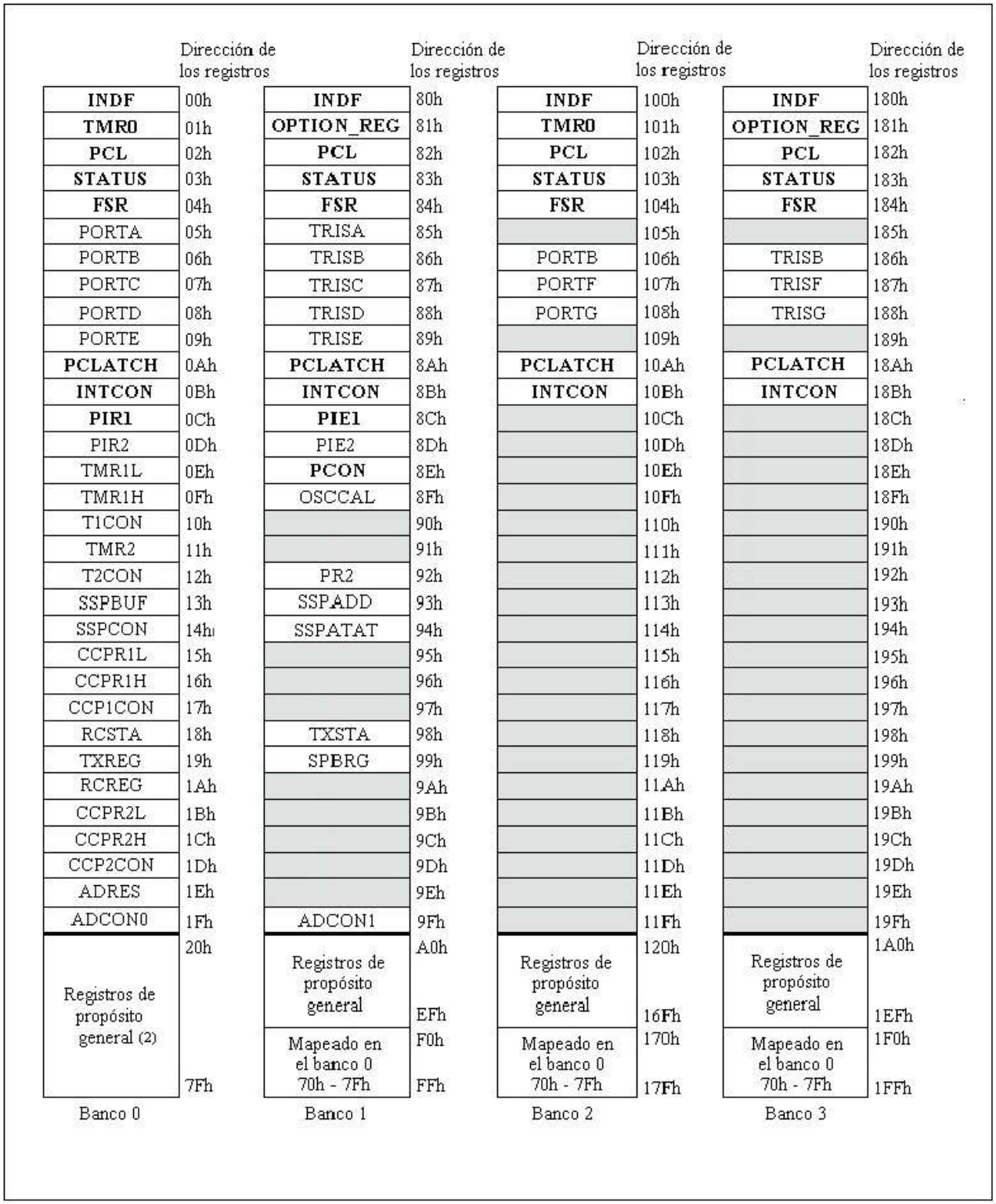
La memoria de datos (figura 1.4) está dividida en dos partes:

a) Memoria de registros especiales (FSR Memory): Está formada por aquellos registros que son usados por la CPU y los periféricos para controlar una operación.

Instituto Universitario de Tecnología para la Informática – Iutepi

b) Memoria de propósito general: Está formada por la memoria que puede usar el programador.

En ambos casos, está dividida hasta en 4 bancos de memoria de hasta 128 bytes, aunque muchos de los modelos no traen implementados estos dos últimos bancos. En algunos modelos de PIC la memoria de propósito general, está sólo implementado físicamente en el banco0, estando el resto de los bancos mapeados en éste.



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

Figura 1.4 Organización de la memoria de datos.

Nota 1: Los registros en negrita están presentes en todos los dispositivos.

2: Puede que no todas las posiciones estén implementadas. Las localizaciones no implementadas se leen como '0'.

El modo de acceso a cada uno de los bancos dependerá del tipo de direccionamiento que se use.

Estos tipos de direccionamiento son:

1. Direccionamiento directo: La dirección final se obtiene con:

a. Los bits RP I -RPO del registro de estado (STATUS), los cuales seleccionan el banco de memoria.

b. Los 7 bits del código de operación, los cuales indican la posición del dato dentro del banco.

2. Direccionamiento indirecto: Con este tipo de direccionamiento lo que se da es la dirección donde se encuentra la dirección del dato. Para ello se usa el registro INDF. Al usar este registro (que no está implementado físicamente), lo que se hace es acceder al registro de datos FSR de 8 bits. Los 7 bits con menos peso del FSR seleccionan la posición dentro del banco, y el bit de mayor peso junto con el bit IRP del registro de estado, seleccionan el banco.

Un programa que borra una zona de memoria puede ser un ejemplo de aplicación de direccionamiento indirecto. El programa sería como el siguiente:

```
            BFC     STATUS,IRP  ;Selección del banco O de memoria.
            MOVLW   0X20        ;Mueve el valor 20h al registro W
            MOVWF   FSR         ;Carga valor de W en FSR
SIGUIENTE   CLRF    INDF        ; Borra la posición de memoria que marca FSR
            INCF    FSR,1       ; Incrementa valor del FSR
            BTFSS   FSR,4       ;Test del bit 4 del FSR, si es 1, no ejecuta la
                                ; siguiente instrucción y salta.
            GOTO    SIGUIENTE   ; Vuelve a SIGUIENTE.
```

Este programa ejemplo borraría la memoria de datos desde la posición h'20' hasta h'2F'.

Registros y recursos comunes. Los Registros de Estado.

Es un registro formado por 8 bits, que contiene el estado de la UAL, del RESET y selecciona el banco de la memoria de datos sobre la que queremos trabajar.. Por esta última causa, el registro de estado se encuentra en todos los bancos, y en la misma posición.

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/w-x	R/W-x
IRP	RP1	RPO	TO	PD	Z	DC	C
bit7							bit0

A continuación se muestra la función de cada bit del registro de estado.

bit 7	IRP: Selección de bancos para el direccionamiento indirecto. 1 = Banco 2,3 (100h — 1FFh) 0 = Banco 0,1 (00h — FFh)
bit 6:5	RP1:RPO: Selección del banco de la memoria de datos para el direccionamiento directo. 11 = Banco 3 (180h — 1FFh) 10 = Banco 2 (100h — 17Fh) 01 = Banco 1 (80h — FFh) 00 = Banco O (00h — 7Fh)
bit 4	TO: Timer Out. 1 = Tras conectar Vdd o ejecutar "CLRWDT" o "SLEEP". 0= Al rebasar el WDT
bit 3	PD: Power Down 1 = Tras conectar Vdd o ejecutar "CLRWDT". 0 = Al ejecutar la instrucción "SLEEP".
bit 2	Z: Bit de cero. 1 = El resultado de una operación es 0. 0 = El resultado es distinto de 0.
bit 1	DC: Acarreo en el 4° bit de menos peso. 1 = Acarreo en la suma y no en la resta. 0= Acarreo en la suma y no en la resta.
bit 0	C: Acarreo en el 8° bit. 1 = Acarreo en la suma y no en la resta. 0= Acarreo en la suma y no en la resta.

Estos dos últimos bits, en la suma representan lo que se conoce comúnmente con acarreo, sin embargo en la resta realizan la función de bit de signo (bit de Borrow), indicando que el

resultado es negativo si Borrow es O, y mayor o igual a O si es 1.

Registro de Opciones.

El registro de opciones es un registro de lectura-escritura, que contiene los bits de configuración del divisor de frecuencia del TMRO/WDT, de las interrupciones externas, del TMRO, y la configuración de las Puerta B con cargas Pull-Up.

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	TOCS	T0SE	PSA	1 PS2	PS1	PSO
bit 7							Bit0

Instituto Universitario de Tecnología para la Informática – Iutepi

bit 7 RBPU : Conexión de cargas Pull-Up para la Puerta B.
1 = Todas las cargas Pull-Up desconectadas.
0 = Todas las cargas Pull-Up conectadas.
bit 6 INTEDG: Tipo de flanco para la interrupción.
1 = RBO/INT sensible a flanco ascendente.
0 = RBO/INT sensible a flanco descendente.
bit 5 T0CS: Fuente de reloj para el TMRO.
1 = Pulsos introducidos por TOCKI (contador).
0 = Pulsos de reloj interno Fosc/4 (temporizador).
bit 4 T0SE: Tipo de flanco activo del TOCKI.
1 = Incremento del TMRO cada flanco descendente.
0 = Incremento del TMRO cada flanco ascendente.
bit 3 PSA: Asignación del divisor de frecuencia.
1 = Se le asigna al WDT.
0= Se le asigna al TMRO.
bit 2-0 PS2:PSO: Valor del divisor de frecuencia.

Valor	División del TMRO	División del 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	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Para conseguir asignar 1:1 al divisor de frecuencia en el TMRO, se le asigna el divisor al WDT.

Palabra de configuración.

La Palabra de Configuración está formada por 14 bits. Sr utiliza para la selección de distintos aspectos de la configuración del dispositivo según las necesidades de la aplicación. Su localización en la memoria de programa es la 2007h. Esta posición no es accesible durante el modo de funcionamiento normal, por lo que estos registros deberán ser configurados en la fase de programación.

CP1	CP0	CP1	CP0	CP1	CP0	-	BODEN	Cp1	CP0	PWRTE	WDTE	FOSC1	FOSCO
bit13													bit0

bit 13-8 CP<1:0>: Bits de protección de código.
5-4:
bit 7 No implementado: Su lectura es 1.
bit 6 BODEN: Detección del “Brown-Out” (Fallo de alimentación)
1 = detección activada.
0 = Detección desactivada.

Instituto Universitario de Tecnología para la Informática - Iutepi

bit 3 PWRTE : Activación del temporizador "Power-Up"
1 = Desactivado.
0 = Activado.
bit 2 WDTE: Activación del "Watchdog" (perro guardián)
1 = Activado
0 = Desactivado.
bit 1-0 FOSC1-FOSC0: Selección del tipo de oscilador.
11 = Oscilador RC.
10 = Oscilador HS
01 = Oscilador XT
00 = Oscilador LP

Oscilador.

El circuito de oscilación se utiliza para generar las señales de reloj en el sistema necesarias para la ejecución de instrucciones y para el funcionamiento de los periféricos.

Existen hasta ocho tipos de osciladores. Los distintos tipos permiten mayor flexibilidad del dispositivo según las necesidades, así los hay de muy bajo coste, como las redes RC, y otros de bajo consumo como son los cristales de cuarzo del modo LP.

La configuración de un oscilador u otro se realiza mediante los bits FOSC2, FOSC1 Y FOSC0 de la palabra de configuración. No todos los modelos poseen la posibilidad de configurar cualquier tipo de oscilador, en estos casos sólo tienen los bit FOSC1 Y FOSC0 en la palabra de configuración. Los valores de estos bits para la configuración de los distintos osciladores son los que se pueden ver en las tablas 1.3 y 1.4

TABLA 1.3 Selección del modo de oscilación con FOSC1:FOSC0.

Bits de configuración FOSC1:FOSC0	Modo OSC	Ganancia de los inversores internos	Comentarios

Instituto Universitario de Tecnología para la Informática – Iutepi

11	RC	-	Solución más barata (sólo requiere una resistencia y un condensador). Máxima variación del tiempo base. Modo del dispositivo por defecto
10	HS	Ganancia alta	Cristal de cuarzo para aplicaciones de alta frecuencia. Es el modo que más potencia consume de los tres cristales.
01	XT	Ganancia media	Cristal de cuarzo para un rango de frecuencias estándar.
00	LP	Ganancia baja	Cristal de cuarzo para aplicaciones de baja frecuencia. Es el modo que menos potencia consume de los tres cristales.
111	EXTRC con CLKOUT	-	Red RC externa sacando la señal de oscilación por la patilla CLKOUT. Solución barata. Máxima variación en el tiempo base. Modo del dispositivo por defecto.
110	EXTRC	-	Red RC externa. Barata solución. Máxima variación en el tiempo base. La señal de oscilación no sale al exterior.
101	INTRC con CLKOUT	-	Red RC interna sacando la señal de oscilación por la patilla CLKOUT. Es la solución más barata. Oscilador de 4MHz.
100	INTRC	-	Red RC interna. La señal de oscilación no sale al exterior. Solución más barata. Oscilador de 4MHz.
011	-	-	Reservado
10	HS	Ganancia alta	Cristal de cuarzo para aplicaciones de alta frecuencia. Es el modo que más potencia consume de los tres cristales.
01	XT	Ganancia media	Cristal de cuarzo para un rango de frecuencias estándar.
00	LP	Ganancia baja	Cristal de cuarzo para aplicaciones de baja frecuencia. Es el modo que menos potencia consume de los tres cristales.

La diferencia entre los tres últimos viene dada por la ganancia de los inversores internos, que son los que modifican la frecuencia. En general siempre se opta por la opción con menor ganancia posible que cumpla las especificaciones. Esto implicará menores corrientes y con ello menor consumo de potencia.

En la figura 1.5 se ve el modo de conexión para resonadores de cristal o de cuarzo. En este caso se conectan a las patillas OSC1 y OSC2, usando esta última como realimentación.

Instituto Universitario de Tecnología para la Informática – Iutepi

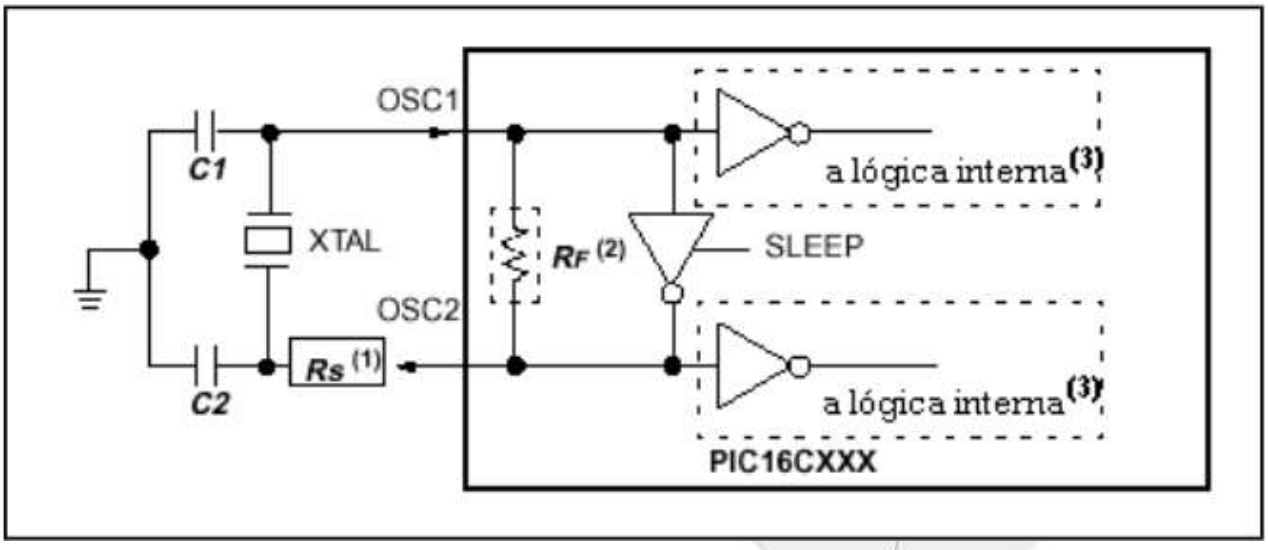


Figura 1.5 Operación de osciladores en los modos HS, XT o LP.

Nota 1: La resistencia en serie, RS, es necesaria para cristales de alta tecnología.

2: La resistencia de realimentación, RF, varía entre 2 y 10 MΩ.

3: Dependiendo del dispositivo, el buffer de la lógica interna puede estar tanto antes como después del inversor.

Para una red RC el modo de conexión utilizará únicamente la patilla OSC1. La frecuencia de oscilación dependerá de VDD, REXT y de CEXT (figura 1.6). El fabricante recomienda que REXTtenga un valor de entre 3KΩ y 100KΩ y que CEXT sea mayor de 20pF. Además la frecuencia del oscilador/4 se obtiene a través de la patilla OSC2, en caso de que esté configurada en los modos EXTRC o INTRC con CLKOUT, se puede usar para sincronización de otras lógicas.

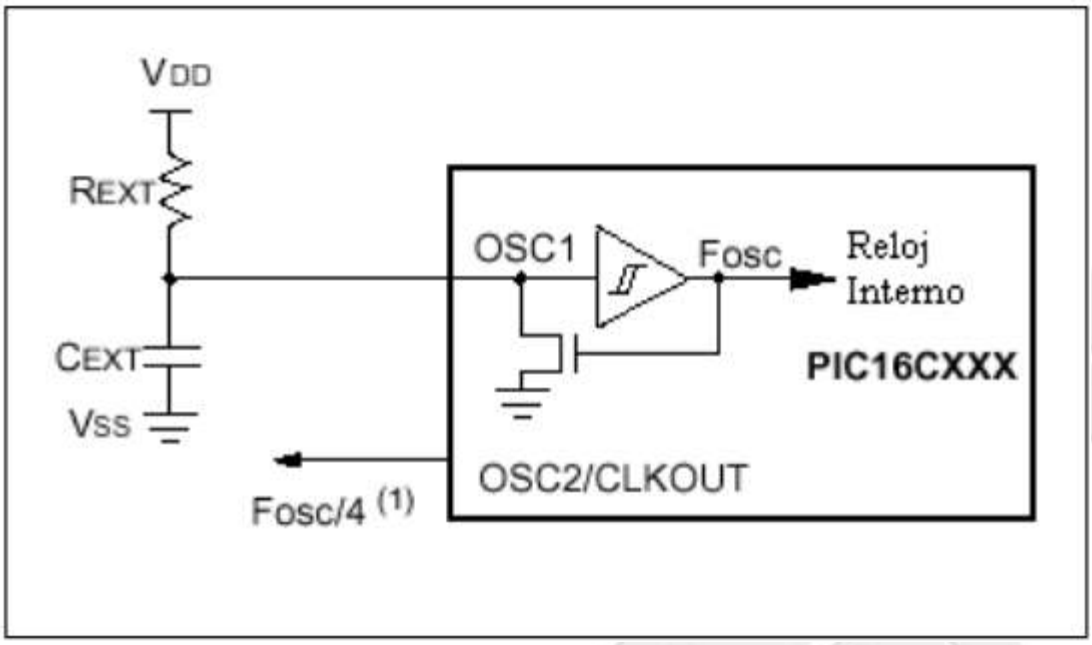


Figura 1.6 Operación en modo RC.

Nota 1: Esta patilla también puede ser configurada como una E/S de propósito general.

Como un dispositivo incrementa su voltaje de alimentación desde 0 hasta VDD, el oscilador tardará un tiempo en funcionar correctamente. Este tiempo de comienzo o Start-Up depende de muchos factores:

a) Para los osciladores RC depende del valor de la resistencia y condensador usado, tiempo de subida de VDD, y temperatura del sistema.

b) Para los osciladores de cristales los factores que intervienen son la frecuencia del cristal, los valores de la resistencia en serie, RS y de los condensadores C1 y C2, el tiempo de subida de VDD, la temperatura del sistema, la selección del modo de oscilador, la composición del circuito oscilador, y el ruido del sistema.

En la figura 1.7 se muestra un ejemplo de Start–Up. Se puede ver que la señal de oscilación está centrada en VDD/2, siendo el valor pico a pico durante el arranque del oscilador bastante bajo (menos del 50% de VDD).

Instituto Universitario de Tecnología para la Informática – Iutepi

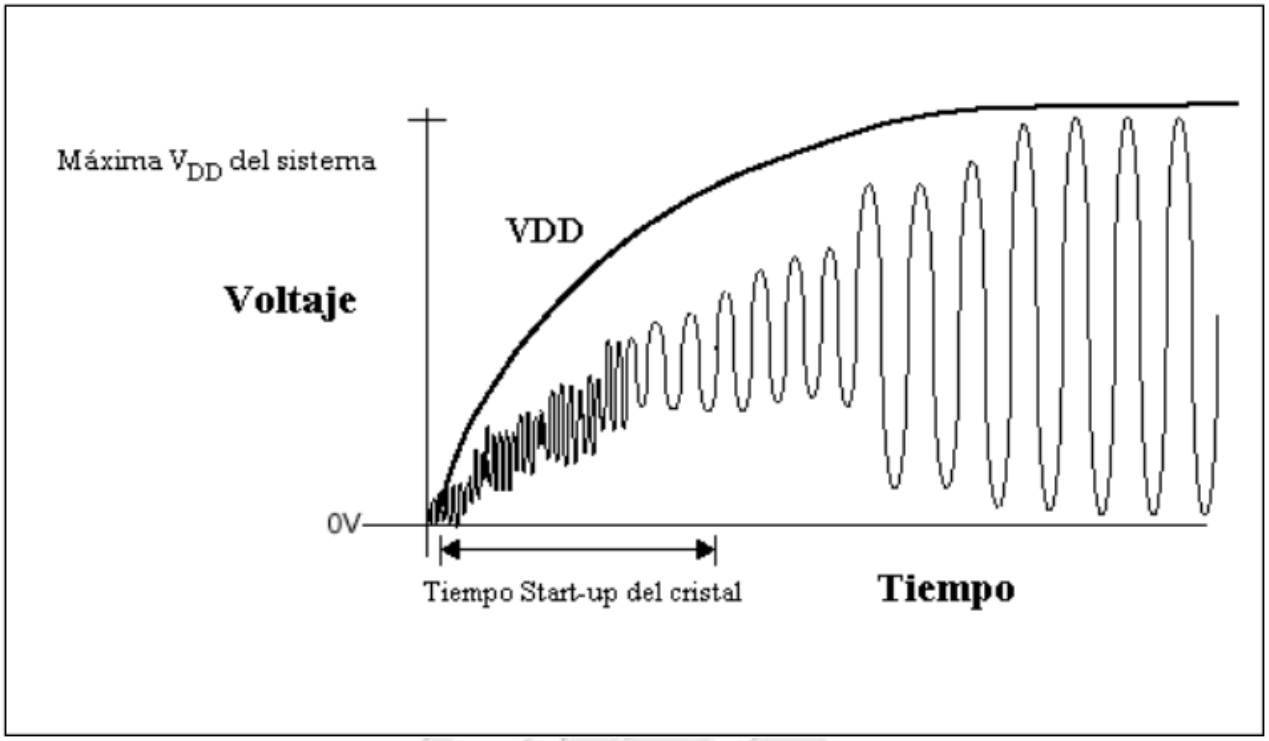


Figura 1.7 Característica Start-Up de un oscilador.

Siempre que se quiera grabar un PIC es necesario consultar las hojas de características para la configuración del oscilador.

Reset

La función del reset es la de dejar al sistema en un estado conocido. Existen varias fuentes de RESET:

a) Power On Reset (POR).

b) Activación de MCLR durante funcionamiento normal.

c) Activación de MCLR en el modo de reposo

d) Desbordamiento del WDT durante funcionamiento normal.

e) Reset de Brown-Out (BOR).

f) Reset por error de paridad (PER).

Instituto Universitario de Tecnología para la Informática – Iutepi

La figura 1.8 se muestra un bloque simplificado de la lógica que gobierna el reset del sistema, para un caso general.

En esta figura se puede ver una zona denominada OST/PWRT, que consiste en:

a) Power-up Timer (PWRT): Es un temporizador que proporciona un retardo de 72ms a partir de un reset de tipo POR o BOR, de tal forma que el PIC se mantiene reseteado durante este tiempo, así al finalizar la temporización la tensión de alimentación tendrá un valor aceptable. Este temporizador se habilita mediante el bit PWRT de la palabra de configuración. Para generar este retardo existe un circuito interno RC dedicado.

b) Oscillator Start-Up Timer (OST): Este oscilador proporciona un retardo de 1024 veces el periodo de oscilación (desde OSC1) después del retardo PWRT. De esta forma se asegura que el oscilador cristal o cerámico es estable cuando el PIC empieza a funcionar. Este retardo sólo funciona cuando el PIC se ha configurado para usar un oscilador XT, HS o LP y el reset es tipo POR, BOR o de wake-up desde el modo de reposo (sleep).

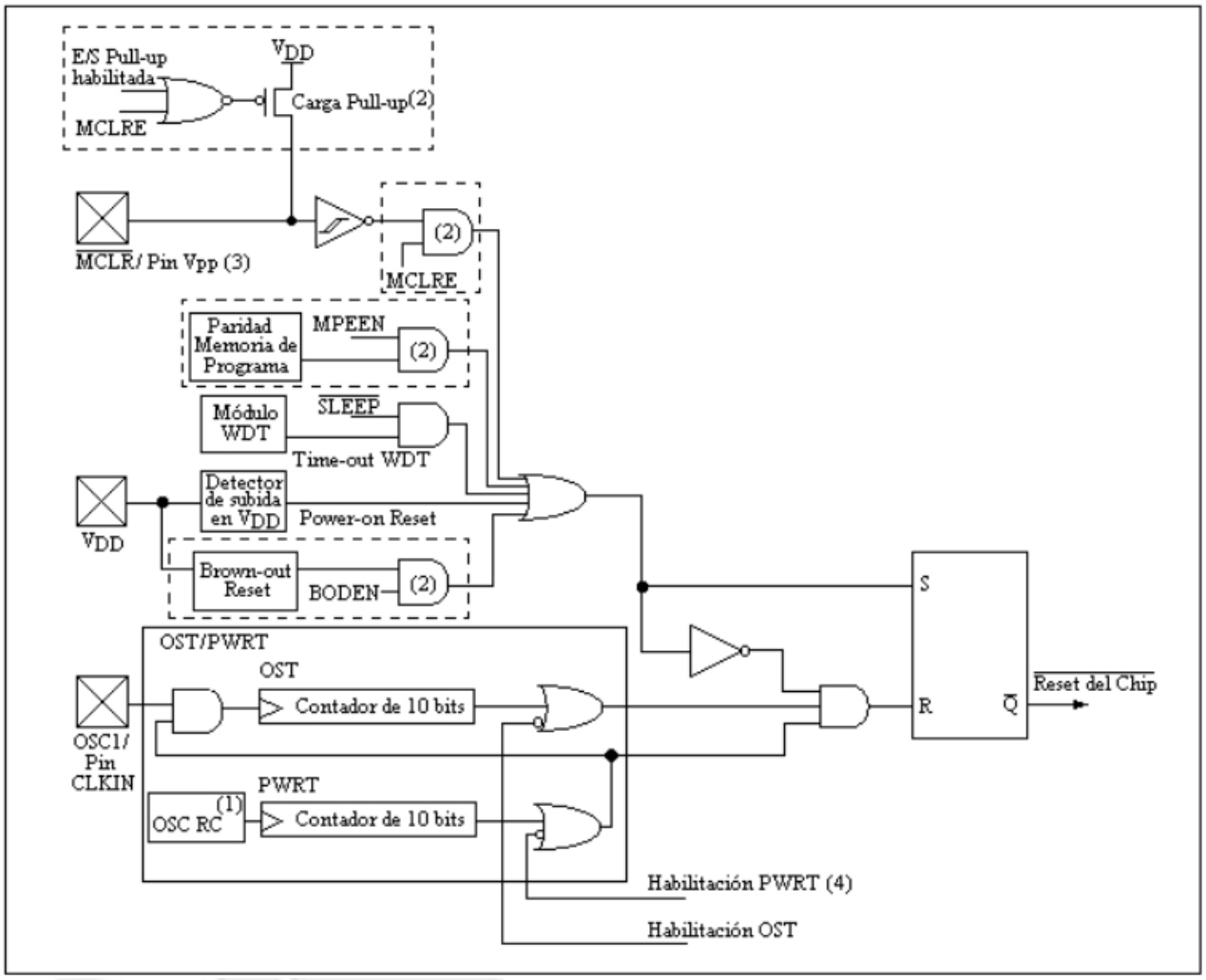


Figura 1.8 Lógica que gobierna el Reset de un sistema.

Nota 1: Este es un oscilador distinto al oscilador del pin CLKIN o al oscilador interno INTRC.
2: Las características que están encerrados en cuadros no están disponibles en todos los modelos, ver hoja características del dispositivo.
3: En algunos modelos, este pin puede estar configurado como entrada de propósito general.
4: En los primeros microcontroladores PIC tenían la configuración de modo que para PWRTE =1 estuviera h habilitado, mientras que el resto de los modelos, la configuración estaba definida para PWRTE =0.

A continuación (tabla 1.4) se muestran los distintos retardos para las posibles situaciones en las que se puede ver el sistema.

TABLA 1.4 Retardos típicos según las situaciones.

CONFIGURACIÓN DEL OSCILADOR	TIEMPO DE POWER-UP		BROWN-OUT RESET	WAKE-UP DESDE SLEEP
	Habilitado	Deshabilitado		
XT, HS, LP	72 ms + 1024TOSC	1024TOSC	72 ms + 1024TOSC	1024TOSC
RC	72 ms	–(1)	72 ms	–(1)

Nota 1: Dispositivos con oscilador Interno/Externo RC tienen un retardo de 250µs.

El Power-on Reset consiste en la activación del reset cuando se detecta la conexión de la alimentación al dispositivo. Dos topologías para el circuito de POR se pueden ver en la figura 1.9, siendo la primera para el caso general y la segunda para el caso en que VDD crezca de forma lenta.

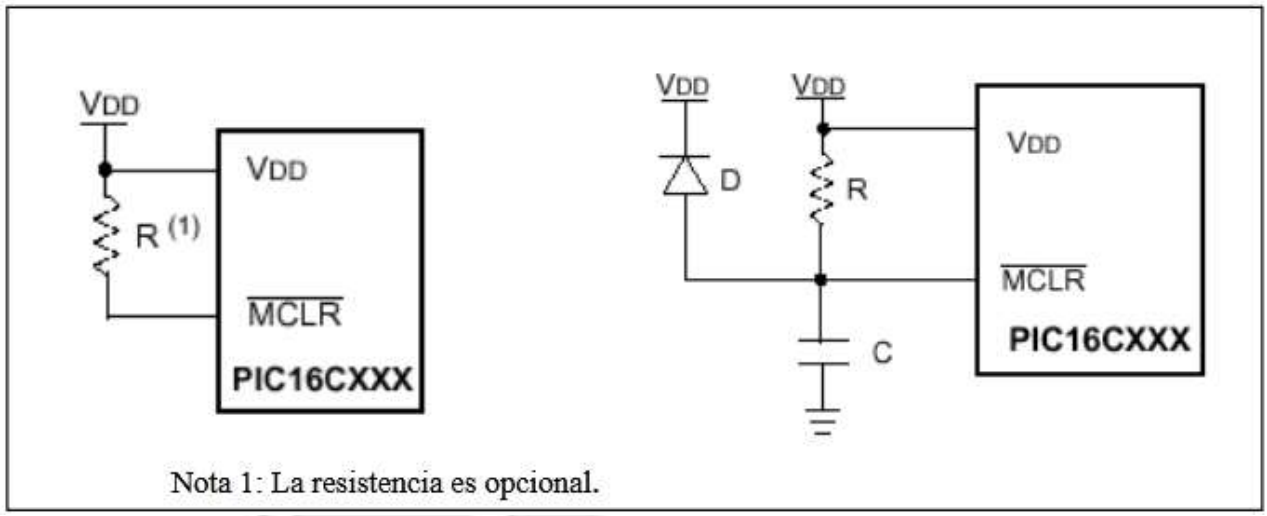


Figura 1.9 Topologías para el circuito de POR.

En el segundo caso el diodo ayuda a la descarga del condensador cuando VDD disminuye su valor.

Las siguientes figuras (1.10– 1.11) muestran el comportamiento temporal de las señales según el caso.

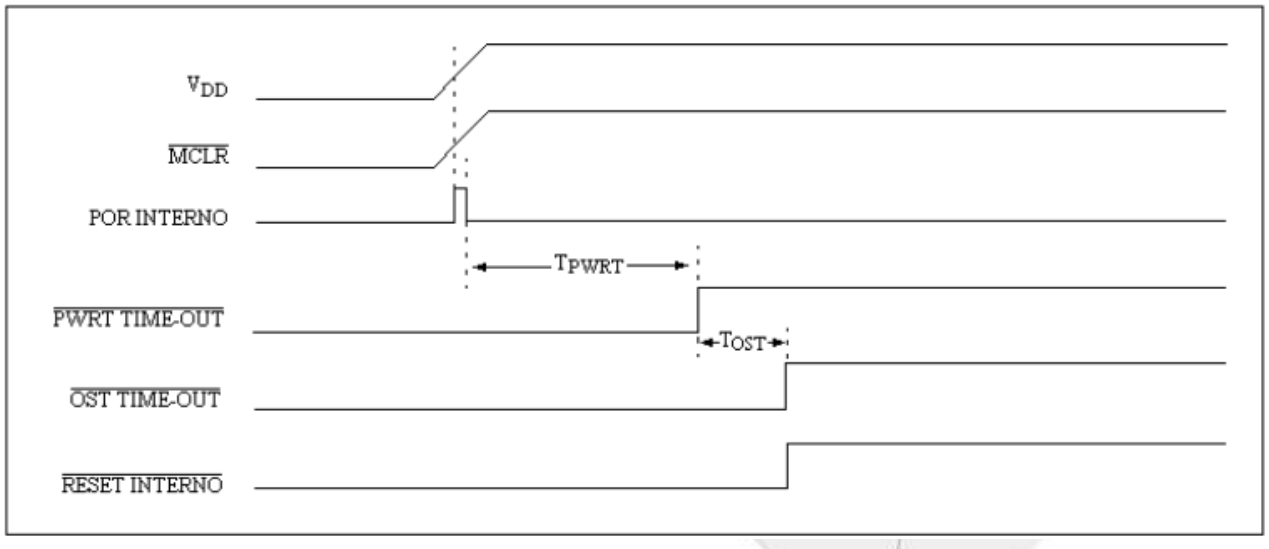


Figura 1.10 Secuencia de Time-out en subida de alimentación.MCLR conectado a VDD.

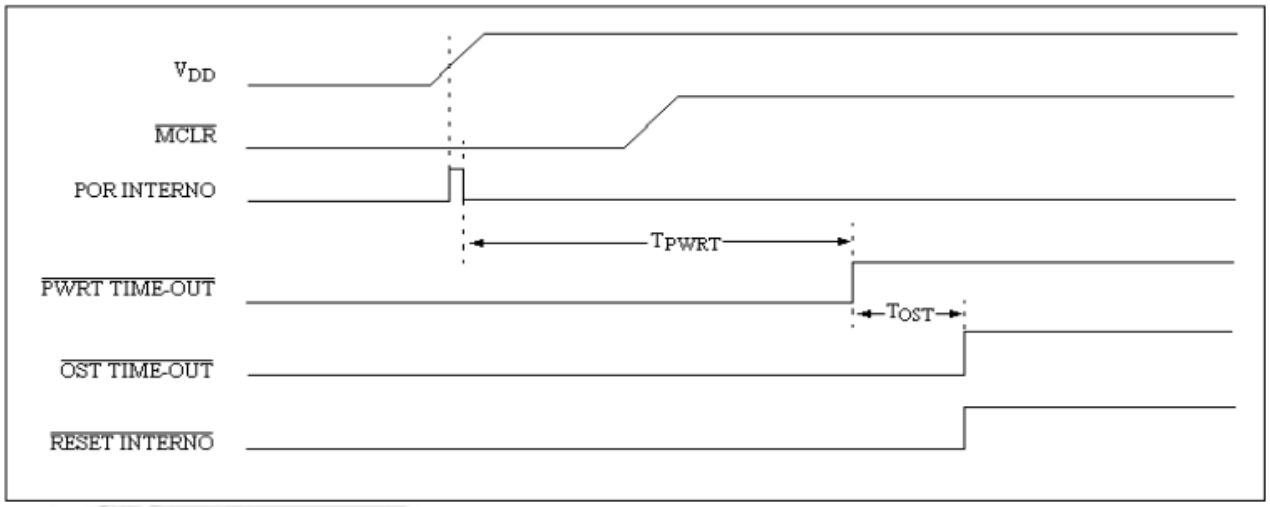


Figura 1.11 Secuencia de Time-out en subida de alimentación.MCLR no conectado a VDD. (Caso1).

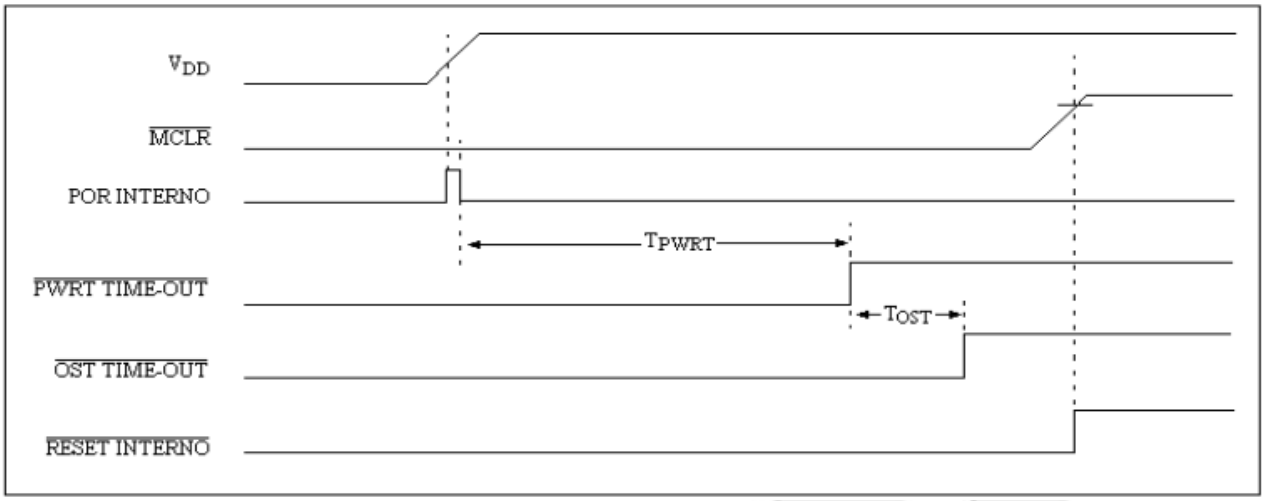


Figura 1.11.1 Secuencia de Time-out en subida de alimentación.MCLR no conectado a VDD. (Caso2).

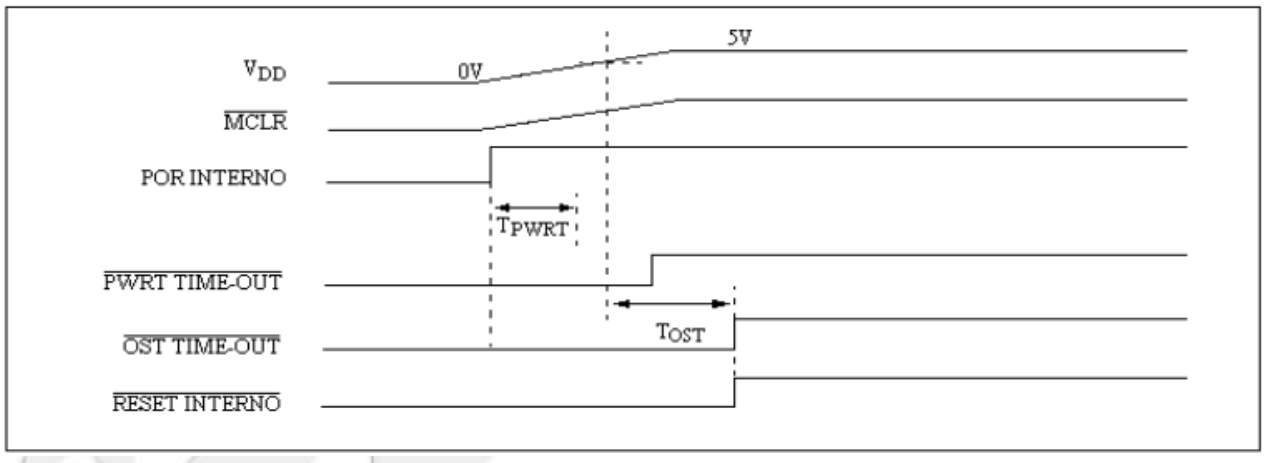


Figura 1.12 Secuencia de Time-out en subida de alimentación. MCLR conectado a VDD(tiempo de subida lento).

El Brown-out Reset consiste en producir un reset en el sistema cuando la tensión de alimentación VDD cae por debajo de un determinado valor, específico para cada PIC. Los que tengan implementados el BOR, tienen un parámetro (35), que es el tiempo mínimo que debe estar VDD por debajo de la tensión de reset, para que éste se produzca. Esto asegura que el elemento no continúa la ejecución del programa fuera de los rangos válidos de operación.

Este tipo de reset se puede habilitar con el bit BODEN de la Palabra de Configuración. Una vez que el nivel de la alimentación se restablezca, hay un retardo de 72ms en desactivarse el reset interno.

Algunas de las situaciones de Brown-Out Reset se muestran en la figura 1.13.

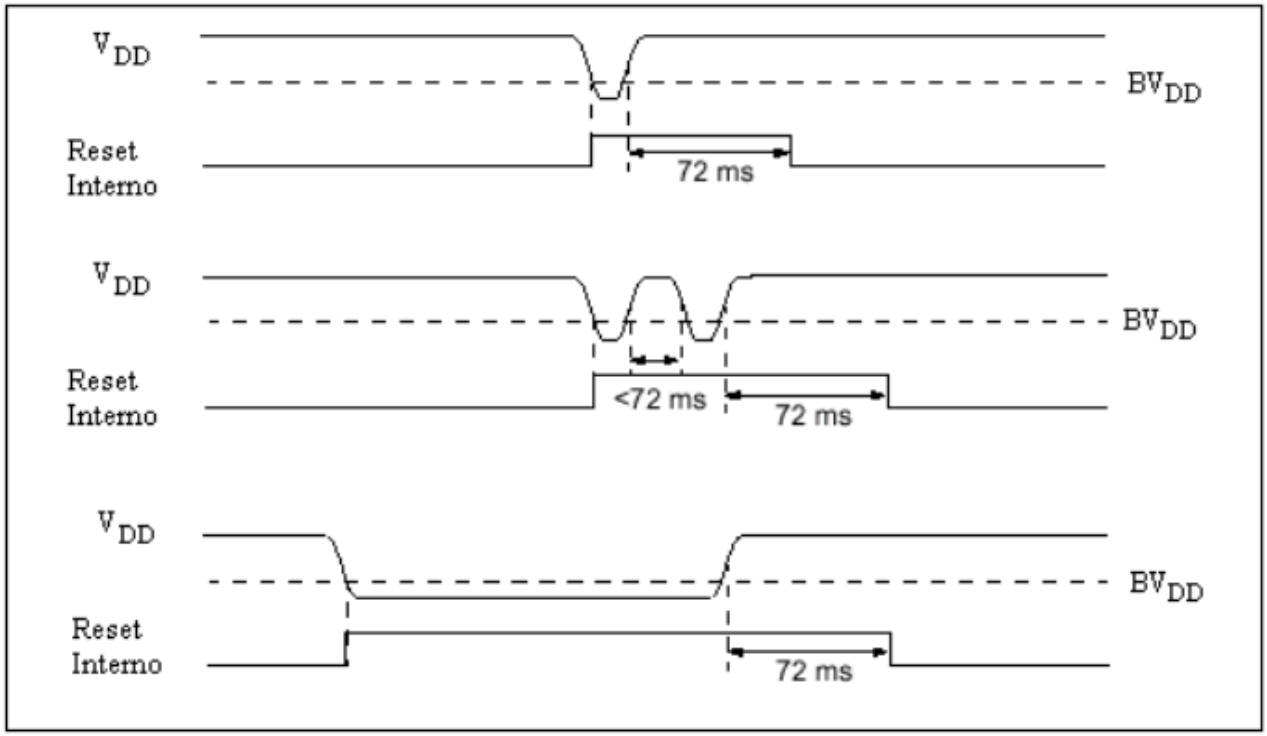
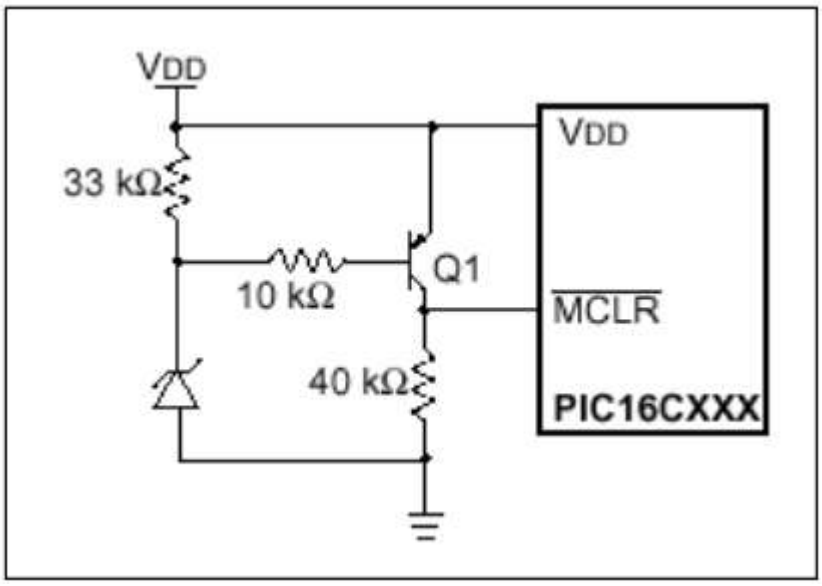


Figura 1.13 Situaciones de Brown-out Reset.

Posibles circuitos usados para un BOR, cuando el circuito no tiene internamente el detector implementado, o cuando la tensión que venga configurada no nos interesa, se muestran en las figuras 1.14 y 1.15.



Instituto Universitario de Tecnología para la Informática – Iutepi

Figura 1.15 Circuito externo 1 de Brown-out.

Este circuito activará el reset cuando VDD esté por debajo de (Vz + 0.7V), donde Vz es el voltaje Zener del diodo.

Nota

1: El circuito de Brown-out Reset interno debe de estar deshabilitado cuando se use esta configuración.

2: Los valores de las resistencias dependen de las características del transistor.

Debido a que existen distintas fuentes de reset, es necesario en determinadas ocasiones, saber qué tipo de reset se ha producido. Esto se hace consultando los bits POR y BOR del registro PCON, y PD del Registro de Estado (tabla 2.4). La tabla 1.5 muestra la forma en que afecta el Reset a los distintos registros.

Programación.

Emprender el estudio de microcontroladores para el desarrollo de proyectos electrónicos que sean la base para nuevas ideas es el objetivo primordial de este ebook, en la cual nos hemos concentrado en dar a los estudiantes algunas herramientas fundamentales con las cuales esperamos abrir un campo de conocimiento en la electrónica de control, a través de un sin número de posibilidades de diseño a partir de una serie de ejemplos prácticos desarrollados en lenguaje Basic para Microcontroladores PIC.

De toda la gama de posibilidades entre las familias de microcontroladores PIC que ofrece Microchip Inc., hemos elegido para empezar el microcontrolador más popular de toda la serie, el PIC16F84, que será utilizado en este ebook para estudiar algunas de sus características a través del desarrollo de actividades que en principio no requieren un nivel de conocimiento elevado y a través del cual daremos los pasos necesarios para adentrarnos en las gamas más altas, de las que se ha seleccionado el microcontrolador PIC16F877 para la realización de proyectos electrónicos de nivel medio y avanzado, y en los que podremos manejar dispositivos periféricos que le dan un gran valor agregado a cada uno de nuestros proyectos y abren posibilidades de desarrollo muy interesantes al lector.

Además, hemos considerado proporcionar la información adecuada referente a las herramientas de desarrollo más importantes en la actualidad, para programación en lenguaje Basic para microcontroladores PIC. En esta ocasión iniciamos con el estudio del compilador PicBasic Pro, de la empresa microEngineering Labs, Inc., la cual ofrece una de las herramientas más populares en el área, debido a que cuenta con una gran variedad de instrucciones que hacen de la programación de microcontroladores una tarea fácil y muy productiva a la hora de desarrollar proyectos que involucren periféricos como pantallas LCD, teclados matriciales, sensores de temperatura, presión, gas, humedad, memorias de datos entre otros, y donde una de las características más relevantes es el considerable ahorro de tiempo, lo que se traduce en efectividad y menos líneas de programa, de tal

Instituto Universitario de Tecnología para la Informática – Iutepi

De manera que el diseñador puede prestar mayor atención a los detalles, logrando así perfeccionar su desempeño en cualquiera de las funciones que desee programar. Otra de las herramientas que hemos decidido incorporar a la obra, es el programador de microcontroladores PIC P16Pro/PicAll, de la página oficial de PicallW, de Boban Dobaj, Diseñador. Este programador soporta una gran cantidad de modelos de las series 12, 16 y 18 de Microchip. Su construcción es sumamente sencilla y de muy bajo costo, además de una serie de ventajas entre las cuales podemos mencionar la alta velocidad de transferencia de datos hacia el dispositivo al momento de ser grabado. Cada capítulo contiene teoría sobre la cual se pretende estudiar el funcionamiento de los microcontroladores y periféricos conectados a él. Para ello hemos desarrollado una serie de prácticas en las que el lector deberá hacer montajes de circuitos en base a los diagramas esquemáticos siguiendo las instrucciones y leyendo detenidamente los comentarios de cada línea de programa. Esperamos con esto proporcionar al lector una base sólida de conocimientos para el desarrollo de proyectos electrónicos, proyectos de robótica y todo aquello que represente una innovación científica en este campo.

Herramientas de Trabajo: Editor, Simulador, Compilador, Programador.

En la elaboración de proyectos electrónicos con microcontroladores PIC, resulta muy importante considerar una serie de herramientas, las cuales vamos a describir a continuación:

Software: para la programación en Lenguaje Basic, contamos con una gran variedad de posibilidades en el mercado, y entre las cuales hemos elegido el compilador Proton Basic.

Es un lenguaje de programación de nueva generación que hace más fácil y rápido el manejo de microcontroladores Pic micro de Microchip. El lenguaje Basic es mucho más fácil de leer y escribir que el lenguaje ensamblador Microchip.

El Proton Basic produce un código que puede ser programado para una variedad de micro controladores PIC que tengan de 8 a 68 pines y varias opciones en el chip incluyendo convertidores A/D, temporizadores y puertos seriales.

Simulador Proteus.

Proteus Design Suite es el programa de simulación electrónica que necesitas si eres estudiante de ingeniería o un profesional del diseño de circuitos.

Proteus no es un nombre que le suene a demasiada gente. Si trabajas en el campo de la medicina quizás lo reconozcas por ser un género de bacteria que incluye diferentes especies como mirabilis o vulgaris que residen en nuestro tracto intestinal. Pero más allá de la microbiología, en el sector del software resulta ser uno de los programas de diseño electrónico más apreciados por estudiantes de ingeniería y profesionales de la electrónica, capaz de ofrecernos una simulación avanzada circuitos electrónicos y microprocesadores.

Se trata de uno de los packs de herramientas electrónicas más completos del mercado ya que en su versión 8.5 (la más nueva de todas) nos permite crear desde nuestro PC todo tipo de PCBs o placas de circuito impreso utilizando casi 800 microcontroladores diferentes, y simular su funcionamiento real directamente desde la vista esquemática

del circuito. Y como no podía ser de otra manera, con los tiempos que corren, integra herramientas con las que diseñar y simular dentro del entorno Arduino, una de las placas más populares del momento.

Los componentes principales de Proteus Design Suite

Este software consta de dos componentes principales en torno a los cuales gira todo el funcionamiento del mismo:

ISIS: son las siglas de Intelligent Schematic Input System o Sistema de Enrutado de Esquemas Inteligente y es el programa que nos permite realizar el diseño en el plano eléctrico del circuito, incluyendo todo tipo de componentes electrónicos como resistencias, bobinas, condensadores, fuentes de alimentación e incluso microprocesadores.

ARES: son las siglas de Advanced Routing and Editing Software o Software de Edición y Rutado Avanzado y es la herramienta dedicada al diseño de placas de circuito impreso o PCBs, con funciones de enrutado, ubicación y edición de componentes electrónicos.

¿Dónde descargar ISIS y ARES? Pues, no puedes hacerte con ellos como aplicaciones standalone, así que para poder aprovecharte de todas sus características tendrás que tener con la versión completa de Proteus, que aunque es de pago, cuenta con una edición de prueba disponible en la web oficial de Labcenter Electronics para que puedas testear sus funciones antes de decidirte si la compras o no.

Además de estos dos programas, este software cuenta con diferentes módulos como VSM que, integrado dentro de ISIS, permite la simulación en tiempo real de diferentes características de los circuitos integrados, o Electra, el módulo de auto-rutado que permite trazar pistas automáticamente entre componentes, buscando el camino más óptimo para mejorar la velocidad del circuito.

Compilador Mikrobasic Pro For Pic

Todo lo que han leído hasta ahora sobre la programación en Basic es sólo teoría. Es útil saberlo, pero no se olvide de que este lenguaje de programación no está tan relacionado con algo concreto y tangible. Podrá experimentar muchos problemas con los nombres exactos de los registros, sus direcciones, nombres de los bits de control particulares, y muchos más al empezar a escribir su primer programa en Basic. El punto es que usted necesita más que una teoría para que el microcontrolador haga algo útil. Teniendo en cuenta de que “Es mejor prevenir que curar”, hay que avisarle de todas las cosas por resolver antes de que empiece a escribir un programa para el microcontrolador. Antes que nada, necesita un programa instalado en la PC que “entiende” el lenguaje de programación a utilizar y que proporciona un entorno de trabajo apropiado. No hay un compilador apropiado para un tipo de compilador, tampoco para todos los microcontroladores. Normalmente se utiliza un software para programar los similares microcontroladores de un fabricante. En las secciones anteriores hemos presentado el lenguaje mikroBasic, especialmente diseñado para programar los PIC. Ahora, cuando sabe lo suficiente sobre ese lenguaje, es hora de presentar el software que utilizará para desarrollar y editar los proyectos. Este software se le denomina Entorno de desarrollo integrado (Integrated Developement Environment - IDE) e incluye todas las herramientas necesarias para desarrollar los proyectos (editor, depurador etc.). Como implica su nombre, mikroBasic PRO por PIC está pensado para escribir los programas para los microcontroladores PIC en Basic. Este compilador contiene las informaciones de arquitectura de los microcontroladores PIC (registros, sus direcciones

Instituto Universitario de Tecnología para la Informática – Iutepi

exactas, módulos de memoria, funcionamiento de sus módulos, juego de instrucciones, disposición de pines etc.). Además incluye las herramientas apropiadas para programar los microcontroladores PIC. Lo primero que tiene que hacer al iniciar el compilador es seleccionar el chip y frecuencia de operación de la lista. Esto no es un final, sino un comienzo. Por fin puede empezar a escribir el programa en Basic. El proceso de crear y ejecutar un proyecto contiene los siguientes pasos:

Crear un proyecto (nombre de proyecto, configuración de proyecto, dependencias entre archivos);

Editar un programa;

Compilar el programa y corrección de errores;

Depurar (ejecutar el programa paso a paso para asegurarse de que se ejecutan las operaciones deseadas);

Programar un microcontrolador (cargar el archivo .hex generado por el compilador en el microcontrolador utilizando el programador PICflash).

Programador de Microcontroladores PIC:

Es una herramienta indispensable con la cual podemos grabar el código generado por el compilador PicBasic para poner en funcionamiento cada uno de los proyectos propuestos en cada capítulo. Existen en internet una gran cantidad de modelos de programadores para microcontroladores PIC, de muy bajo costo y fácil construcción. Consideramos una buena experiencia realizar el montaje de cualquiera de estos diseños, aunque en esta oportunidad nuestra recomendación es el programador P16Pro/Picallw.

Figura 1.16 Quemador de Pics

Programadores

PICStart Plus (puerto serie y USB)

Promate II (puerto serie)

MPLAB PM3 (puerto serie y USB)

ICD2 (puerto serie y USB)

ICD3 (USB)

PICKit 1 (USB)

IC-Prog 1.06B

PICAT 1.25 (puerto USB2.0 para PIC y Atmel)

WinPic 800 (puerto paralelo, serie y USB)

PICKit 2 (USB)

PICKit 3 (USB)

Terusb1.0

Eclipse (PIC y AVR. USB)

MasterProg (USB)

Set de Instrucciones en BASIC para PICs.

@

Sintaxis:

@ instrucción en lenguaje ensamblador

Esta instrucción es utilizada para insertar dentro del código PicBasic, líneas de programa en lenguaje ensamblador. En este caso, cada línea en lenguaje ensamblador debe llevar el símbolo @ al inicio.

Ejemplo:

```
        Led Var Byte    ' Declaración de Variable Led
        TRISB = $00     ' Configura el puerto B como salida
        Led = $00       ' Inicializamos la variable Led
Inicio:
        @bsf _Led,0     ' pone en 1 el bit 0 de la variable Led
        Pause 1000      ' Pausa de 1 segundo
        PORTB = Led     ' Saca dato por el puerto B
        Pause 1000      ' Pausa de 1 segundo
        @bcf _Led,0     ' pone en 0 el bit 0 de la variable Led
        PORTB = Led     ' Saca dato por el puerto B
        GoTo Inicio     ' Salto a inicio
```

End

Nota Importante: para acceder a una variable declarada en PicBasic, desde el lenguaje ensamblador se debe anteponer el símbolo "_" (Guión bajo), ya que de lo contrario no será posible el acceso a ésta. En el ejemplo, se puede ver que para acceder a la variable "Led" desde el lenguaje ensamblador, se antepuso el guión bajo a la variable, quedando "_Led".

ADCin

Sintaxis*:*

ADCin *canal, Variable*

Lee una entrada del conversor A/D y el resultado es almacenado en una variable.

Esta instrucción solo es válida para microcontroladores que tienen convertidor A/D, por ejemplo, el PIC16F877, el PIC18F442, el PIC18F452, el PIC18F458 entre otros.

Asm… EndAsm

Sintaxis:

Asm

*

*

Instrucciones en lenguaje ensamblador

*

*

EndAsm

Esta instrucción permite insertar un conjunto de instrucciones en lenguaje ensamblador y al igual que en la instrucción "@", también aplica el uso del símbolo "_" (Guión Bajo), el cual debe anteceder a las variables que se deseen utilizar.

Branch

Sintaxis:

Branch Variable,[Etiqueta1, Etiqueta2,…EtiquetaN]

La instrucción Branch hace un salto a una etiqueta dependiendo del valor de la variable, es decir, si la variable es igual a 0, el salto se hace a la etiqueta 1; si la variable es igual a 1, el salto se hace a la etiqueta 2; si la variable es igual a 2, el salto se hace a la etiqueta 3, y así sucesivamente.

Ejemplo:

```
I var Byte          ' Declaración de Variable I
TRISB = $00         ' Configura el puerto B como salida
PORTB = $00         ' Inicializa el puerto B
I = 0       ' Inicializa la variable I
inicio:
Branch I,[Led1,Led2,Led3]
Led1:
PORTB   = %00000001     ' enciende          el led en RB0
pause   1000    ' pause de          1 segundo
PORTB   =           %00000000       ' apaga el          led
I = I       +           1           '           suma 1 a            la variable I
GoTo inicio         '           salta a inicio
Led2:


PORTB = %00000010
pause 1000
PORTB = %00000000
I=I+1
GoTo inicio
Led3:
 '          enciende el led en RB1
'           pause de 1 segundo
'           apaga el Led
```

```
'           suma 1 a la variable I
'           salta a inicio
 PORTB  = %00000100      ' enciende          el led en RB2
pause    1000      ' pause de          1 segundo
PORTB    = %00000000      ' apaga el          Led
I = 0                ' Inicializa la variable I
GoTo inicio          ' salta a inicio
End
```

Uso del Entorno de Programación en BASIC con PROTON.

El PROTON PLUS IDE es un entorno de programación basado en un BASIC estructurado orientado a entrada y salida de señales. La utilización de sencillas instrucciones de alto nivel, permite programar los Microcontroladores para controlar cualquier aplicación llevada a cabo por un proceso. Las instrucciones de PBASIC PROTON PLUS IDE permiten controlar las líneas de (entrada /salida), realizar temporizaciones, realizar trasmisiones serie asincrónica, utilizar el protocolo SPI, programar pantallas LCD, capturar señales analógicas, emitir sonidos, etc. y todo ello en un sencillo entorno de programación que facilita la creación de estructuras condicionales y repetitivas con instrucciones como IF...THEN o FOR...NEXT y la creación de etiquetas de referencia.

Algunas aplicaciones de los Microcontroladores

La única limitante de los Microcontroladores es su imaginación. La facilidad de un puerto abierto de (entrada / salida), la capacidad de evaluación de señales para luego decidir una acción y poder controlar dispositivos externos. Hacen que el microcontrolador sea el cerebro de los equipos. Estos son algunos ejemplos de áreas de aplicaciones:

- Electrónica Industrial (Automatizaciones)
- Comunicaciones e interfase con otros equipos (RS-232)
- Interfase con otros Microcontroladores
- Equipos de Mediciones

Instituto Universitario de Tecnología para la Informática – Iutepi

• Equipos de Diagnósticos

• Equipos de Adquisición de Datos

• Robótica (Servo mecanismos)

• Proyectos musicales

• Proyectos de Física

• Proyectos donde se requiera automatizar procesos artísticos

• Programación de otros microcontroladores

• Interfase con otros dispositivos de lógica TTL:

1. Teclado

2. Pantallas LCD

3. Protocolo de comunicación: RS232, I2, SPI

4. Sensores

5. Memorias

6. Real Time Clock (RTC)

7. A/D, D/A, Potenciómetros Digitales

E/S de los Microcontroladores

La dirección de entrada y salida de un contacto dado está enteramente bajo el control de su programa. Cuando un contacto es declarado como una entrada de información, tiene muy poco efecto en los circuitos conectados con él, con menos de 1 microamperio (uA) de consumo interno.

Hay dos propósitos para poner un pin en modo de entrada de información: (1) leer en modo pasivo el estado (1 o 0) de un circuito externo, o (2) para desconectar las salidas que manejan el pin. Para que el consumo de corriente sea él más bajo posible, las entradas de información deben siempre estar cerca de +5 voltios o cercano a la tierra. Los

pines no utilizados en sus proyectos no se deben dejar libres en modo de entrada. Los pines no usados deben ser declarados como salida aunque no estén conectados; esto es para evitar que las entradas estén interpretando el ruido externo como señales lógicas.

Cuando un pin está en modo de salida, internamente está conectado a la tierra o +5 voltios a través de un interruptor muy eficiente del circuito CMOS. Si se carga ligeramente (< 1mA), el voltaje de la salida estará dentro de algunos mili voltios cercanos de la fuente de alimentación (tierra para 0; +5V para 1). Cada pin puede manejar unos 25 mA. Pero Cada puerto de 8 pines no debe exceder de los 50 mA con el regulador externo; los pines de RB0 al RB7 conforman un Puerto B de 8 BITS del PIC 16f877A como ejemplo.

Una vez seleccionado el pic con el que se va a trabajar aparecerá en la sección de Code Explorer la carpeta con sus características de operación obteniendo información cuando se abre la carpeta del pic seleccionado esto es aplicable para todos.

Device 16F877A

Device 16F84A

Device 18F2550

…

…

Configuración del oscilador: Para operar con un oscilador externo se utiliza el comando XTAL=x, siendo x el rango de trabajo del oscilador por ejemplo:

XTAL=4 ; Oscilador de 4 MHZ

XTAL=8 ; Oscilador de 8 MHZ

…

Definición de salidas: Se utiliza el comando Symbol precedido del alias y del pin del pic a utilizar por ejemplo:

Symbol LED=PORTD.0

Symbol SW1 = PORTB.4

Retardos: Esta función realiza retardos según el número de ciclos de instrucción especificado en los valores posibles van desde 1 a 255. Un ciclo de instrucción es igual a cuatro periodos de reloj.

DelayMS time

Esta función realiza retardos del valor especificado en time. Dicho valor de tiempo es en milisegundos y el rango es 0-65535 sirve para obtener retardos más largos así como retardos 'variables'.

Ejemplos:

DelayMS 500 ' Esperar 500ms

DelayMS 1000 ' Esperar 1segundo

DelayUS time

Esta función realiza retardos del valor especificado en time. Dicho valor es en microsegundos y el rango va desde 0 a 65535.

Ejemplos:

DelayUS 500 ' Esperar 500Us

DelayUS 1000 ' Esperar 1ms

High: Sirve para colocar el pin a uno lógico es decir en ON, ejemplo:

High LED

High PORTD.0

LOW: Sirve para colocar el pin a uno cero lógicos es decir en OFF, Ejemplo:

Low LED

Low PORTD.0

GoTo: comando para crear un bucle cerrado y continuación del programa

Nuestro primer programa quedaría así:

titilar un led por el puerto D del pic 16f877A cada 500 ms

Device 16F877A

XTAL=4 ; Oscilador de 4 MHZ

Symbol LED=PORTD.0

INICIO:

High LED

DelayMS 500

Low LED

DelayMS 500

GoTo INICIO

Bien vamos con el segundo proyecto en proton con el comando:

ALL_DIGITAL = True ' Coloca todo los pines como digitales

también de utilizan los condicionales de pbp IF , THEN , ELSE , ENDIF

IF ...THEN

IF Comp { AND/OR Comp ... } THEN Label

IF Comp { AND/OR Comp ... } THEN

Declaración

ELSE

Declaración

ENDIF

Efectúa una o más comparaciones .Cada término Comp puede relacionar una variable con una constante u otra variable e incluye uno de los operadores listados anteriormente .

IF ... THEN evalúa la comparación en términos de CIERTO o FALSO .Si lo considera cierto , se ejecuta la operación posterior al THEN . Si lo considera falso , no se ejecuta la operación posterior al THEN .Las comparaciones que dan 0 se consideran falso .Cualquier otro valor es cierto .Asegúrese de usar paréntesis para especificar el orden en que se deben realizar las operaciones .De otra manera , la prioridad de los operadores lo determina y el resultado puede no ser el esperado .

IF..THEN puede operar de dos maneras. De una forma, el THEN en un IF..THEN es esencialmente un

GOTO. Si la condición es cierta, el programa irá hacia la etiqueta que sigue al THEN. Si la condición es falsa, el programa va a continuar hacia la próxima línea después del IF..THEN. Otra declaración no puede ser puesta después del THEN; sino que debe ser una etiqueta.

```
If LED = 1 Then alarma

‘ si el LED está a uno lógico(1), salta a la etiqueta alarma
```

En la segunda forma, IF..THEN puede ejecutar condicionalmente un grupo de declaraciones que sigan al THEN. Las declaraciones deben estar seguidas por un ELSE o un ENDIF para completar la estructura.

```
If LED_1 = 0 Then ' Chequea estado del Led

LED_1 = 1 'Invierte estado del Led

LED_1 = 0

EndIf
```

si el valor de LED no es cero lógico entonces establecer y terminar el condicional .

Operaciones lógicas básicas

Existen 3 operaciones lógicas llamadas: AND, OR y NOT.

1. AND esta función es verdadera cuando todas sus entradas son verdaderas. Y es falso cuando cualquiera de sus entradas son falsas. Se interpreta como la multiplicación binaria.

2. OR esta función es falsa cuando todas sus entradas son falsas. Y es verdadera cuando cualquiera de sus entradas sea verdadera. Se interpreta como la suma binaria.

3. NOT es la negación del resultado si es verdadero lo convierte en falso. Si es falso lo convierte en verdadero.

Estas son las 3 operaciones fundamentales en la lógica binaria, a partir de estas funciones se derivan otras más que son las combinaciones de las 3 funciones básicas.

Formato de conversión numérica del PROTON PLUS IDE

El editor PROTON PLUS IDE utiliza símbolos para identificar los distintos sistemas numéricos. Los números hexadecimales se representan con el signo de moneda ($), los números binarios con el símbolo de porcentaje (%), los caracteres ASCII encerrados entre comillas (") y los números decimales de forma directa. Vea el siguiente ejemplo:

75 ‘Decimal

%01001 ‘Binario

$65 ‘Hexadecimal

"A” ‘ASCII "

Las 3 instrucciones siguientes contienen el mismo significado:

PORTB = 14

PORTB = $E

PORTB = %1110

Manejo de lcd con proton plus ide

```
Device = 16F877

XTAL = 4

LCD_DTPIN = PORTD.4

LCD_RSPIN = PORTD.2

LCD_ENPIN = PORTD.3

LCD_INTERFACE = 4

LCD_LINES = 2

LCD_TYPE = 0

ALL_DIGITAL = True

DelayMS 150

Cls

Main:

Print At 1,1, "TUTO PROTON PLUS"

While 1=1

Wend
```

CONFIGURACION OSCILADOR EXTERNO EN PIC BASIC DEL PIC 16F88

El primer paso es elegir en PIC SIMULATOR IDE, desde el menú "Opciones" -> "Select Microcontroller", el microcontrolador PIC16F88. Luego, debemos configurar los bits correspondientes. Lo destacable por ahora de esta configuración es que estamos dejando la memoria (FLASH y EEPROM) sin protección, que el pin RESET se va a comportar como I/O y que usaremos como oscilador el oscilador interno INTRC. En el caso que nos atañe utilizaremos el oscilador interno de este PIC ya que es muy completo y estable. Cuando digo completo es porque nos permite trabajar a varias frecuencias distintas entre ellas a 4Mhz y 8Mhz. Y lo mismo que para el WDT, hay que terminar de configurarlo desde nuestro código fuente.

REGISTRO PARA CONFIGURAR OSCILADOR INTERNO DEL PIC 16f88

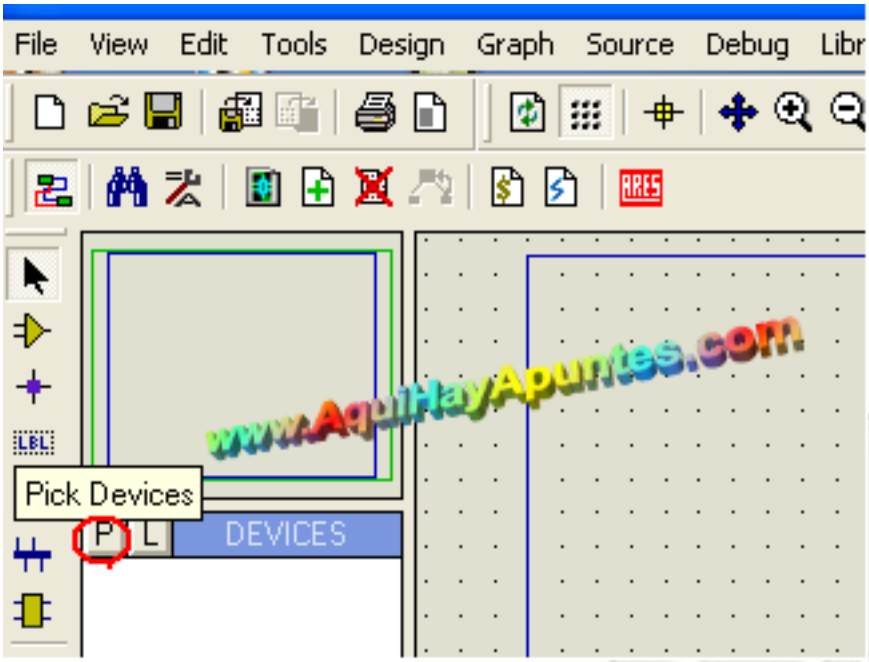
Para configurar la frecuencia de trabajo del Oscilador Interno del PIC 16f88 solo hay que cambiar los bits 4, 5 y 6 del registro OSCCON. Se deben cambiar los bits que corresponden por el valor al que se quiera hacer trabajar al oscilador interno, por cierto se pueden poner en hex. O en binario (%01000110). Además se tienen que configurar los bits en Options- Configure Bits del PIC simulador ide., y poner que se va a trabajar con el oscilador interno: OSCILLATOR SELECTION: INTOSC. En documento anexo PDF está la explicación de los Bits del registro OSCCON.

Un ejemplo de un programa de parpadeo de un led, configurando el oscilador interno.

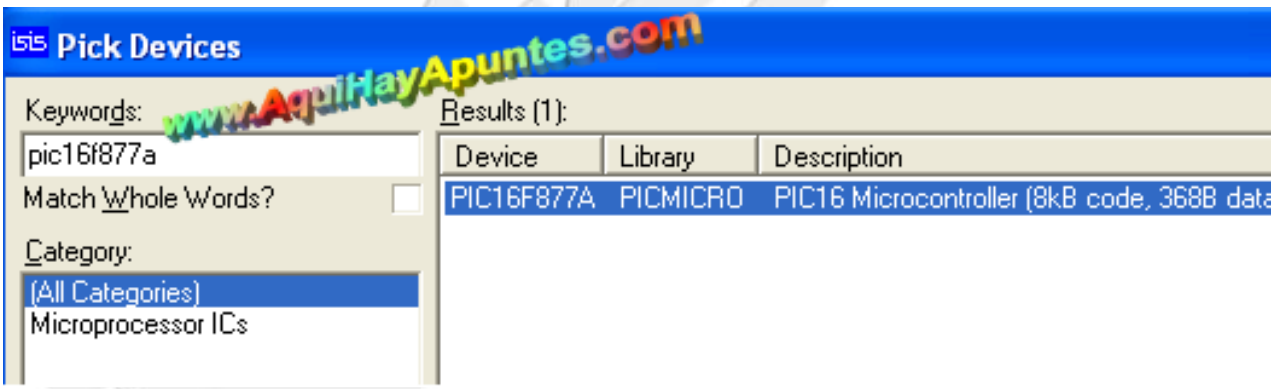
```
AllDigital

OSCCON = %%01100110 'Se configura reloj interno a 4Mhz

TRISB = 0

inicio:

PORTB.0 = 1

WaitMs 500

PORTB.0 = 0

WaitMs 500

Goto inicio
```

Uso del Simulador PROTEUS Aplicado al PICs de la Gama Media.

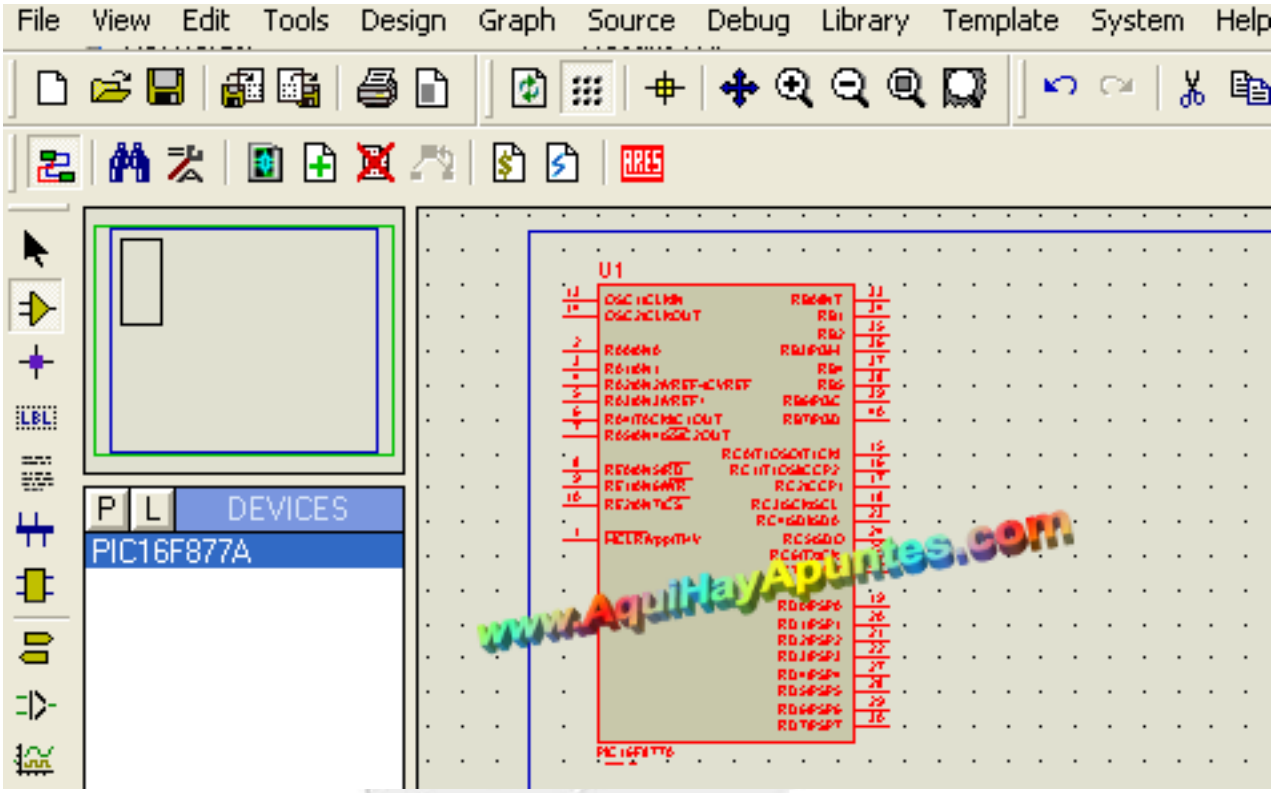
Arrancamos nuestro simulador Proteus y pasamos a colocar nuestros dispositivos en el área de trabajo. Empezaremos colocando el PIC, para ello hacemos clic en el botón que pone Pick Devices según se muestra en la figura de abajo y las sucesivas del tutorial tomado de www.aquihayapuntes.com:



En la ventana que nos aparece en el campo Keywords escribimos el nombre de nuestro PIC.



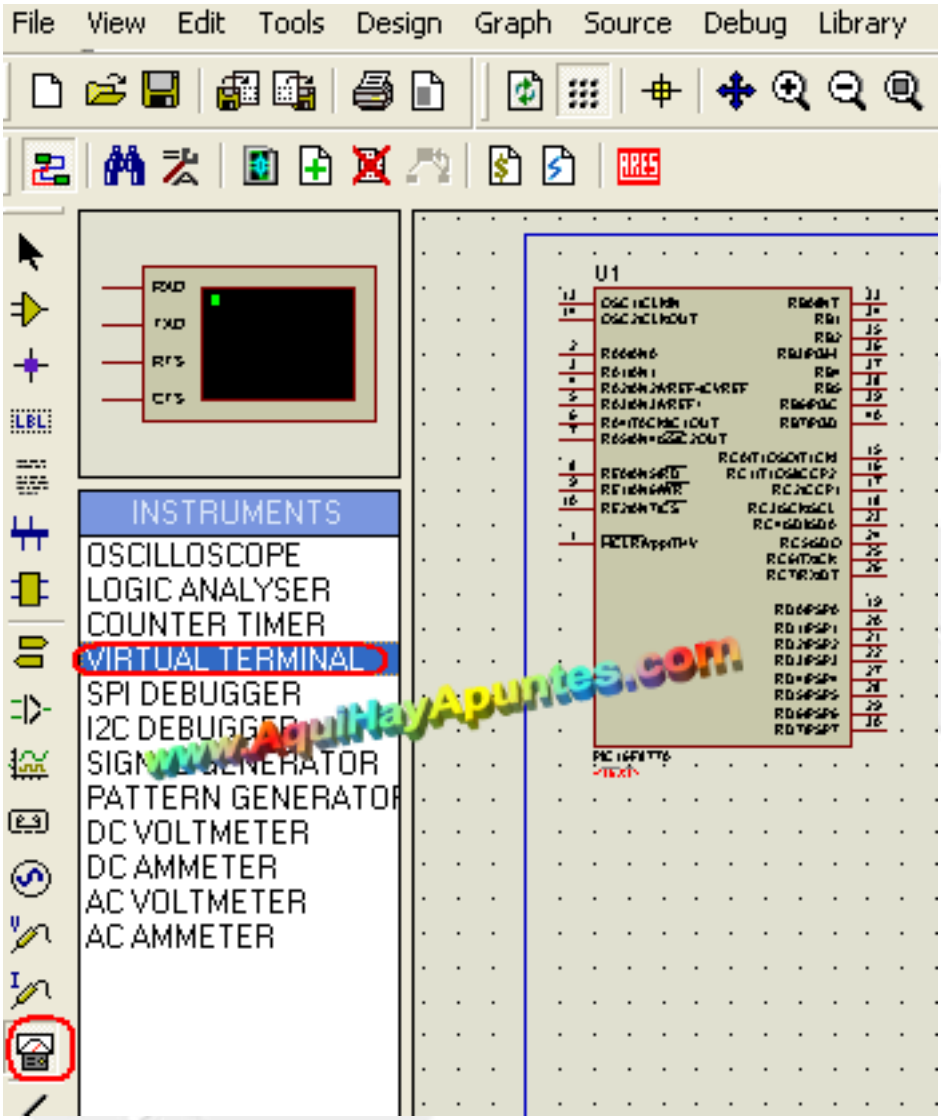
Una vez seleccionado hacemos doble clic sobre el para incorporarlo a nuestro proyecto.



Bien, vamos por el segundo y último elemento que necesitamos para simular nuestro programa. Hay que tener en cuenta que el simulador es capaz de hacer funcionar nuestro circuito sin algunos elementos que serían necesarios si decidimos montar nuestro circuito en una placa real (por ejemplo la alimentación del PIC y el cristal de cuarzo).

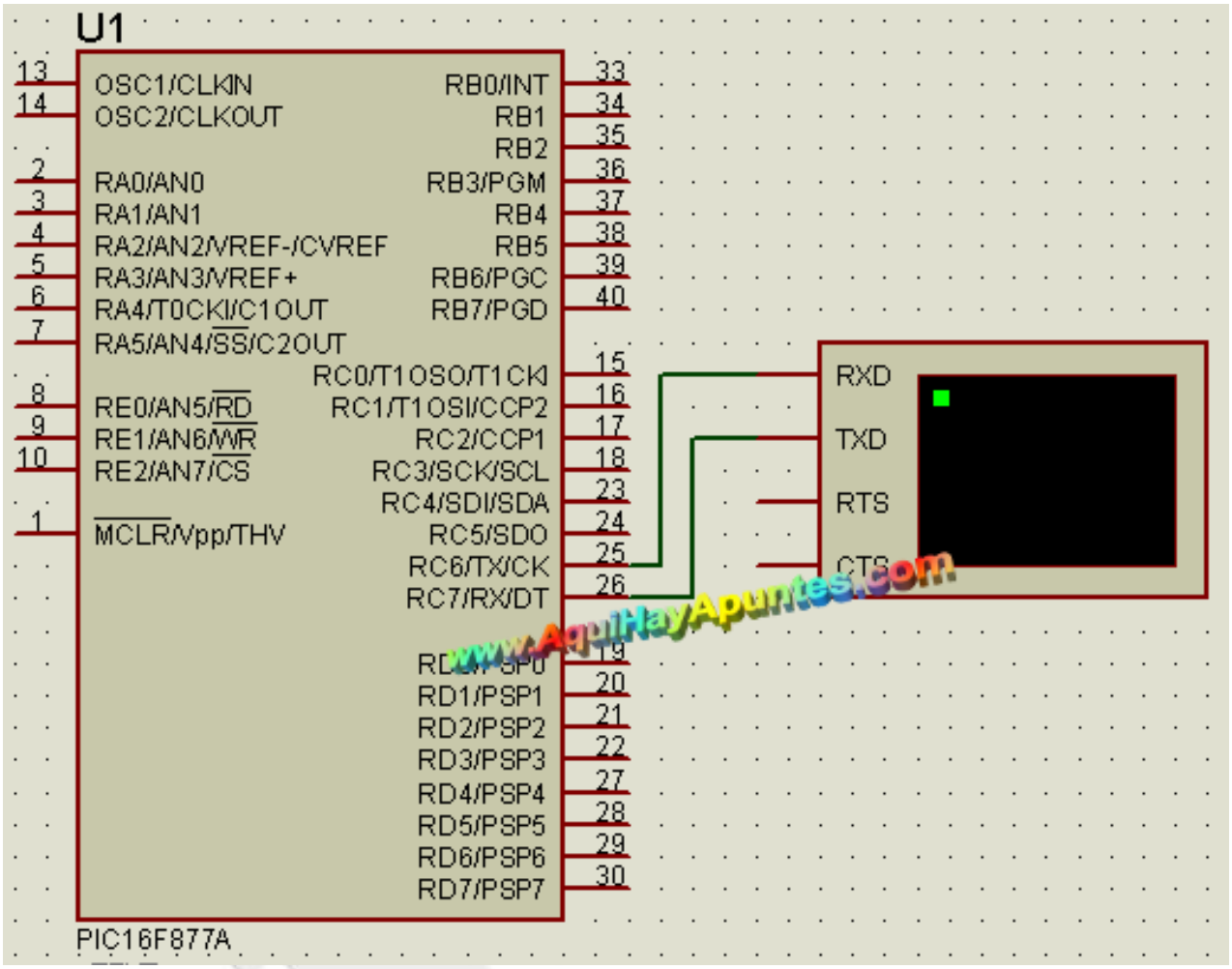
El segundo elemento que necesitamos es un Terminal Virtual que hará las veces de monitor, para poder ver las salidas en formato texto de nuestro PIC como si se tratará del símbolo del sistema en un ordenador de escritorio con el Windows instalado. En la figura de abajo se muestra dónde podemos incorporar dicho instrumento.

Instituto Universitario de Tecnología para la Informática – Iutepi



Con esto ya tendremos los dos elementos necesarios para simular nuestros programas, recordemos que en este curso se va a ver las generalidades del lenguaje C aplicadas a este compilador, en el caso de las aplicaciones prácticas que empezaremos pronto en otro artículo tendremos que hacer un circuito independiente para cada ejemplo ya que cada uno de ellos incorporará elementos diferentes como: diodos Led, motores, teclados, displays, etc.

La interconexión de los dos dispositivos es muy sencilla según se muestra en la figura de abajo, solo hay que hacer clic con el puntero del ratón en forma de lápiz entre los terminales que queremos conexionar:



El pin del PIC que habíamos elegido como transmisión de datos en nuestro programa irá conectado al terminal RXD de recepción de datos en el Terminal Virtual y viceversa.

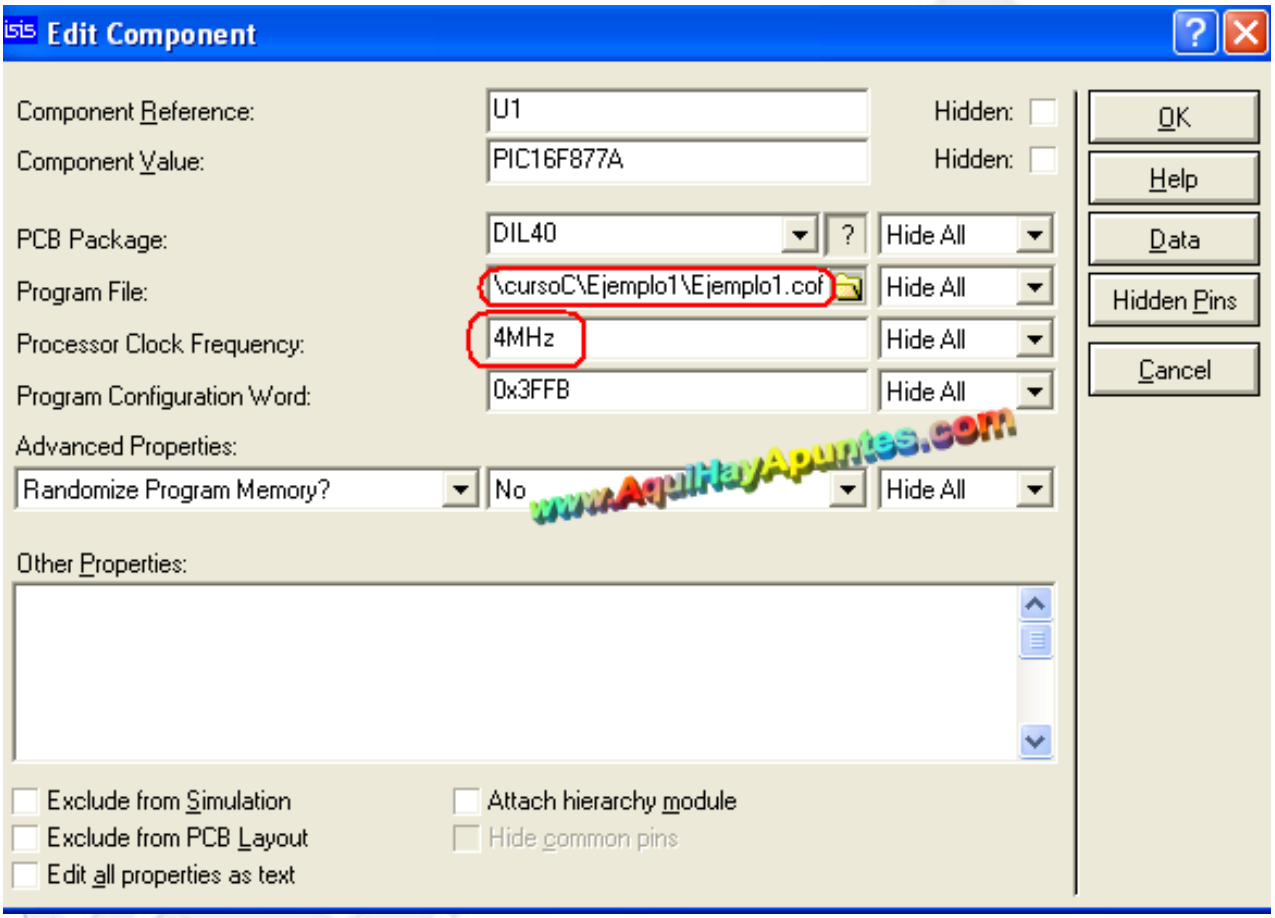
Autoevaluación

- ¿Cuáles son las gamas en las que se clasifican los microcontroladores?
- ¿Qué es un PIC de gama media?
- Explique cómo se efectúa la programación de un PIC
- Explique que es un Editor, Simulador, Compilador y Programador
- Mencione y explique algunas instrucciones básicas en Basic para Pics
- Explique la instrucción condicional If y algún ejemplo del uso de la misma
- Explique la instrucción condicional Go to y algún ejemplo del uso de la misma
- Explique el uso del simulador Proteus con circuitos con Pics

Instituto Universitario de Tecnología para la Informática – Iutepi

El Cargador de Programas del PIC.

Tomando el tutorial de la página www.aquihayapuntes.com veremos la manera correcta, en las sucesivas figuras, de cargar nuestro programa en el PIC para poder simularlo, para ello hacemos doble clic sobre el PIC y nos aparecerá la ventana de la figura de abajo:



Los valores que en un principio tenemos que introducir para que nuestra simulación funcione son los que están señalados en la figura de arriba. En Program File pincharemos sobre la carpeta y seleccionaremos el archivo con extensión .cof que se había creado al compilar nuestro programa, si en vez de este seleccionamos el que tiene extensión .Hex funcionará igual pero no podremos realizar la simulación paso a paso. El otro valor a tener en cuenta es que la frecuencia del reloj del PIC debe coincidir con el valor que le habíamos puesto en el programa en nuestro caso 4 MHz.

Una vez hecho esto guardamos nuestro proyecto.

Autoevaluación

- Explique cómo utilizar el cargador de programas
- Encienda su Pc busque familiarizar con el software

Manejo de Interruptores y Leds.

Los pines en los puertos de un microcontrolador pueden ser configurados también como entradas, como se detalla en el contenido del proyecto #1, donde se explica claramente que al poner un "1" en un bit de un registro de configuración de puerto, ya sea TRISA o TRISB, para el caso del PIC16F84A, éste se comportará como entrada. Realice el montaje de la figura 3.1 y analice el programa que se muestra a continuación.

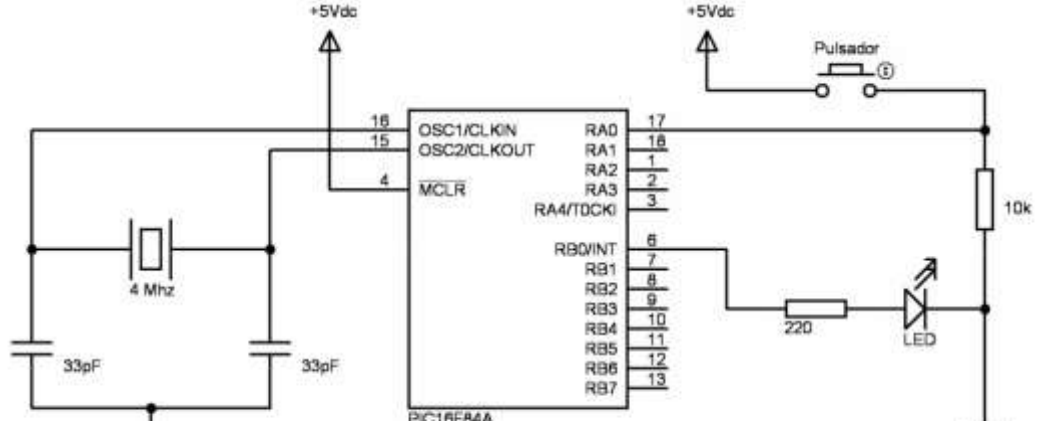


Figura 3.1 Interruptores y Leds

Con la instrucción If – Then podemos tomar decisiones a lo largo de un programa, basadas en condiciones específicas definidas por el programador.

El siguiente programa hace destellar un LED conectado en RB0, solo cuando el pulsador es activado:

```
'****************************************

'*  Nombre      : Proyecto3.pbp       *
'*  Autor       : Nombre del Autor    *
'*  Copyright   : Copyright (Año)     *
'*  Fecha       : Fecha               *
'*  Versión     : 1.0                 *
'****************************************


Define Osc 4            ' Define el Oscilador para un Cristal ' de
                        4 Mhz.


    TRISA = %11111      ' configura el Puerto A como Entrada
    TRISB = %00000000 ' Configura el Puerto B como Salida


    PORTB = $00         ' Inicializa el puerto B


Inicio:


    If PORTA.0 = 1 Then PORTB.0 = 1 ' Pregunta si RA0 = 1, si se cumple
```

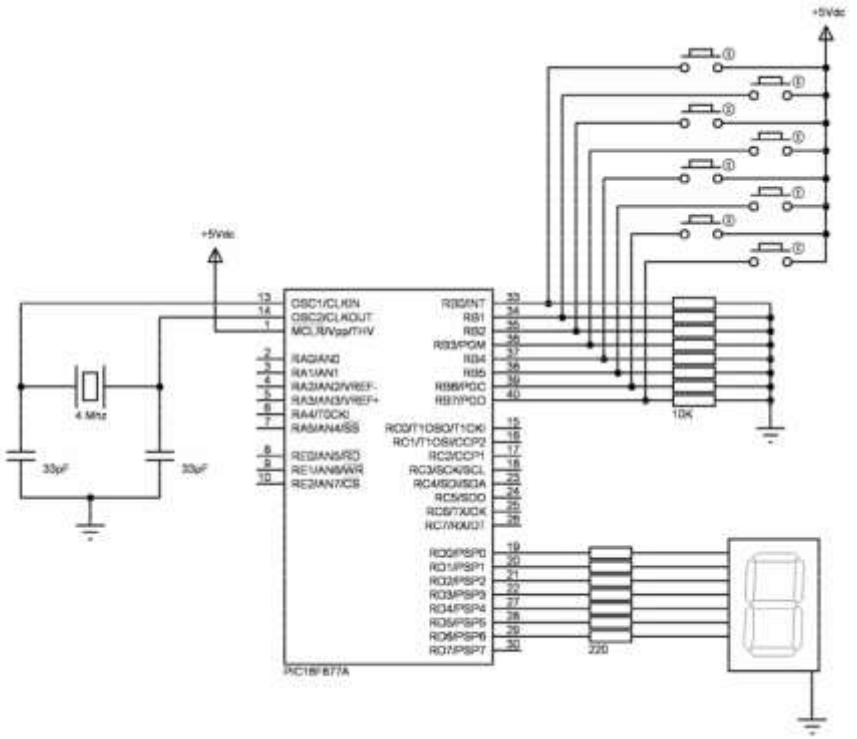
```
                                    ' la condición entonces enciende el Led.

Pause 1000' Hace una pausa de 1 segundo (1000 ms)


Low PORTB.0         ' Apaga el Led


Pause 1000          ' Hace una pausa de 1 segundo (1000 ms)
GoTo inicio         ' Salta a la etiqueta "Inicio"


End
```

Para verificar si el pulsador está activado, se pregunta si RA0 = 1. Esto es posible gracias a la instrucción “IF” la cual genera un resultado siempre que la condición planteada se cumpla, y para lo cual debemos utilizar necesariamente su complemento “Then” seguido de la acción a ser tomada.

En este caso si el pulsador ha sido activado, entonces RA0 = 1, es decir, se cumple la condición y por lo tanto RB0 = 1, es decir, el LED enciende.

Displays.

En este ejemplo empleamos un microcontrolador PIC16F877A, con el cual nos hemos planteado la lectura de ocho pulsadores conectados al puerto B, de tal manera que al activar uno de ellos podemos mostrar un dígito decimal en un Display de siete segmentos.



En el diagrama esquemático de la figura anterior 3.2 se pueden observar ocho pulsadores normalmente abiertos, los cuales una vez activados generan un estado lógico alto en el puerto seleccionado (puerto "B"), el cual ha sido configurado como entrada.

El display de 7 segmentos de cátodo común, se encuentra conectado al puerto "D", donde el bit menos significativo RB0 corresponde al segmento "a" del display, RB1 corresponde al segmento "b", RB2 corresponde al segmento "c", RB3 corresponde al segmento "d", RB4 corresponde al segmento "e", RB5 corresponde al segmento "f" y RB6 corresponde al segmento "g".

El siguiente programa es una forma básica de tomar una lectura de cada pulsador conectado al puerto "B" y generar un resultado en el puerto de salida al cual hemos conectado un display de 7 segmentos:

```
Define Osc 4          ' Define el Oscilador para un Cristal ' de
                      4 Mhz.

    TRISB = $FF       ' Configura el Puerto B como Entrada
```


Instituto Universitario de Tecnología para la Informática – Iutepi

```
    TrisD = $00      ' Configura el Puerto D como Salida

Inicio:

    ' A continuación se verifica cada pin del puerto B,

    ' si hay un 1 lógico en alguna de las entradas el
    ' puerto D se actualiza con el dato correspondiente

    ' para generar en el Display un dígito decimal.


                                ' gfedcba
                                ' |||||||

    If PORTB.0 = 1 Then PortD = %00111111  ' Enciende los segmentos correspondientes ' al
                                           dígito "cero" en el display.

    If PORTB.1 = 1 Then PortD = %00000110  ' Enciende los segmentos correspondientes ' al
                                           dígito "uno" en el display.

    If PORTB.2 = 1 Then PortD = %01011011  ' Enciende los segmentos correspondientes ' al
                                           dígito "dos" en el display.

    If PORTB.3 = 1 Then PortD = %01001111  ' Enciende los segmentos correspondientes ' al
                                           dígito "tres" en el display.

    If PORTB.4 = 1 Then PortD = %01100110  ' Enciende los segmentos correspondientes ' al
                                           dígito "cuatro" en el display.

    If PORTB.5 = 1 Then PortD = %01101101  ' Enciende los segmentos correspondientes ' al
                                           dígito "cinco" en el display.

    If PORTB.6 = 1 Then PortD = %01111101  ' Enciende los segmentos correspondientes ' al
                                           dígito "seis" en el display.

    If PORTB.7 = 1 Then PortD = %00000111  ' Enciende los segmentos correspondientes ' al
                                           dígito "siete" en el display.

    GoTo Inicio      ' Salta a la etiqueta "Inicio"

    End
```

Potencia (Triacs. Relés. Optoacopladores).

El Opto-acoplador:

El opto-acoplador es un componente muy útil cuando se requiere acoplar circuitos electrónicos digitales con etapas de manejo de potencia o con otros circuitos.

Este componente en una de sus versiones, se compone básicamente de un diodo LED el cual se encarga de iluminar un fototransistor, para que éste conduzca corriente a través del colector.

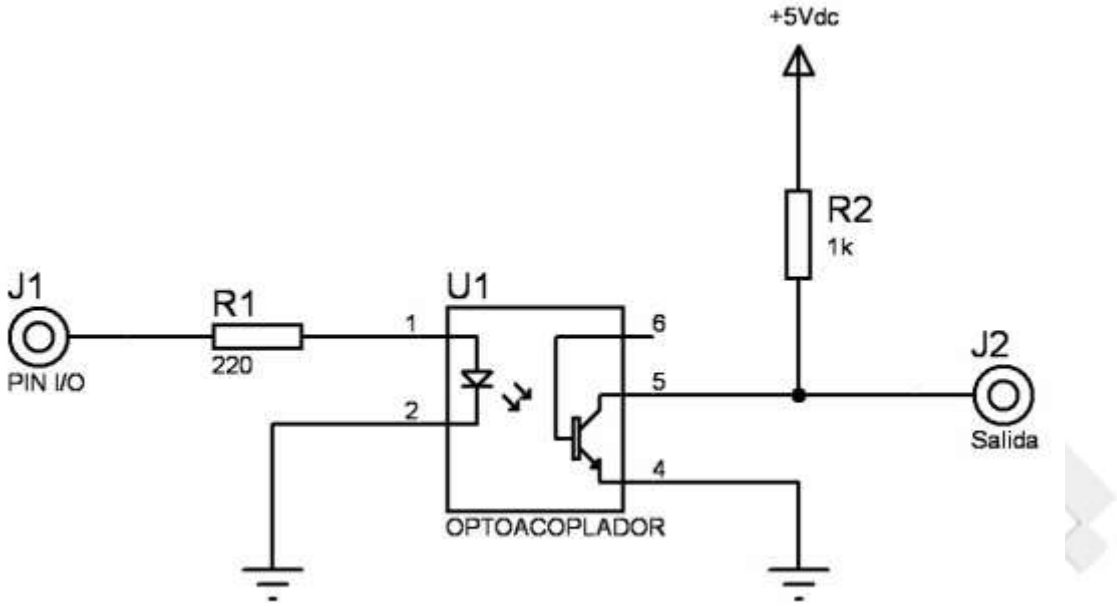


Figura 3.3 Optoacoplador

En la configuración de la figura 3.3, cuando en el pin I/O aplicamos un 1 lógico (5V), el LED del opto-acoplador enciende y el fototransistor conduce la corriente a tierra; por lo tanto, en la salida tendremos un 0 lógico (0V).

Si apagamos el LED, el transistor no conduce, de tal manera que en la salida tendremos un 1 lógico (5V).

En la configuración de la figura 3.4, cuando en el pin I/O aplicamos un 1 lógico (5V), el LED del opto-acoplador enciende y el fototransistor conduce para poner en la salida un 1 lógico (5V). Mientras haya un 0 lógico en la entrada, el fototransistor permanecerá abierto entre el emisor y colector, dando como resultado un 0 lógico (0V) en la salida.

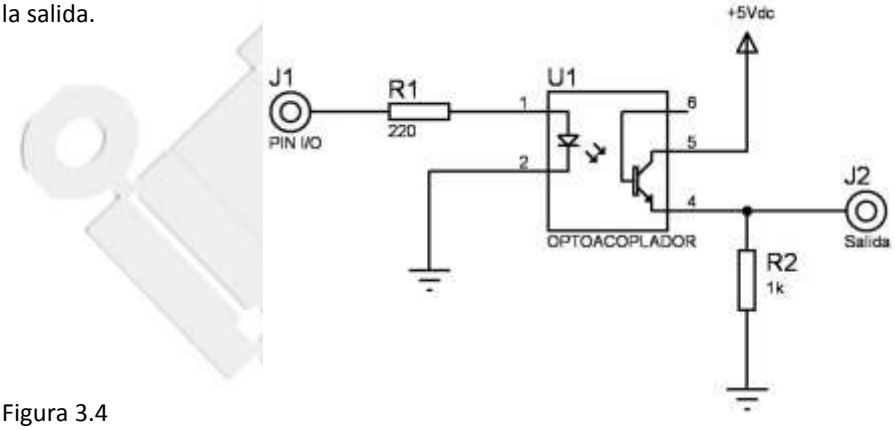


Figura 3.4

Una configuración muy común para el control de dispositivos de potencia como motores eléctricos, luces incandescentes, solenoides, etc., se puede ver en la figura 3.5, la cual se basa en cualquiera de los dos circuitos antes

mencionados (figura 3.3 y figura 3.4), en la cual se ha incluido un relé través del cual circulará la corriente necesaria entre sus contactos, para hacer funcionar cualquiera de estos dispositivos de potencial.

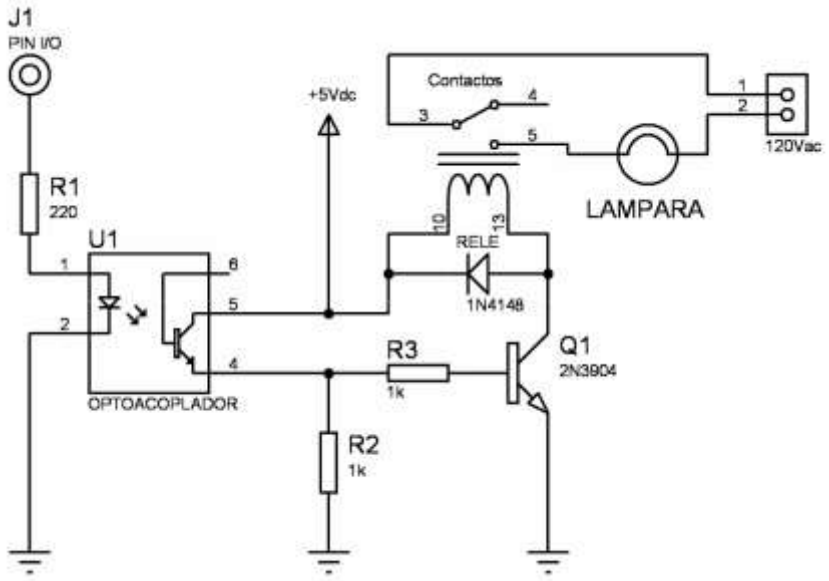


Figura 3.5

Relés.

Es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Fue inventado por Joseph Henry en 1835.

Dado que el relé es capaz de controlar un circuito de salida de mayor potencia que el de entrada, puede considerarse, en un amplio sentido, como un amplificador eléctrico. Como tal se emplearon en telegrafía, haciendo la función de repetidores que generaban una nueva señal con corriente procedente de pilas locales a partir de la señal débil recibida por la línea. Se les llamaba «relevadores».

Triacs.

Un TRIAC o Tríodo para Corriente Alterna es un dispositivo semiconductor, de la familia de los tiristores. La diferencia con un tiristor convencional es que éste es unidireccional y el TRIAC es bidireccional. De forma coloquial podría decirse que el TRIAC es un interruptor capaz de conmutar la corriente alterna.

Su estructura interna se asemeja en cierto modo a la disposición que formarían dos SCR en direcciones opuestas. Posee tres electrodos: A1, A2 (en este caso pierden la denominación de ánodo y cátodo) y puerta (gate). El disparo del TRIAC se realiza aplicando una corriente al electrodo de gate/puerta.

Autoevaluación

- Explique cómo deben ser declarados los interruptores y los leds en un programa?
- Mencione las diferencias entre un display cátodo común y ánodo común.
- Como influye en la programación con displays el uso de uno u otro display
- ¿Qué utilidad poseen los optoacopladores en un circuito?
- Dibuje los esquemas de conexión de un optoacoplador en un circuito
- ¿Por qué se prefiere el uso de optoacopladores cuando se trabaja con PIC y motores eléctricos?
- ¿Qué son los relés y triacs?
- ¿Cómo se conectan a un PIC?

Instituto Universitario de Tecnología para la Informática – Iutepi

Módulos:

Conversor A/D.

El CAD conversor analógico digital PIC (no todos los PIC lo tienen, para los ejemplos se utilizará el **PIC16F877A**), permite medir señales analógicas en forma digital, para ello el PIC cuenta con pines por donde le llegará la señal analógica, estos pines deben configurarse como entradas analógicas, el conversor analógico digital PIC cuenta con un circuito que carga un condensador interno al PIC con la tensión analógica que le está llegando a la entrada analógica, luego la tensión almacenada en el condensador lo convierte en un número binario de 10 bits que representará la tensión almacenada en el condensador, este número binario se guarda en sus registros **ADRESH** y **ADRESL** de 8 bits cada uno pero estos actúan como un solo registro de 16 bits, en el registro ADRESH se guardan los bits más significativos y en el registro ADRESL se guardan los bits menos significativos, el número que representa la tensión almacenada en el condensador y guardado en forma binaria dentro de estos registros será de 10 bits para el PIC16F877A, la cantidad de bits de este número depende del conversor analógico digital PIC del microcontrolador PIC utilizado.

De Interrupciones.

Módulo KBI:Permite generar interrupciones por teclado externamente.

Módulo IRQ:Es un módulo que permite al MCU recibir interrupciones externas mediante un puerto de entrada.

Módulo SIM: Generación y control de clock, control de Master reset, control de interrupciones, manejo de la temporización en el MCU.

Otros Módulos.

Módulos CCP

El módulo CCP (Captura/Comparación/PWM) es un periférico que le permite medir y controlar diferentes eventos. El modo de captura proporciona el acceso al estado actual de un registro que cambia su valor constantemente. En este caso, es el registro del temporizador Timer1. El modo de comparación compara constantemente valores de dos registros. Uno de ellos es el registro del temporizador Timer1. Este circuito también le permite al usuario activar un evento externo después de que haya expirado una cantidad de tiempo predeterminada. PWM (Pulse Width Modulation - modulación por ancho de pulsos) puede generar señales de frecuencia y de ciclo de trabajo variados por uno o más pines de salida. El microcontrolador PIC16F887 dispone de dos módulos CCP - CCP1 y CCP2. Ambos son idénticos en modo normal de funcionamiento, mientras que las características del PWM mejorado están disponibles sólo en el modo CCP1. Ésta es la razón por la que en este capítulo se describe detalladamente el funcionamiento del módulo CCP1. Con respecto al CCP2, se presentarán sólo las características que lo distinguen del CCP1.

MÓDULO CCP1

Instituto Universitario de Tecnología para la Informática – Iutepi

Una parte central de este circuito es un registro CCPR1 de 16 bits que consiste en registros CCPR1L y CCOR1H. Se utiliza para capturar y comparar sus valores con los números almacenados en el registro del temporizador Timer1 (TMR1H y TMR1L).

CCP1 EN MODO DE CAPTURA

En este modo, el registro del temporizador Timer1 (que consiste en los TMR1H y TMR1L) se copia al registro CCP1 (que consiste en los CCPR1H y CCPR1L) en las siguientes situaciones:

Cada flanco ascendente (1 -> 0) en el pin RC2/CCP;

Cada flanco descendente (0 -> 1) en el pin RC2/CCP1;

Cada cuarto flanco ascendente (0 -> 1) en el pin RC2/CCP1; y

Cada decimosexto flanco descendente (0 -> 1) en el pin RC2/CCP1.

Una combinación de cuatro bits (CCP1M3 - CCP1M0) del registro de control determina cuál de estos eventos causará transmisión de dato de 16 bits. Además, se deben cumplir los siguientes requisitos::

El pin RC2/CCP1 debe estar configurado como entrada; y

El Timer1 debe funcionar como temporizador o contador síncrono.

Autoevaluación

- ¿Qué es un módulo Analógico- Digital
- ¿Qué utilidad representa un módulo A-D en un circuito con PIC?
- Explique que es un módulo de interrupciones
- ¿Qué es un módulo CCP Y Modo de captura?
- Realice un circuito de un semáforo simple con un PIC 16F84.

Instituto Universitario de Tecnología para la Informática – Iutepi

Conclusión

Vivimos unos tiempos con un gran desarrollo tecnológico. Estamos rodeados de todo tipo de aparatos que, no hace tanto parecían inalcanzables: ordenadores, teléfonos móviles de altas prestaciones, cámaras de vídeo y fotografía, reproductores, juguetes, máquinas, automatismos de todo tipo y un larguísimo etcétera.

Todo ello ha sido posible gracias a la evolución, entre otras áreas, de la electrónica y de la informática. Dentro del mundo de la electrónica, la aparición de un componente muy especial, supuso el empuje definitivo. Estamos hablando del "microcontrolador".

Conscientes de la enorme importancia que tienen los microcontroladores en nuestra vida diaria, especialmente los denominados "PIC", hemos querido incluir en la oferta formativa del Campus Tecnológico Virtual este ebook dirigido a los estudiantes que cursan la materia Microprocesadores II y de esta forma sirva de material complementario, para así comprender más a cabalidad los tópicos que el programa estipula.

Bibliografía

1. Microcontroladores PIC: sistema integrado para el autoaprendizaje. MARCOMBO,

EDICIONES TECNICAS 2007, MARCOMBO S.A. Enrique Mandado Pérez, Luis Menéndez Fuertes.

2. Arquitectura y programación de Microcontroladores Juan Manuel Orduña Huertas, Vicente Arnau Llombar. Universidad de Valencia 1996.
3. Microcontroladores: fundamentos y aplicaciones con PIC. Ramón Pallás Areny. 3Q editorial.
4. Principios de los microcontroladores Norberto Malpica Dpto. de Tecnología Electrónica universidad Central de Venezuela.
5. www.aquihayapuntes.co
6. Getting Started with Arduino. Massimo Banzi, 2da Ed O´REILLY.