UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ENTRENAMIENTO PARA MICROCONTROLADOR PIC16F84 Y PRÁCTICAS

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO MECÁNICO ELECTRICISTA

PRESENTA:

SÁNCHEZ FLORES, HUGO

ASESOR: VILLAFUERTE GARCÍA, ALEJANDRO

MÉXICO, D. F. 2001

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Mi agradecimiento:

En primer lugar a mis padres, porque sin ellos no estaría yo aquí ni podría haber logrado llegar hasta este punto de mi carrera ni tener una razón para seguir adelante.

A mis hermanos, que han estado conmigo en cada paso que he dado y que han influido, presionado y acatarrado para la realización de este trabajo.

A Hugo y Blanca (es curioso, ya no puede uno hablar de uno sin mencionar al otro) quienes, hoy por hoy, son los mejores amigos que pueda alguien tener, y que sin su apoyo y entusiasmo, este trabajo no llevaría hoy mi nombre.

A todos los precursores del hoy desaparecido Cubículo de Investigación y Desarrollo Tecnológico (Asesoría de Proyectos): Toño, Pancho, Silvia, Aarón, Enrique, Irma, Bogart y demás instructores de laboratorio de quienes aprendí más en un semestre de convivencia que en un año de clases. Gracias por abrirme las puertas al conocimiento y la experiencia que da el ayudar a un compañero sin cobrar ni un centavo.

A Rafael y Angélica, porque nadie me conoce como ustedes y por estar presentes conmigo aunque físicamente estemos a kilómetros de distancia uno del otro. Mi amistad y lealtad por siempre.

A mi nena Danis, por estar conmigo desde los últimos años de la carrera hasta la fecha, por apoyarme, aguantarme y aceptarme con todos mis defectos haciendo de mi una mejor persona.

Al Ingeniero Eleazar M. Pineda Díaz, quien como asesor de esta tesis todas sus valiosas propuestas, observaciones y correcciones nos orientaron a presentar un buen trabajo.

A la Universidad Nacional Autónoma de México que significa para mí más de lo que pudiera yo expresar con palabras y que está más allá de lo que dicen en las noticias, que para los verdaderos universitarios se convierte en parte de nuestra alma.

A todos los compañeros de la carrera quienes como amigos, conocidos, alumnos o simples copiones han influido de manera positiva en mi desarrollo profesional.

ALEJANDRO

29/7/03

A Alejandro (✞)

Quisiera verte de frente y solo puedo hacerlo en mis recuerdos, también quisiera decirte tantas cosas y solo me queda hacerlo en mis sueños, sin embargo quiero que este trabajo sea parte de un tributo a Ti, a quien acaeció dejando un enorme vacío y tristeza en nuestras vidas, pues desde que te fuiste, tengo un dolor en el corazón que me atormenta con cada recuerdo tuyo, más aún de saber que fui yo quien te llevó a cumplir tu cita con el destino. Quiero pedirte perdón por lo que dije sin pensar y ahora me arrepiento. Pero no creas que tú estás perdonado, pues también tienes una promesa conmigo sin cumplir. **Cuando llegue el momento estaré contigo, mientras tanto seguiré esperando**...

A mi Morena

Quiero dedicarte este trabajo como muestra de cariño y agradecimiento. Has sido una gran motivación, pues sin tus palabras de aliento en los momentos de flaqueza y desesperación estoy seguro de haber abandonado la misión. Gracias por ser todo lo que necesito, y aceptarme con todos mis defectos y aún así escogerme para emprender el gran proyecto de la vida como tu compañero; quiero también que sepas que por siempre seré tu amigo y en mí encontrarás un hombro en donde enjugar tus lágrimas, y una fuente infinita de esfuerzos por hacerte sonreír. oh! si, oh! si.

Gracias por conferirme tu tranquilidad, espero no defraudarte y el destino me preste suficiente tiempo, pero si tengo que probar el sabor a vencido, ten miedo de Mayo y **ten miedo de mí,** porque te estaré observando. Y sin ser fatalista, espero ser lo que tu necesitas, y así ambos iniciar la construcción de un futuro prometedor y lleno de cariño. Sin embargo también quiero que sepas que habrá días que...¡¡¡te haré la vida imposible!!! Ja!, no es cierto, pero lo que si es cierto es que tendremos días malos y sinsabores, que espero que tomados de la mano sigamos adelante, pues tú bien sabes que haré de mi un refugio.

A mis padres

Por que sin su apoyo y confianza, no hubiera logrado completar este trabajo, por los valores que me inculcaron para hacerme crecer como profesionista y ser humano.

A ti Mamá porque siempre has tenido los brazos abiertos cuando me he sentido desolado y abatido; por tus horas de desvelo cuando hemos estado enfermos y ser vigía de nuestro bienestar; por tu comprensión y respeto hacia mis decisiones que aunque erróneas son parte de mi crecer; por que cuando te miro a los ojos me encuentro conmigo mismo; porque con todos mis muchos defectos aprecias más mis contadas virtudes; porque me has enseñado que el llorar no es sólo para las mujeres, sino para el ser humano. Quiero que sepas que todos esos esfuerzos los he valorado, y que **te agradezco infinitamente y de todo corazón todo lo que has hecho por mí**.

A ti Papá porque has sido un incomparable amigo y maestro, que desde siempre has estado junto a nosotros para guiarnos y apoyarnos; porque con tus enseñanzas y tu mano firme y cariñosa nos has orientado sobre el mejor camino a seguir; porque con tus palabras de aliento siempre me he sentido reconfortado y así nunca me he sentido solo; porque cuando he fracasado has estado silenciosamente a mi lado, y con tu mano me has hecho levantar la frente; porque me has enseñado a valorar las cosas y a trabajar por ellas; por ser mi maestro de tiempo completo del cual nunca he dejado de aprender; porque con tus manos has construido nuestro hogar y **has cargado en tus hombros con la gran responsabilidad de ser un Padre; con admiración, cariño y respeto, gracias.**

A los dos, porque con cariño han sido el sostén de nuestra familia; porque a pesar de las desgracias han logrado mantenerse y mantenernos unidos, haciendo de la nuestra una gran familia.

A mis hermanos

A esos mafiosos, quiero decirles a todos y cada uno de ellos (Irma, Carmen, Enrique, Alejandro[†] y Alma) **gracias**, aunque no muchas, pues a ustedes les debo mis traumas, pues con una infancia llena de golpes y gandalleces de su parte no hay mucho que agradecer, ja!. Ya en serio, dicen que uno no puede escoger a la familia, pero yo ahora que los conozco, los volvería a escoger una y otra vez, hasta eso ya ven que no les guardo mucho rencor. Irma, gracias por estar al pendiente de todos nosotros, y hacer tuyos nuestros problemas, por eso te has ganado a pulso el puesto de **la mayor**, ¿y que crees? ya te vas a deshacer de esa esclavita; Carmen, mi querida **pelangocha**, espero nos queden muchos desayunos de peleas sin tregua ni descanso, quiero que sepas que he encontrado en ti a una gran amiga y confidente, gracias por abrirle las puertas a Blanca, eres una carnala a todo dar; Enrique **gracias por haber regresado a casa**, donde siempre y para siempre serás bienvenido, gracias por tu apoyo y no dejarme solo, pues ya tuve suficiente con la pérdida de un hermano; Alma, si bien me he encargado de hacerte la vida imposible, pero eso fue encargo de mis papás no creas que ha sido algo personal, quiero pedirte disculpas si a momentos se me pasó la mano, y sábete que cuentas conmigo.

A mi familia (anexados e Indexados)

A Alfredo, Meli, Alfredo Jr., Julieta, Rodrigo, Enrique Jr., Alberto, Hugo Alejandro, Luis Enrique, César Ricardo, Abigail, y a mi tío Aldo esto es una dedicatoria a todos ustedes, Alfredo Padre, gracias especialmente a ti pues has sido portador de una parte de la armonía de la familia, espero hayas encontrado con nosotros un nuevo hogar.

A mis amigos

Alets y Danis

Alets; gracias porque a pesar de nuestras diferencias has sabido tenerme paciencia, y demostrarme que eres un fenomenoide sin igual, además claro de ser un gran amigo; porque los momentos buenos y malos que pasamos desarrollando el trabajo no los voy a olvidar nunca. A los dos porque nuestra amistad ha perdurado espero que nuestras múltiples aventuras no acaben y no se olviden de Veracruz, ni de Oaxtepec, y mucho menos se vayan a olvidar de la historia del cacahuate, ni de la hamaca, ni de la toma de agua, y las que nos falten.

Bogart e Irma

Gracias por estar siempre con nosotros, que aunque no han estado físicamente, no saben como ni cuanto nos hemos reído de ustedes, por ser motivación para concluir este trabajo, pues ¿cómo va a ser posible que los que no saben ni cómo terminaron la carrera ya estén titulados y nosotros no?, por los gratos momentos que hemos pasado juntos, claro con Pueblita siempre presente.

Verónica

Gracias por estar conmigo a pesar de los años (que han pasado por los demás claro, porque tú y yo siempre jóvenes), por tenerme siempre presente y estar al pendiente de mí, quiero que cuentes conmigo aún en el tiempo y la distancia.

A los Padres de mi nena

José Antonio y Celia, quienes me abrieron las puertas de su hogar y me dieron un lugar en su familia sin reservas, gracias por confiarme el cuidado de su hija, tengan por seguro que no los defraudaré.

A todos mis compañeros y amigos, aún si no los he mencionado, gracias por la oportunidad de conocerlos y aprender de ustedes.

Al Ing. Antonio Rivera Z.

Gracias Toño por el apoyo incondicional.

A la Universidad Nacional Autónoma de México

Por ser la institución que por excelencia se mantiene por encima de cualquier acto político, creando profesionistas con un gran valor intelectual y espiritual, valores que conocemos quienes nos sabemos unamitas y portamos en alto y con orgullo el estandarte de la Universidad.

A sus profesores, quienes nos legaron sus conocimientos, principalmente al Ing. Méndez † (el padre de toda I.M.E.) y al Ing. Daniel Aldama por las oportunidades dadas.

Al Ing. Eleazar M. Pineda Díaz

Por la asesoría y paciencia en el desarrollo y conformado de este trabajo.

A la empresa Microchip Technology

Que a través del Ing. Jim Lyons y el Ing. Eugenio Manrique, representantes para Latino América y México respectivamente, donó equipo y componentes empleados en el desarrollo de este trabajo.

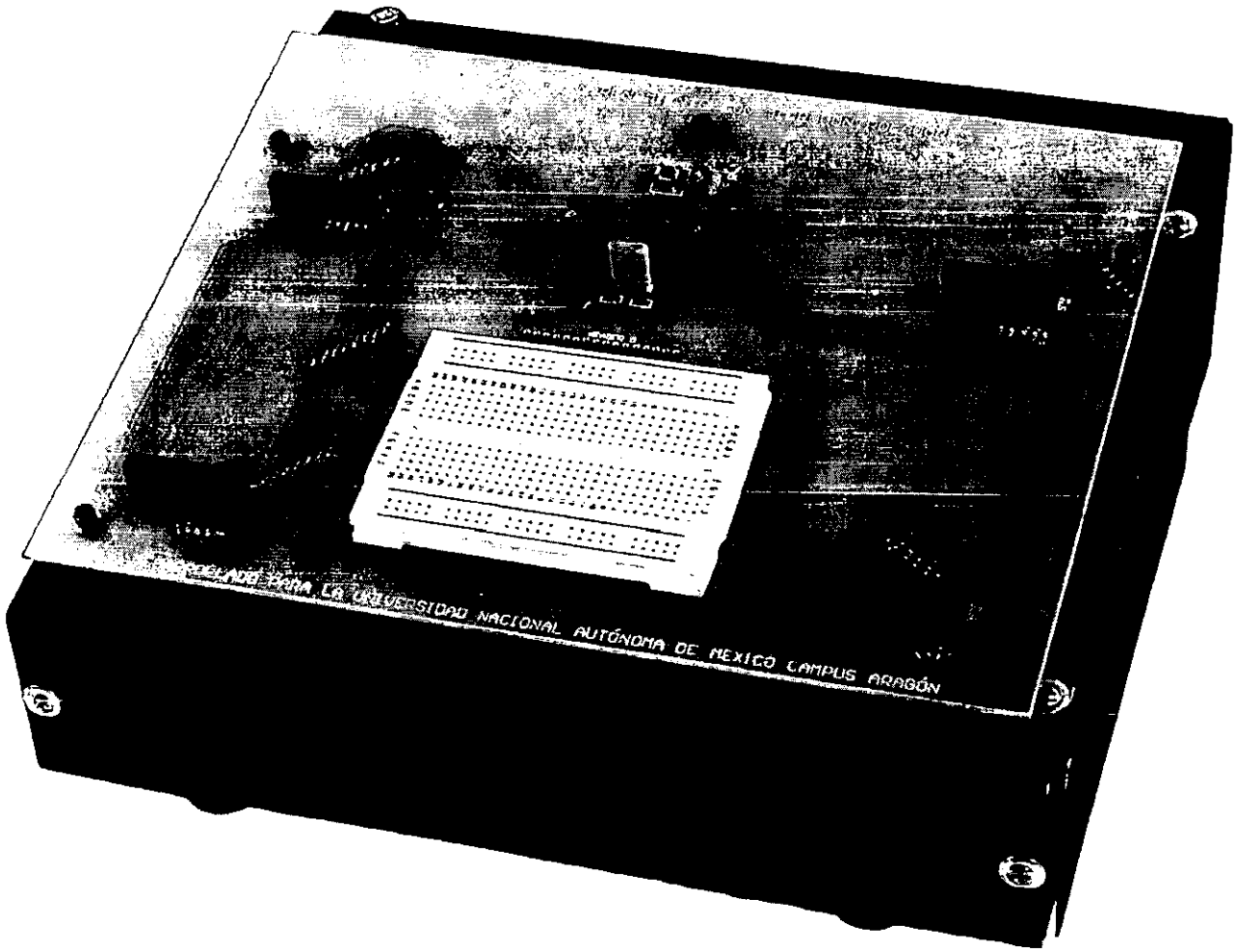
HUGO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ENTRENAMIENTO PARA MICROCONTROLADOR PIC16F84 Y PRÁCTICAS

POR:

HUGO SANCHEZ FLORES

ALEJANDRO VILLAFUERTE GARCÍA


DISEÑADO PARA LA UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO CAMPUS ARAGÓN

INDICE

INTRODUCCIÓN

El avance de la electrónica desde la invención del transistor ha traído consigo el desarrollo de nuevas tecnologías en dispositivos semiconductores, desde circuitos integrados de Baja Escala de Integración (LSI) como son las compuertas básicas hasta dispositivos de Ultra Alta Escala de Integración (UHSI) como son los microprocesadores que se encuentran en las Computadoras Personales. Estos últimos han sido incorporados a la vida cotidiana del hombre, sin embargo su misma arquitectura y elevado desempeño los hace costosos y complejos en tareas sencillas de automatización.

Esto creó la necesidad de desarrollar dispositivos que se emplean en funciones específicas, que sean rápidos y económicos. La respuesta a esta necesidad fue el **microcontrolador** y es quizá, el componente electrónico más versátil que existe; ya que cada vez es más frecuente encontrar aparatos que los utilizan como elemento de control. Por ejemplo en las cafeteras eléctricas, los hornos de microondas, los televisores, las videograbadoras, teléfonos celulares, conmutadores telefónicos, sistemas de encendido de automóviles, controles de temperatura, por mencionar tan sólo algunos.

Sin embargo, existen muchas ramas en las que esta innovación tecnológica no ha sido aplicada, de ahí la importancia de conocer y aprender a manejar estos dispositivos. Un eficiente uso, manejo y aplicación de microcontroladores son herramientas necesarias para enfrentar el reto que impone la creciente necesidad de modernización tecnológica y para satisfacer a la industria de México, que requiere personal idóneo en el área de diseño y desarrollo.

Con el objetivo de crear bases didácticas en el manejo y empleo de microcontroladores, este trabajo es la propuesta de un sistema de entrenamiento para la Universidad Nacional Autónoma de México, en la que hemos elegido a una familia de microcontroladores: los microcontroladores PIC de **Microchip Technology Inc.** y específicamente el **PIC16F84**.

Para conformar el marco teórico de este trabajo, se estructuraron los dos primeros capítulos:

Capítulo 1, donde se estudian las principales diferencias entre los microprocesadores y los microcontroladores, como son: arquitectura, manejo de instrucciones, ciclos de operación, organización y manejo de memoria. En este mismo capitulo se describen las características comunes que existen entre las gamas que componen a la familia de microcontroladores PIC.

El capítulo 2 contiene la información de la estructura interna del microcontrolador PIC16F84, la cual está formada por: registros de configuración, registros de control, mapas de memoria, temporizadores y puertos. Para esto nos apoyamos en esquemas descriptivos y útiles para la fácil comprensión de la operación interna del microcontrolador, pretendiéndose con esto el mejor aprovechamiento de los recursos con que cuenta este microcontrolador para su empleo en una aplicación específica.

El marco práctico de la tesis, está formado por los últimos tres capítulos, que son:

Capítulo 3, proporciona toda la información necesaria de forma concisa y detallada, de la implementación del sistema de entrenamiento, es decir, los diagramas esquemáticos y el funcionamiento de los elementos y dispositivos que componen los módulos del sistema.

En el capítulo 4 se muestran las herramientas necesarias a nivel software, como son: editor de texto, ensamblador y grabador; que nos darán apoyo para la realización de proyectos y prácticas propuestas.

El capítulo 5 contiene el manual de prácticas con algunas aplicaciones del PIC, que en conjunto con el sistema de entrenamiento constituye la mejor manera de adquirir los conocimientos y la habilidades para el manejo, operación y desarrollo de proyectos con microcontroladores PIC.

Así mismo se incluyen los Apéndices A, B y C con las especificaciones ópticas, eléctricas y su tabla de caracteres de la pantalla de cristal líquido (LCD), también las especificaciones y características eléctricas del microcontrolador PIC16X84 y de la interfaz de comunicación MAX 232 respectivamente.

CAPITULO 1

CONCEPTOS GENERALES

1.1. Microcontroladores.

El desarrollo de la electrónica con cada nuevo dispositivo de estado sólido trae consigo, técnicas de diseño diferentes, que por lo general son más simples. Por ejemplo, en los años 60, para construir un reloj con indicación digital, se necesitaba acoplar una cantidad grande de circuitos lógicos como: contadores, divisores, decodificadores y redes combinatorias. Al mismo tiempo, el diseñador debía poseer conocimientos muy claros sobre cada uno de los elementos, para realizar con éxito la integración.

A partir de 1970, el panorama digital cambió radicalmente cuando apareció en la mesa de los diseñadores un nuevo supercomponente: *EL MICROPROCESADOR*. Consolidadas las técnicas digitales se creó la necesidad de profundizar en el estudio de las estructuras de los microprocesadores, memorias, tecnologías de integración, conjunto de instrucciones, programación en lenguaje de máquina y adaptación de periféricos. Es la época de oro del 8080, el Z-80, el 6809, el 6502 entre otros, utilizados como circuitos centrales en las aplicaciones de control y que ahora forman parte de un museo de circuitos electrónicos.

Aproximadamente en 1980, los fabricantes de circuitos integrados iniciaron la difusión de un nuevo circuito para control, medición e instrumentación, al que llamaron *microcomputador en un solo chip* o bien *microcontrolador*; es decir que un *microcontrolador es un circuito integrado de alta escala de integración que contiene toda la estructura* (arquitectura) *de un microcomputador*, o sea, CPU (Central Process Unit - Unidad Central Proceso), RAM (Random Access Memory - Memoria de Acceso Aleatorio), ROM (Read Only Memory – Memoria de Sólo Lectura) y circuitos de I/O (Input/Output – Entrada/Salida), todo programable y alojado en un solo bloque de silicio, cerámica y materiales conductores.

El microcontrolador es un computador dedicado porque en su memoria solo reside un programa destinado a gobernar una aplicación determinada; sus líneas de I/O soportan la conexión de los sensores y actuadores del dispositivo a controlar y todos los recursos complementarios disponibles, tienen como única finalidad atender sus requerimientos. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada.

1.1.1. Diferencia entre un microcontrolador y un microprocesador.

Se comprende mejor la diferencia entre ambos componentes, si se estudia el proceso de diseño a partir de un microprocesador, el cual involucra los siguientes pasos:

1.- Selección de los circuitos.
2.- Mapa de memoria y de I/O (Entradas/Salidas).
3.- Diseño del circuito decodificador de direcciones.
4.- Montaje del circuito y del programa en la ROM.

En esta clase de circuitos con microprocesador, es fundamental el diseño de la red lógica que decodifica el bus de direcciones, para asignarle a cada memoria o puerto de I/O su posición dentro del mapa de memoria, planificado en el diseño. En la figura 1.1 se presenta el diagrama a bloques de un sistema mínimo necesario en el diseño de control con microprocesador.

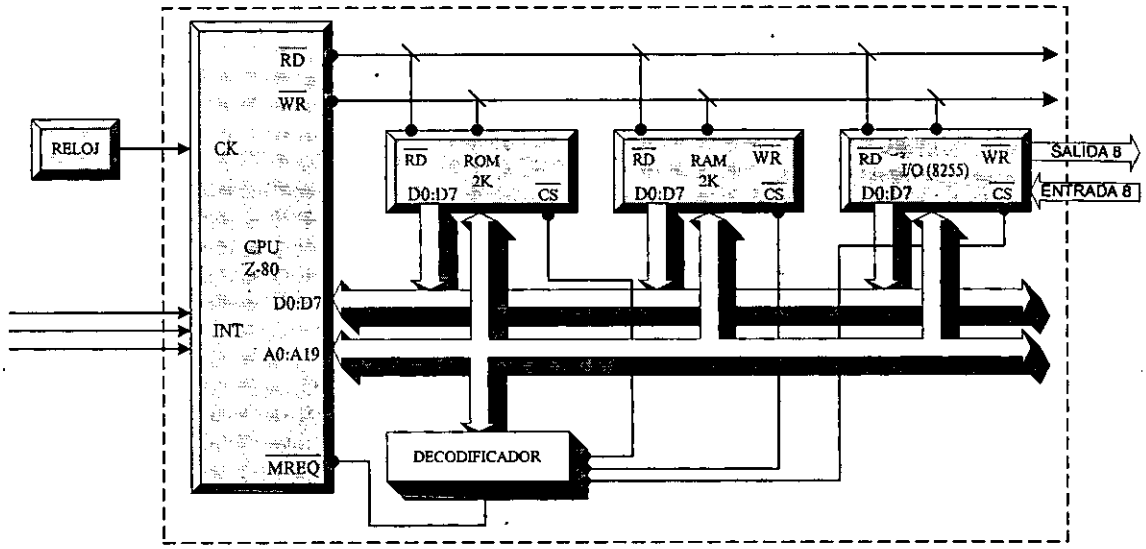


Figura 1.1 Microcomputador implementado con microprocesador Z-80

Observe la estructura o arquitectura resultante: está compuesta por las partes básicas de todo computador (CPU, RAM, ROM, I/O) interconectadas por tres tipos de buses[1] (direcciones, datos, control). En la memoria ROM se debe escribir el programa de control que le dará "vida" al computador. Observe también como los tres buses quedan a disposición del usuario para expansiones futuras de memoria y puertos de I/O. Hasta hace unos años, todo circuito de control, como el que se expone, se constituía utilizando un sistema mínimo de computador similar al de la figura 1.1. Ahora la estrategia es diferente.

Lo que muestra la figura 1.1 es el carácter constante y permanente de la estructura de un microcomputador en las aplicaciones de control, medición e instrumentación. No es difícil comprender, entonces, por qué los fabricantes de circuitos integrados decidieron producir un supercomponente que contiene todos los elementos de un computador en un solo circuito integrado ("chip"). En la figura 1.1 se demarca, con una línea punteada, los componentes básicos de un computador que integra un microcontrolador en un solo circuito. La idea de un microcomputador, para el diseñador de controles, se limita, ahora, a algo similar a lo que se ilustra en la figura 1.2. Se necesita solamente alimentación de corriente continua (Vcc), un cristal o una red RC externa para definir la frecuencia de operación, grabar el programa de control en la memoria ROM, y ya se tienen listos los puertos de entrada y salida para hacer conexión con el mundo exterior.

[1] Bus: se entiende como un grupo de líneas de conexión común entre dispositivos, ya sean internos o externos.

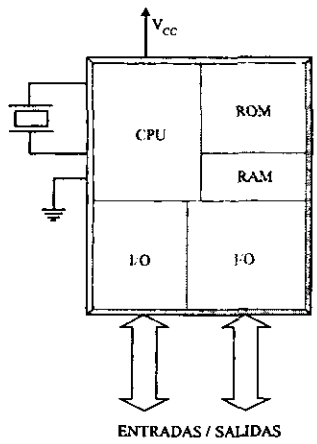


Figura 1.2 Idea general de un microcontrolador

¿Cuáles son las diferencias más notables entre un microcomputador realizado con microprocesador y uno con microcontrolador? Hay varias y las ventajas existen para las dos partes. Es mejor decir que cada uno tiene su ubicación especial dentro de las aplicaciones electrónicas.

- La CPU del microcontrolador es más simple, y sus instrucciones están orientadas, más que todo a la operación de cada uno de los bits de entrada y salida. Sin embargo, la estructura de la mayoría de los microprocesadores tiene su correspondiente versión de microcontrolador en el mercado.

- La memoria RAM de datos que ofrecen estos elementos es mínima. La razón es simple: las aplicaciones de control e instrumentación primitivas no necesitan almacenar grandes cantidades de información temporal.

- La memoria ROM del programa es limitada. Por lo general, no mayor a 4 Kilobytes.

- No es necesario diseñar circuitos decodificadores complejos porque el mapa de memoria y de puertos I/O, está implícito en el controlador. Por esta razón el circuito impreso de las aplicaciones es muy simple, en algunas ocasiones puede llegar a ser de una cara.

- La mayoría de los microcontroladores tienen dificultad para entregar al usuario los buses de direcciones, de datos y de control de la CPU, como lo hace fácilmente el sistema de la figura 1.1. Algunos controladores lo hacen a través de los puertos de entrada/salida, utilizando señales especiales de sincronización. Estos buses y señales se pueden emplear para implementar expansión de memoria RAM y ROM por fuera del microcontrolador.

- La velocidad de operación de los microcontroladores es más lenta que la que se puede lograr con los sistemas de microprocesadores. Sin embargo, hay noticias del desarrollo de circuitos controladores que funcionarán por encima de los 50 MHz.

- De manera similar a los sistemas utilizados con los microprocesadores para escribir, ensamblar y depurar programas en lenguaje de máquina, se requiere un sistema de desarrollo para cada familia de microcontroladores; éste esta compuesto por un paquete de software con editor, ensamblador y simulador de programas y al mismo tiempo se necesita un hardware para grabar la memoria EPROM del microcontrolador.

1.1.2. Recursos comunes de los microcontroladores.

Al estar todos los microcontroladores integrados en un chip, su estructura fundamental y sus características básicas son muy parecidas. Todos deben disponer de los bloques esenciales:

1. *Procesador.*
2. *Memoria no volátil (ROM).*
3. *Memoria temporal (RAM).*
4. *Líneas de I/O.*
5. *Oscilador de reloj.*

Arquitectura básica

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de Von Newmann, en el momento presente se impone la arquitectura Harvard.

La arquitectura de Von Newmann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accesa a través de un sistema de buses único (direcciones , datos y control), figura 1.3.

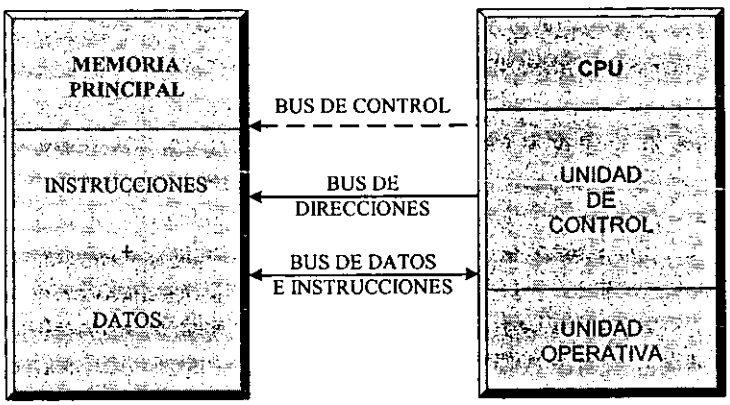


Figura 1.3 En la arquitectura de Von Neuman la memoria contiene indistintamente datos e instrucciones.

La arquitectura Harvard dispone de dos memorias independientes: una que contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias. Lo anterior se muestra en la figura 1.4.

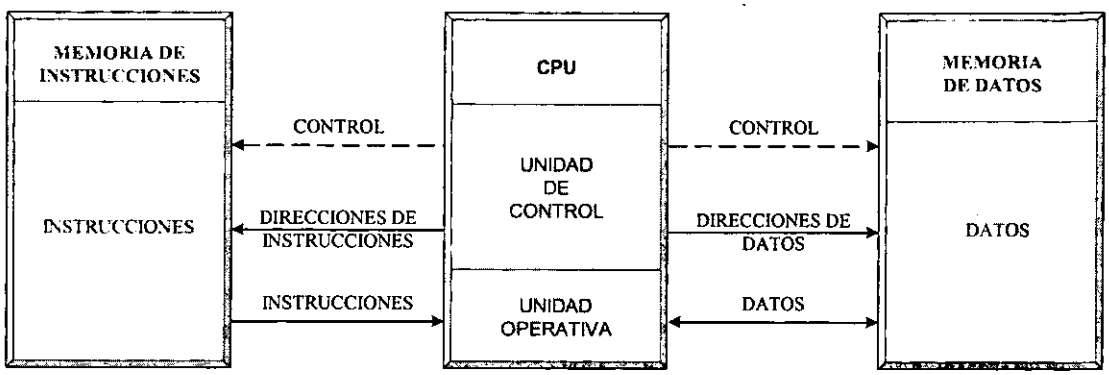


Figura 1.4 La arquitectura Harvard dispone de dos memorias independientes para datos y para instrucciones, permitiendo accesos simultáneos.

Los microcontroladores PIC, a los cuales pertenece la familia del microcontrolador seleccionado para este trabajo, utilizan este último tipo de arquitectura.

El procesador o CPU.

El procesador o CPU es el elemento más importante dentro del microcontrolador y determina sus principales características, tanto a nivel hardware como software.

Se encarga de direccionar la memoria de instrucciones, recibir el código de operación de la instrucción en curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado.

Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales:

CISC. Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Complex Instruction Set Cumputer - Computadores de Juego de Instrucciones Complejo). Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos de reloj para su ejecución.

Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como subrutinas.

RISC. Tanto la industria de los computadores comerciales como la de los microcontroladores están decantándose hacia la filosofía RISC (Reduced Instruction Set Computer – Computador de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo.

Un circuito de tipo RISC aumenta el rendimiento del computador con el efecto del paralelismo implícito, que consiste en la **segmentación** del procesador (*pipe-line*), descomponiéndolo en etapas para poder procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez.

El alto rendimiento y elevada velocidad que alcanzan los modernos procesadores, como el que poseen los microcontroladores PIC, se debe a la conjunción de tres técnicas:

- ***Arquitectura Harvard.***
- ***Arquitectura RISC.***
- ***Segmentación.***

SISC. En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido es específico, o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Specific Instruction Set Computer – Computador de Juego de Instrucciones Específico).

Memoria no volátil (ROM).

El microcontrolador está diseñado para que en su memoria de programa se almacenen todas las instrucciones del programa de control. No hay posibilidad de utilizar memorias externas de ampliación.

Como el programa a ejecutar siempre es el mismo, debe estar grabado de forma permanente. Los tipos de memoria adecuados para soportar esta función admiten 5 versiones diferentes:

ROM con mascara. Es una memoria no volátil de solo lectura cuyo contenido se graba durante la fabricación del chip. El elevado costo del diseño de la mascara solo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

OTP (*One Time Programmable*). El microcontrolador contiene una memoria no volátil de solo lectura Programable Una sola Vez por el usuario. Es el usuario quien puede escribir el programa en el circuito mediante un sencillo grabador controlado por un programa desde una PC.

La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien en la construcción de prototipos y series muy pequeñas.

EPROM *(Erasable Programmable Read Only Memory* – Memoria de Solo Lectura Borrable y Programable) Los microcontroladores que disponen de memoria EPROM pueden borrarse y grabarse muchas veces. La grabación se realiza como en el caso de los OTP con un grabador gobernado por una PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico.

EEPROM *(Electrical Erasable Programmable Read Only Memory* – Memoria de Solo Lectura Borrable y Programable Eléctricamente). Se trata de memorias de solo lectura, programables y borrables eléctricamente. Tanto la programación como el borrado, se realiza eléctricamente desde el propio grabador y bajo el control programado de una PC.

Los microcontroladores dotados de memoria EEPROM, una vez instalados en el circuito, pueden borrarse y grabarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

El número de veces que puede borrarse y grabarse una memoria EEPROM es finito, por lo que no es recomendable una reprogramación continua. Son muy idóneos en la enseñanza y en la ingeniería de diseño.

Se va extendiendo en los fabricantes la tendencia a incluir una pequeña zona de memoria EEPROM en los circuitos programable para guardar y modificar cómodamente una serie de parámetros que adecúan el dispositivo a las condiciones del entorno.

Este tipo de memoria es relativamente lenta.

FLASH. Se trata de un memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos potencia y es más pequeña.

A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM.

La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado.

Memoria temporal (RAM).

Los datos que manejan los programas varían continuamente, y esto exige que la memoria que los contiene deba ser de lectura y escritura, por lo que la memoria RAM estática (SRAM) es la más adecuada, aunque sea volátil.

Hay microcontroladores que disponen como memoria de datos, una de lectura y escritura no volátil, del tipo EEPROM. De esta forma, un corte en el suministro de la alimentación no ocasiona la pérdida de la información, que está disponible al reiniciarse el programa. El PIC16C84, el 16F83 y el 16F84 disponen de 64 bytes de memoria EEPROM para contener datos.

Puertos de I/O (Entrada/Salida)

La principal función de las terminales que posee la cápsula de un microcontrolador es soportar las líneas de I/O que comunican al computador interno con los periféricos exteriores. Según los controladores de periféricos que posea cada modelo de microcontrolador las líneas de I/O se destinan a proporcionar el soporte a las señales de entrada, salida y control; dichas señales manejan información en paralelo (transmisión y/o recepción de todos los bits de datos en un ciclo de reloj) y se agrupan en conjuntos de ocho que reciben el nombre de **puertos**. Hay modelos con líneas que soportan la comunicación en serie (transmisión y/o recepción de un bit de datos en un ciclo de reloj); otros disponen en conjuntos de líneas que implementan puertos de comunicación para diversos protocolos, como el USB (Universal Serial Bus – Bus Serial Universal).

El reloj principal

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

Generalmente, el circuito de reloj está incorporado en el microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C (Resistencia – Capacitor).

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva en consecuencia un incremento de consumo de energía.

1.1.3. Recursos auxiliares.

Según las aplicaciones a las que orienta el fabricante cada modelo de microcontrolador, incorpora una diversidad de complementos que refuerzan la potencia y la flexibilidad del dispositivo. Asimismo el objetivo del diseñador es encontrar el modelo óptimo que satisfaga todos los requerimientos de su aplicación con un costo mínimo.

Entre los recursos auxiliares se destacan los siguientes:

1. ***Temporizadores o Timers**. Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).*

2. ***Convertidores Analógico / Digital (ADC) y Digital / Analógicos (DAC)**. Se emplean para auxiliar al microcontrolador en el procesamiento de señales analógicas.*

3. ***Comparadores analógicos**. El microcontrolador dispone de un Amplificador Operacional para verificar el valor de una señal analógica.*

4. ***Protección ante fallos de alimentación**. Se trata de un circuito que **reinicia** al microcontrolador cuando el voltaje de alimentación (V_{DD}) es inferior a un nivel de voltaje mínimo.*

5. ***Estado de reposo o de bajo consumo**. Es un estado en el que el sistema queda congelado y el consumo de energía se reduce al mínimo.*

6. ***Generador de PWM** (Pulse Wide Modulation - **Modulación por Ancho de Pulso**). Son circuitos que proporcionan en su salida impulsos de ancho variable, que se ofrecen al exterior a través de las terminales de conexión.*

1.2. Familias de PIC´s

En 1965, la empresa **GI** creó una división de microelectrónica, **GI Microelectronics Division**, que comenzó su historia fabricando memorias EPROM y EEPROM, que conformaban las familias AY3-XXXX y AY5-XXXX. A principios de los años 70 diseñó el microprocesador de 16 bits CP1600, razonablemente bueno pero que no manejaba eficazmente las I/O (Entradas/Salidas). Para solventar este problema, en 1975 diseñó un chip destinado a controlar I/O: el **PIC (Peripheral Interface Controller)**. Se trataba de un controlador rápido pero limitado y con pocas instrucciones pues iba a trabajar en combinación con el CP1600.

La arquitectura del **PIC**, que se comercializó en 1975, era sustancialmente la misma que la de los actuales modelos PIC 16C5X. En aquel momento se fabricaba con tecnología NMOS (Lógica de transistores MOSFET[2] canal N) y el producto sólo se ofrecía con memoria ROM y con un pequeño pero robusto microcódigo.

La década de los 80 no fue buena para **GI**, que tuvo que reestructurar sus negocios, concentrando sus actividades en los semiconductores de potencia. La **GI Microelectronics Division** se convirtió en una empresa subsidiaria, llamada **GI Microelectronics Inc.** Finalmente, en 1985, la empresa fue vendida a un grupo de inversionistas con capital de riesgo, los cuales, tras analizar la situación, rebautizaron a la empresa con el nombre de **Arizona Microchip Technology** y orientaron su negocio a los **PIC**, las memorias EPROM paralelo y las EEPROM serie. Se comenzó rediseñando los **PIC**, que pasaron a fabricarse con tecnología CMOS (Complementary Metal-Oxide Semiconductor – Semiconductor Complementario de Metal-Oxido), surgiendo la familia de gama baja **PIC16C5X**, considerada como la clásica.

Una de las razones del éxito de los **PIC** se basa en su utilización. Cuando se aprende a manejar uno de ellos, conociendo su arquitectura y su repertorio de instrucciones, es muy fácil emplear otro modelo.

Hasta 1997 **Microchip** disponía de 52 versiones diferentes de PIC´s y cada año aumenta considerablemente su lista.

Microchip dispone de cuatro familias de microcontroladores de 8 bits para adaptarse a las necesidades de la mayoría de los clientes potenciales, dichas familias se describen a continuación.

1.2.1. Gama miniatura.

Se trata de un grupo de PIC de reciente aparición que ha acaparado la atención en el mercado. Su principal característica es su reducido tamaño, al disponer todos sus componentes en un encapsulado de 8 terminales. Se alimenta con un voltaje de corriente continua comprendido entre 2.5 V y 5.5 V, y consumen menos de 2 mA cuando trabajan a 5 V y 4 MHz. El formato de sus instrucciones puede ser de 12 o de 14 bits y su repertorio es de 33 o 35 instrucciones, respectivamente. En la figura 1.5 se muestra el diagrama de distribución de terminales de uno de los miembros de esta familia.

---

[2] MOSFET (Metal-Oxide Semiconductor Field Effect Transistor- Transistor de Efecto de Campo de Semiconductor de Metal-Oxido) que comúnmente se conocen como FET de compuerta aislada.

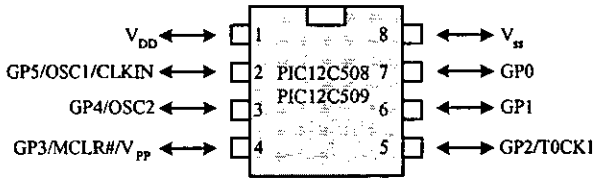


Figura 1.5 Diagrama de terminales de los PIC12CXXX de la gama miniatura.

Nota: aunque los PIC miniatura sólo tienen 8 terminales, pueden destinar hasta 6 como líneas de I/O para los periféricos porque disponen de un oscilador interno RC.

En la tabla 1.1 se presentan las principales características de los modelos de esta familia. Los modelos PIC12C5XX pertenecen a la gama baja, siendo el tamaño de las instrucciones de 12 bits; mientras que los PIC12C6XX son de la gama media y sus instrucciones tienen 14 bits; **pero se clasifican en esta gama debido al numero de terminales, aún cuando sus prestaciones son superiores**. Los modelos PIC12F6XX poseen memoria **Flash** para el programa y EEPROM para los datos.

MODELO	MEMORIA PROGRAMA	MEMORIA DATOS	FRECUENCIA MAXIMA	LINEAS I/O	ADC 8 BITS	TEMPORIZADORES	TERMINALES
PIC12C508	512 X 12	25 X 8	4 MHz	6		TMR0 + WDT	8
PIC12C509	1024 X 12	41 X 8	4 MHz	6		TMR0 + WDT	8
PIC12C670	512 X 14	80 X 8	4 MHz	6		TMR0 + WDT	8
PIC12C671	1024 X 14	128 X 8	4 MHz	6	2	TMR0 + WDT	8
PIC12C672	2048 X 14	128 X 8	4 MHz	6	4	TMR0 + WDT	8
PIC12C680	512 X 12 FLASH	80 X 8 16 X 8 EEPROM	4 MHz	6	4	TMR0 + WDT	8
PIC12C681	1024 X 14 FLASH	80 X 8 16 X 8 EEPROM	4 MHz	6		TMR0 + WDT	8

Tabla 1.1 Características de los modelos PIC12C(F)XXX de la gama miniatura.

1.2.2. Gama baja.

Se trata de una familia de PIC de recursos limitados, pero con una de las mejores relaciones *costo / prestaciones*. Sus versiones están en encapsulados de 18 (figura 1.6) hasta 28 terminales y pueden alimentarse a partir de una tensión de 2.5 V, lo que los hace ideales en las aplicaciones que funcionan con baterías. Tienen un repertorio de 33 instrucciones cuyo formato consta de 12 bits. No admiten ningún tipo de interrupción y la pila[3] solo dispone de dos niveles. En la tabla 1.2 se presentan las características mas importantes de los modelos que comprenden esta familia.

[3] Memorias **pila (LIFO)**: son memorias de acceso serie en las que la información que entra en la última operación de escritura es la que sale en la primera operación de lectura que se realice (Last In – First Out)

MODELO	MEMORIA PROGRAMA (x 12 BITS)		MEMORIA DATOS (BYTES)	FRECUENCIA MAXIMA	LINEAS I/O	TEMPORIZADORES	TERMINALES
	EPROM	ROM					
PIC16C52	384		25	4 MHz	4	TMR0 + WDT	18
PIC16C54	512		25	20 MHz	12	TMR0 + WDT	18
PIC16C54A	512		25	20 MHz	12	TMR0 + WDT	18
PIC16CR54A		512	25	20 MHz	12	TMR0 + WDT	18
PIC16C55	512		24	20 MHz	20	TMR0 + WDT	28
PIC16C56	1 K		25	20 MHz	12	TMR0 + WDT	18
PIC16C57	2 K		72	20 MHz	20	TMR0 + WDT	28
PIC16CR57B		2 K	72	20 MHz	20	TMR0 + WDT	28
PIC16C58A	2 K		73	20 MHz	12	TMR0 + WDT	18
PIC16CR58A		2K	73	20 MHz	12	TMR0 + WDT	18

Tabla 1.2 Principales características de los modelos de gama baja.

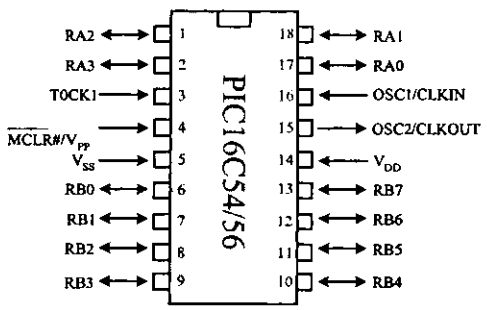


Figura 1.6 Diagrama de terminales del PIC de gama baja 16C54/56.

1.2.3. Gama media.

Es la gama más variada y completa de los PIC. Abarca modelos con encapsulados desde 18 terminales (figura 1.7) hasta 68, cubriendo varias opciones que integran abundantes periféricos. El repertorio de instrucciones es de 35 de 14 bits cada una y compatible con el de la gama baja. Sus distintos modelos contienen todos los recursos que se precisan en las aplicaciones de los microcontroladores de 8 bits. También disponen de interrupciones y una **pila** de 8 niveles que le permite el anidamiento de subrutinas.

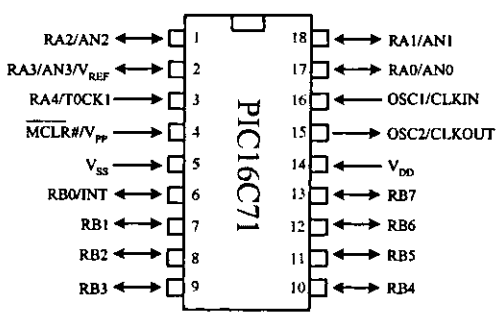


Figura 1.7 Diagrama de terminales del PIC16C71 de la gama media.

La gama media puede clasificarse en las siguientes subfamilias:

- Gama media estándar (PIC16C55X).
- Gama media con comparador analógico (PIC16C62X/64X/66X).
- Gama media con módulos de captura, modulación de ancho de pulso y puerto serie (PIC16C6X).
- Gama media con convertidor analógico-digital de 8 bits (PIC16C7X).
- Gama media con convertidor analógico-digital de precisión (PIC14000).
- Gama media con memoria **Flash** y EEPROM (PIC16X8X).
- Gama media con controlador para LCD (Liquid Crystal Display -- Pantalla de Cristal Líquido).

Encuadrado en la gama media también se halla la versión PIC14C000, que soporta el diseño de controladores inteligentes para cargadores de baterías, pilas pequeñas, fuentes de alimentación ininterrumpibles y cualquier sistema de adquisición y procesamiento de señales que requiera la gestión de energía de alimentación. Los PIC14C000 admiten cualquier tecnología de las baterías como *NiCd (Níquel – Cadmio), Pb (Plomo) y Zinc.* La tabla 1.3 describe las características principales de los modelos PIC16X8X.

MODELO	MEMORIA PROGRAMA	MEMORIA DATOS		REGISTROS ESPECIFICOS	TERMPORIZADOR ES	COMPARADORES ANALOGICOS	ADC 8 BITS	INTERRUPCIONES	LINEAS I/O	RANGO VOLTAJE	TERMINALES
		RAM	EEPROM								
PIC16C84	1 K x 14 EEPROM	36	64	11	TMR0 + WDT			4	13	2 – 6	18
PIC16F84	1 K x 14 FLASH	68	64	11	TMR0 + WDT			4	13	2 – 6	18
PIC16F83	512 x 14 FLASH	36	64	11	TMR0 + WDT			4	13	2 – 6	18
PIC16CR84	1 K x 14 ROM	68	64	11	TMR0 + WDT			4	13	2 – 6	18
PIC16CR83	512 x 14 ROM	36	64	11	TMR0 + WDT			4	13	2 – 6	18
PIC16C620	512 x 14 EEPROM	80	64	11	TMR0 + DWT	2		4	13	2 – 6	18
PIC16C71	1 K x 14	36	64	11	TMR0 + WDT		4	4	13	2 – 6	18

Tabla 1.3 Características relevantes de los modelos PIC16XXX de la gama media.

Nota: En la nomenclatura de la subfamilia 16XXX, C significa que la memoria de instrucciones es EEPROM; F significa que la memoria de instrucciones es del tipo Flash y CR significa que la memoria de instrucciones es ROM y se graba desde fábrica, y sólo se usa para grandes series.

1.2.4. Gama alta.

Se alcanzan las 58 instrucciones de 16 bits en el repertorio de instrucciones y sus modelos disponen de un sistema de gestión de interrupciones vectorizadas muy potente. También incluyen variados controladores de periféricos, puertos de comunicación serie y paralelo con elementos externos y un multiplicador hardware de gran velocidad.

Quizá la característica más destacable de los componentes de esta gama es su *arquitectura abierta*, que consiste en la posibilidad de ampliación del microcontrolador con elementos externos. Para este fin, las terminales tienen a disposición las líneas de los buses de datos, direcciones y control, a las que se conectan memorias o controladores de periféricos. Esta filosofía de construcción del sistema es la que se emplea en los microprocesadores y no suele ser una práctica habitual cuando se emplean microcontroladores.

En la tabla 1.4 se muestran las características más relevantes de los modelos de esta gama, que sólo se emplean en aplicaciones muy especiales con grandes requerimientos y en la figura 1.8 el diagrama de terminales de un miembro de la gama alta.

MODELO	MEMORIA PROGRAMA	MEMORIA DATOS (RAM)	REGISTROS ESPECIFICOS	TEMPORIZADORES	CAP	PWM	ADC 10 BITS	INTERRUPCIONES	LINEAS I/O	MULTIPLICADOR HARDWARE	TERMINALES
PIC17C42A	2 K x 16	232	48	4 + WDT	2	2		11	33	8 x 8	40 / 44
PIC17C43	4 K x 16	454	48	4 + WDT	2	2		11	33	8 x 8	40 / 44
PIC17C44	8 K x 16	454	48	4 + WDT	2	2		11	33	8 x 8	40 / 44
PIC17C752	8 K x 16	454	76	4 + WDT	4	3	12	18	50	8 x 8	64 / 68
PIC17C756	16 K x 16	902	76	4 + WDT	4	3	12	18	50	8 x 8	64 / 68

Tabla 1.4 características mas destacadas de los modelos PIC17CXXX de la gama alta.

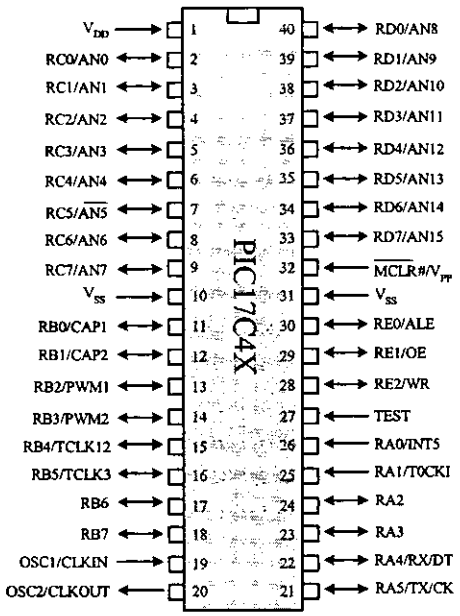


Figura 1.8 Diagrama de terminales del PIC17C4X de gama alta.

Con vistas al siglo XXI, **Microchip** ha lanzado la serie mejorada PIC18CXXX.

En resumen, en la tabla 1.5 se presentan los parámetros que mejor representan y diferencian las cuatro familias de PIC.

CARACTERISTICAS / FAMILIA	MEMORIA DE PROGRAMA	MEMORIA DE DATOS	TERMINALES I/O	VECTORES DE INTERRUPCION	TEMPORIZADORES	OTROS PERIFÉRICOS	RANGOS DE PRECIOS (USD)
GAMA MINIATURA PIC12CXXX	512 – 2K PALABRAS 12 – 14 BITS	25 – 128 BYTES	6	NINGUNO	1 DE 8 BITS	WDT, ADC	1 – 10
GAMA BAJA PIC16C5X	512 – 2K PALABRAS 12 BITS	24 – 73 BYTES	12 – 20	NINGUNO	1 DE 8 BITS CON DIVISOR DE FRECUENCIA	WDT	2.50 - 15
GAMA MEDIA PIC16CXX	512 – 4K PALABRAS 14 BITS	31 – 192 BYTES	13 – 33	1	DE 1 A 3 DE 8 BITS	WDT, COMPARADORES, ADC, SPI/I^2, CAPTURA/COMPARACION, PWM, SCI.	5 - 35
GAMA ALTA PIC17CXX	2K – 8K PALABRAS 16 BITS	232 – 454 BYTES	33	4	4 DE 8/16 BITS	WDT, COMPARADORES, ADC, SPI/I^2, CAPTURA/COMPARACION, PWM, SCI.	13 - 30

Tabla 1.5 Principales características que distinguen a las cuatro gamas de microcontroladores PIC.

1.3. Recursos comunes de los PIC´s.

A pesar de las diferencias entre familias de microcontroladores PIC, existen recursos en común entre ellos como son: operación de las frecuencias de trabajo y circuitos osciladores, dichos recursos se describen a continuación.

1.3.1. El reloj y frecuencia de trabajo.

La frecuencia de trabajo del microcontrolador es un parámetro fundamental al momento de establecer la velocidad en la ejecución de las instrucciones y el consumo de energía.

Cuando un PIC 16X8X funciona a 10 MHz, que es su máxima frecuencia, le corresponde un ciclo de instrucción de 400 ns, puesto que cada instrucción tarda en ejecutarse cuatro períodos de reloj, o sea, 4 * 100 ns = 400 ns. Todas las instrucciones del PIC se realizan en un **ciclo de instrucción**, menos las de salto que tardan el doble.

Los impulsos de reloj que entran por la terminal OSC1/CLKIN y se dividen por 4 internamente, dando lugar a las señales Q_1, Q_2, Q_3 y Q_4, mostradas en la figura 1.9.

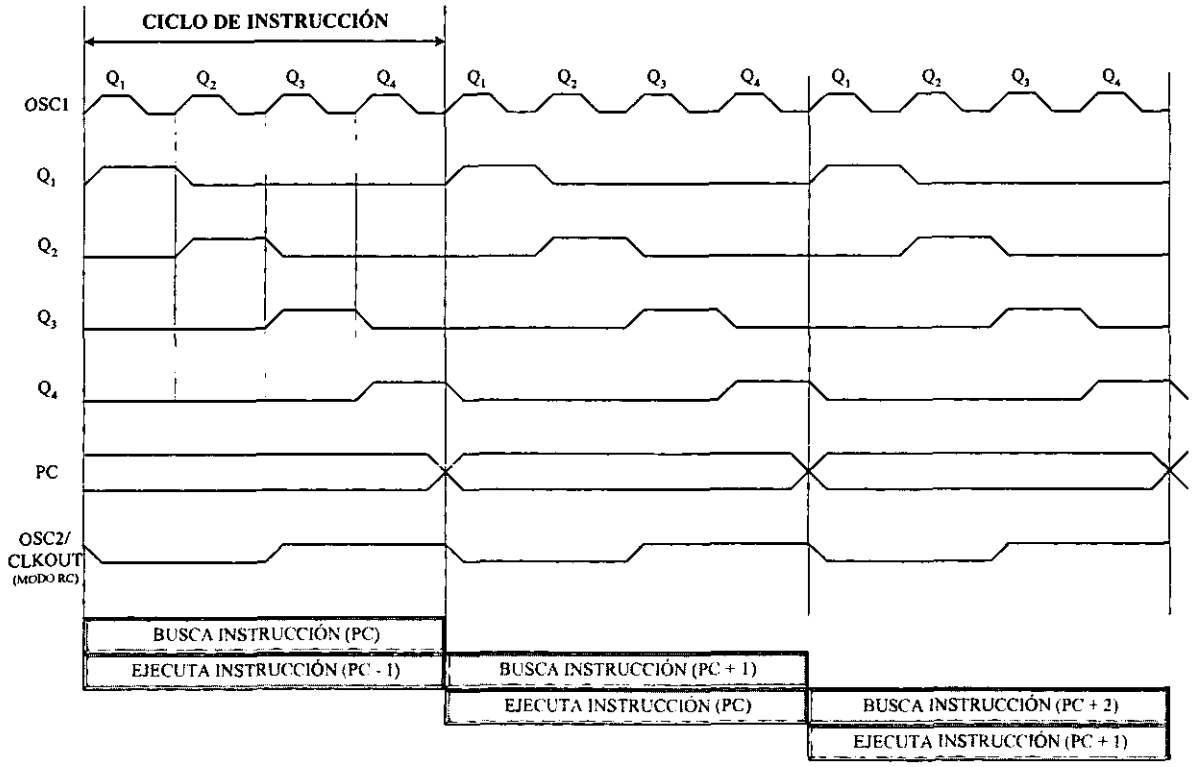


Figura 1.9 Cronograma de reloj y ciclo de instrucción.

Durante un ciclo de instrucción, que comprende las cuatro señales mencionadas, se desarrollan las siguientes operaciones:

Q_1: Durante este impulso se incrementa el Contador de Programa (PC).
Q_2: Durante este impulso se busca el código de la instrucción en la memoria del programa y se carga en el Registro de Instrucciones.
Q_3-Q_4: Durante la activación de estas dos señales se produce la decodificación y la ejecución de la instrucción.

En realidad cada instrucción conlleva dos ciclos de instrucción, el primero destinado a la **fase de búsqueda** y el otro a la **fase de ejecución**. Sin embargo, la estructura segmentada del procesador permite realizar simultáneamente la fase de ejecución de una instrucción y la de búsqueda de la siguiente. Cuando la instrucción ejecutada corresponde a un salto, no se conoce hasta completarla por lo que se sustituye la fase de búsqueda por una instrucción **NOP** (No Operar) mientras se ejecuta un salto. Esta característica se muestra gráficamente en la figura 1.10 y en ella se puede apreciar cómo las instrucciones de salto precisan un ciclo más.

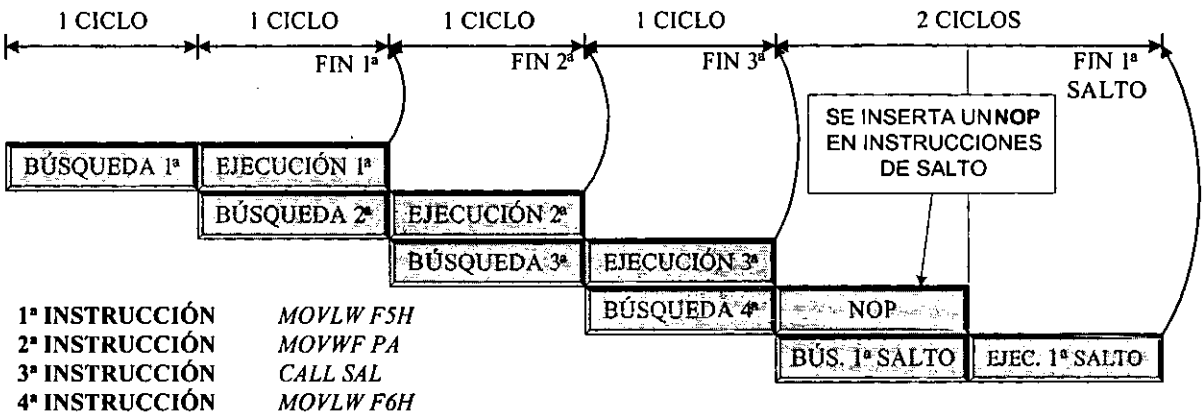


Figura 1.10 Fase de búsqueda y ejecución.

Para el funcionamiento del circuito de reloj interno se precisa colocar en el exterior una fuente de señal con una frecuencia fija disponiendo los microcontroladores PIC de dos terminales para soportar dicha señal:

OSC1/CLKIN: Es una terminal a la que se conecta la señal de entrada de la fuente externa de frecuencia, que puede estar implementada por un cristal de cuarzo, por un resonador cerámico o una red RC.

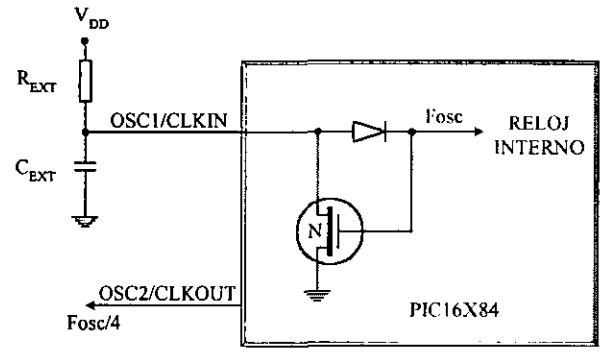
OSC2/CLKOUT: Se trata de una terminal de conexión de la salida de cristal externo.

Tipos de osciladores.

Los PIC admiten cuatro tipos de osciladores externos que le proporcionan la frecuencia de funcionamiento. El tipo empleado debe especificarse en dos bits (**FOSC1** y **FOSC2**) de la **Palabra de Configuración** que se analiza en el Capítulo 2.

*Oscilador tipo **RC.***

Se trata de un oscilador de bajo costo, formado por una simple resistencia (R_{EXT}) y un condensador (C_{EXT}). Proporciona una estabilidad mediana de la frecuencia, cuyo valor depende de los dos elementos de la red RC, la configuración de este oscilador se muestra en la figura 1.11.



F. Oscilación	R_{EXT}	C_{EXT}
625 KHz	10 K	20 pF
80 KHz	10 K	220 pF
80 KHz	10 K	0.1 pF

Figura 1.11 Esquema del oscilador tipo RC, se indican los valores de los elementos R_{EXT} y C_{EXT} para obtener algunas de las frecuencias de oscilación.

En el esta configuración en la terminal OSC2/CLKOUT se obtiene la cuarta parte de la frecuencia de oscilación, delimitando los ciclos de instrucción. En esta alternativa la empresa **Microchip** recomienda usar una R_{EXT} con un valor comprendido entre 5 kΩ y 100 kΩ y un C_{EXT} con más de 20 pF.

*Oscilador tipo **HS** (High – Speed, Alta Velocidad)*

Se trata de un oscilador que alcanza una alta velocidad comprendida entre 8 MHz y 20 MHz y está basado en un cristal de cuarzo o un resonador cerámico.

*Oscilador tipo **XT** (Xtal – Cristal)*

Es un oscilador de cristal o resonador para frecuencias estándar comprendidas entre 100 kHz y 4 MHz.

*Oscilador tipo **LP** (Low – Power, Baja Potencia)*

Es un oscilador de bajo consumo con cristal o resonador diseñado para trabajar en un rango de frecuencias comprendido entre 32 kHz y 200 kHz.

Para cualquiera de las versiones de osciladores HS, XT o LP el cristal de cuarzo o resonador cerámico se coloca entre las terminales OSC1 y OSC2, como se muestra en la figura 1.12, y en la tabla 1.6 se muestran; según el tipo de oscilador usado y el rango de frecuencia, los valores de los condensadores sugeridos para obtener algunas frecuencias de trabajo.

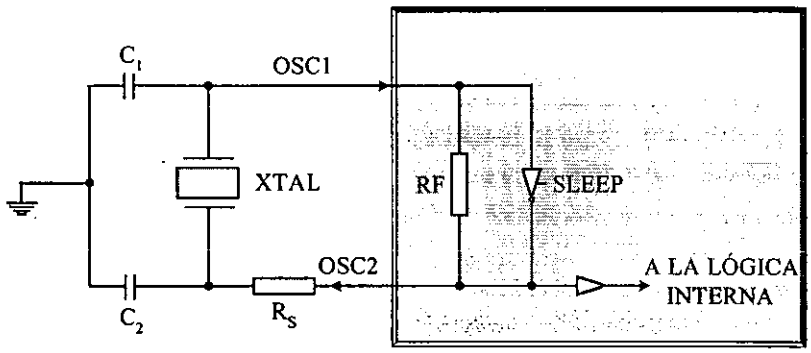


Figura 1.12 Configuración de oscilador externo.

La resistencia R_S no es necesaria en las versiones XT, pero sí en las versiones HS. Únicamente las características exactas del cuarzo permiten saber si es o no necesaria y cuál debe ser su valor.

TIPO DE OSCILADOR	FRECUENCIA DEL CRISTAL	RANGO DE C_1	RANGO DE C_2
LP	33 kHz	33 – 68 pF	33 – 68 pF
	200 kHz	15 – 33 pF	15 – 33 pF
XT	100 kHz	68 – 100 pF	68 – 100 pF
	2 MHz	10 – 22 pF	10 – 22 pF
	4 MHz	10 – 22 pF	10 – 22 pF
HS	8 MHz	22 – 47 pF	22 – 47 pF
	20 MHz	22 – 47 pF	22 – 47 pF

Tabla 1.6 Valores de C_1 y C_2 sugeridos para algunas frecuencias de trabajo.

1.3.2. Circuito de reset

Al igual que con los circuitos osciladores, todos los circuitos de las familias PIC adoptan el mismo método para reinicializar externamente el microcontrolador. Los PIC disponen de una terminal única de RESET denominada MCLR#, esta terminal presenta algunas particularidades para quien está habituado a los microcontroladores clásicos.

Estos dispositivos incorporan internamente una circuitería de RESET que entra en funcionamiento automáticamente cuando se conecta la alimentación. Si este procedimiento es suficiente (no existe necesidad de RESET externo manual, por ejemplo) y si la velocidad de crecimiento de la tensión de alimentación es lo bastante alta (típicamente superior a 0.05V/ms), no se necesita circuitería adicional. En la figura 1.13a se resume la conexión del circuito de RESET: la terminal MCLR# se conecta a la tensión de alimentación positiva V_{DD}.

Si esta velocidad de crecimiento de la tensión no se alcanza, o cuando se necesite un control de RESET externo, o incluso si utiliza un cristal de cuarzo de frecuencia relativamente baja y, por lo tanto, el tiempo que se tarda en entrar en oscilación es grande, debe implementarse el circuito de la figura 1.13b

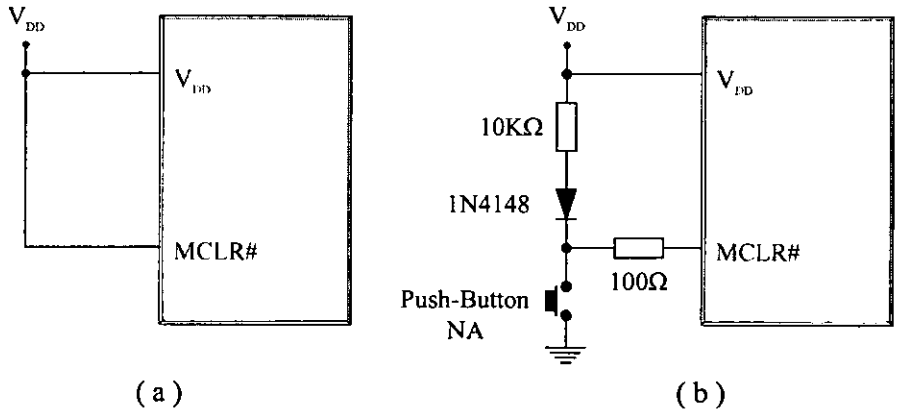


Figura 1.13 Opciones de conexión de la terminal MCLR

El diagrama mostrado en la figura 1.13b es un circuito con el que los usuarios de microcontroladores ya están familiarizados.

Por último, si la tensión de alimentación es susceptible de variar en proporciones que el buen funcionamiento del circuito pueda comprometerse, es aconsejable prever una circuitería de RESET capaz de activarse si la alimentación desciende por debajo de un determinado umbral, ya que esto la circuitería interna no lo hace. Microchip propone dos esquemas como los que se muestran en la figura 1.14.

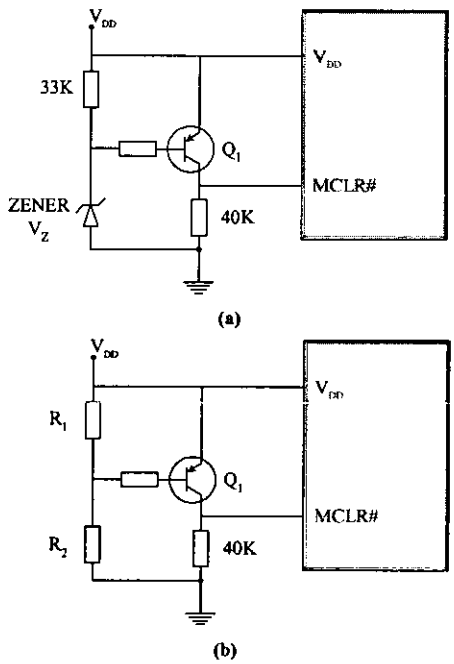


Figura 1.14 Circuitos externos de protección.

CAPITULO 2

EL PIC16F84

2.1. Descripción general.

El PIC16X84 pertenece a la familia de la gama media y dentro de ella es uno de los más pequeños, sólo tiene 18 terminales. Además es el que dispone de menos recursos. Pero se ha elegido a este PIC, en sus dos versiones C y F, porque cumple con los requisitos esenciales para **enseñar** a manejar los microcontroladores y comenzar a diseñar proyectos. ***Es práctico, es sencillo y es económico***, poniéndolo como el paradigma para su empleo en todos los pequeños proyectos que realizan los aficionados, los estudiantes y quienes prefieren progresar en sus conocimientos de lo fácil a lo difícil.

Como se mencionó, el éxito de los **PIC** se basa en su utilización y la semejanza que existe entre ellos, ya que cuando se aprende a manejar uno de ellos, conociendo su arquitectura y repertorio de instrucciones, es muy fácil emplear otro modelo.

Otra razón es debido al tipo de memoria de programa que posee. En el caso del PIC16C84 se trata de una EEPROM de 1 K palabras[4] de 14 bits cada una. El PIC16F84 tiene la misma capacidad de memoria de instrucciones, pero del tipo Flash. Ambos disponen de 64 bytes de EEPROM como memoria de datos auxiliar y opcional.

La memoria EEPROM y la Flash son eléctricamente grabables y borrables, lo que permite escribir y borrar el programa bajo prueba manteniendo el microcontrolador en el mismo zócalo y usando el mismo dispositivo para grabar y borrar. Esta característica supone una gran ventaja con la mayoría de los microcontroladores, que tienen como memoria de programa reescribible una tipo EPROM. Se graban eléctricamente, pero para borrarlas hay que someterlas durante cierto tiempo a rayos ultravioleta, lo que implica sacar del zócalo al circuito integrado y colocarlo en un borrador de EPROM.

Microchip ha introducido la memoria Flash porque tiene mejores posibilidades de aumentar su capacidad con relación a la EEPROM. También por su mayor velocidad y menor consumo.

Otra ventaja del PIC16X84 en cuanto a simplificar el proceso de escritura, borrado y reescritura de programas, tan necesario en la fase de diseño, es su sistema de grabación de datos, que se realiza en serie. Para escribir un programa en la memoria se manda la información en serie usando sólo dos de sus terminales: la RB6 para la señal de reloj y la RB7 para los bits de datos serie. La distribución de las terminales de este microcontrolador se muestran en la figura 2.1.

[4] 1 K palabras equivale a 1024 palabras, desde 000H hasta 3FFH.
H declara notación hexadecimal.

La terminal RB6 funciona como entrada de circuito de reloj y la terminal RB7 como entrada de datos durante el proceso de grabación de la memoria de programa.

En seguida se enlistan las características más relevantes del PIC16X84:

- Memoria de programa: 1 K x 14, EEPROM (PIC16C84) y Flash (PIC16F84).
- Memoria de datos: 36 bytes (PIC16C84) y 68 bytes (PIC16F84).
- Memoria de datos EEPROM: 64 bytes para ambos modelos.
- Pila: 8 niveles.
- Interrupciones: 4 tipos diferentes.
- Juego de instrucciones: 35.
- Encapsulado: plástico DIP (Dual In Package – Doble En Encapsulado) de 18 terminales.
- Frecuencia de trabajo: 10 MHz. Máxima.
- Temporizadores: TMR0 y WDT (Watch Dog Timer – Temporizador Perro Guardián).
- Líneas de I/O digitales: 13 (5 Puerto A y 8 Puerto B).
- Corriente máxima absorbida: 80 mA Puerto A y 150 mA Puerto B.
- Corriente máxima suministrada: 50 mA Puerto A y 100 mA Puerto B.
- Corriente máxima absorbida por línea: 25 mA.
- Corriente máxima suministrada por línea: 20 mA.
- Voltaje de alimentación (V_{DD}): de 2 a 6 VDC.
- Voltaje de grabación (V_{PP}): de 12 a 14 VDC.

2.2. Arquitectura interna.

Para lograr una compactación de código de operación optima y una velocidad superior a la de sus competidores los microcontroladores PIC incorporan en su procesador tres de las características más avanzadas en las grandes computadoras:

1. ***Procesador tipo RISC.***
2. ***Procesador segmentado.***
3. ***Arquitectura Harvard.***

Con la incorporación de estos recursos los PIC son capaces de ejecutar en un ciclo de instrucción todas las instrucciones, excepto las de salto que tardan el doble. Una condición imprescindible es la simetría y la ortogonalidad en el formato de las instrucciones, que en el caso de los PIC de la gama media tienen una longitud de 14 bits. De esta forma se consigue una compactación en el código del programa, para un PIC16F84 es 2.24 veces superior al de un 68HC05 funcionando a la misma frecuencia.

El juego de instrucciones se reduce a 35 y sus modos de direccionamiento se han simplificado al máximo.

Con la estructura segmentada se pueden realizar simultáneamente las dos fases en que se descompone cada instrucción, al mismo tiempo que se esta desarrollando la fase de ejecución de una instrucción se realiza la fase de búsqueda de la siguiente.

El aislamiento y las diferencias de los dos tipos de memoria permite que cada uno tenga la longitud y el tamaño más adecuados. De esta forma en el PIC16X84 la longitud de los datos es de un byte, mientras que la de las instrucciones es de 14 bits.

Otra característica relevante de los PIC es el manejo intensivo del banco de registros, los cuales participan de una manera muy activa en la ejecución de las instrucciones, como se muestra en la figura 2.2, la ALU (Arithmetic Logic Unit – Unidad Lógica Aritmética) efectúa sus operaciones lógico-aritméticas con dos operandos, uno que recibe desde el registro de trabajo W (Work), que hace las veces de acumulador en los procesadores convencionales, y otro que puede provenir de cualquier registro o del propio código de operación (OP) a través de un multiplexor[6]. El resultado de la operación puede almacenarse en cualquier registro o en W. Esta funcionalidad da un carácter completamente ortogonal a las instrucciones, lo que significa que pueden utilizar cualquier registro como operando fuente y destino. La memoria de datos RAM implementa en sus posiciones los registros específicos (SFR – Special Function Register) y los de propósito general (GPR – General Purpose Register).

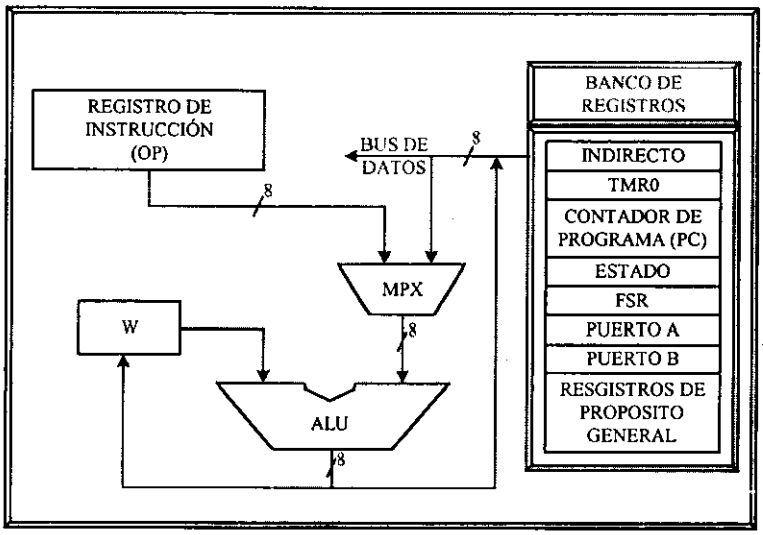


Figura 2.2. Dispositivos que proporcionan los operandos a la ALU.

La arquitectura interna del PIC16F84 se muestra en la figura 2.3 y consta de 7 bloques fundamentales que son:

1°. Memoria de programa EEPROM de 1 K x 14 bits ligado al Contador de Programa (PC) y a una pila de 8 niveles.
2°. Memoria de datos formada por dos áreas. Una RAM donde se alojan 22 registros de propósito específico (FSR) y 36 de propósito general (GPR), y otra del tipo EEPROM de 64 bytes.
3°. Bus de datos con la ALU de 8 bits y un registro de trabajo W del que normalmente recibe un operando y envía el resultado. El otro operando puede provenir del bus de datos o del propio código de la instrucción (literal).
4°. Recursos conectados al bus de datos, tales como puertos de I/O y temporizador TMR0.
5°. Base de tiempos y circuitos auxiliares.

[6] Multiplexor: Circuito combinacional que selecciona información binaria de una de varias líneas de entrada y la dirige a una sola línea de salida.

6°. Direccionamiento de la memoria RAM en el modo **directo** a través del Registro de Instrucciones e **indirecto** a través del registro FSR.
7°. Lógica de control y decodificación de instrucciones.
8°. Bus específico de 11 bits para las instrucciones *GOTO* y *CALL* (se analiza en la página 34).

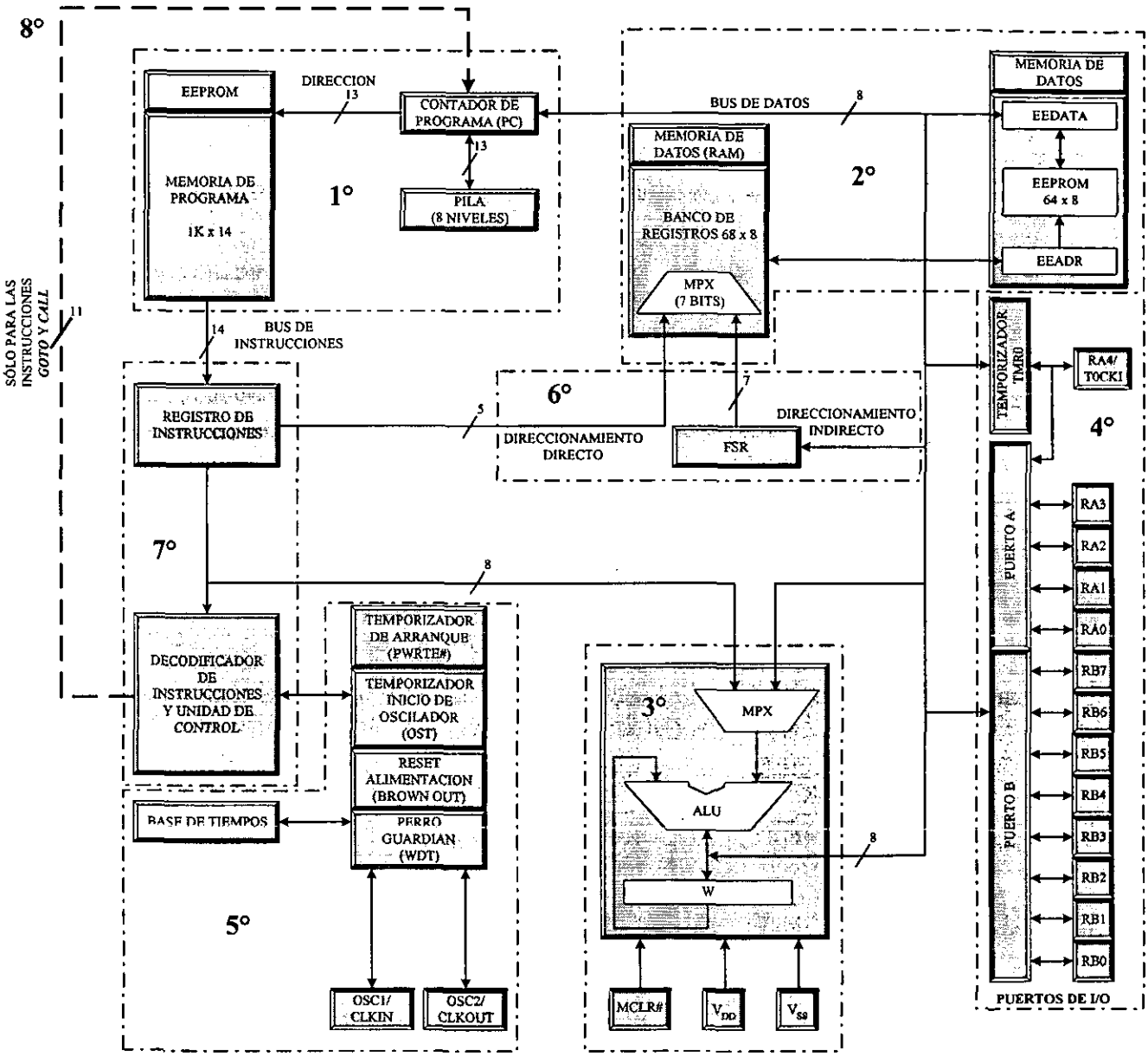


Figura 2.3 Arquitectura interna del PIC16F84.

Para analizar de forma global el funcionamiento del microcontrolador vamos a tomar como ejemplo la ejecución de una instrucción. El ciclo comienza con la *fase de búsqueda* que la inicia el Contador de Programa (PC) facilitando la dirección de la memoria de instrucciones donde se ubica. Su código binario de 14 bits se lee y se carga en el Registro de Instrucciones, desde donde se transfiere al Decodificador y a la Unidad de Control. A veces, dentro del código de la instrucción, existe el valor de un operando que se introduce a la ALU, o bien una dirección de la memoria de datos donde reside otro operando.

La ALU es la encargada de realizar la operación lógico-aritmética que indica la instrucción decodificada. Uno de los operandos lo recibe desde el registro W y el otro desde otro registro o de la propia instrucción.

Tanto el banco de Registros Específicos, en el que cada registro tiene una función concreta, como el de Registros de Propósito General residen en la RAM. La EEPROM de datos puede contener datos que no se desee perder al quitar la alimentación, pero su acceso está controlado por los registros EEDATA y EEADR.

Las operaciones de I/O con los periféricos la soportan los puertos A y B. Existe un temporizador (TMR0) que se encarga de las funciones de control de tiempos. Finalmente, hay unos circuitos auxiliares que dotan al procesador de posibilidades de seguridad, reducción del consumo de energía y de RESET.

2.2.1. Organización de la memoria.

La memoria del PIC16F84 se clasifica en memoria de programa y memoria de datos.

La memoria de datos se divide en RAM de propósito general, y los registros de función específica (SFR- Special Function Register). Y también contiene la memoria EEPROM de datos. Esta memoria está indirectamente mapeada, es decir, un indicador de dirección indirecta especifica la dirección de la memoria EEPROM de datos para Leer/Escribir. Los 64 bytes de la memoria EEPROM de datos tienen el rango de 00H-3FH.

Memoria de datos.

La memoria de datos del PIC16X84 dispone de dos zonas diferentes:

1*f* **Área de RAM estática o SRAM**, donde reside el banco de SFR´s y el banco de GPR´s. El primer banco tiene 24 posiciones de tamaño byte, aunque dos de ellas no son operativas, y el segundo 36 (68 en el PIC16F84).

2*f* **Área EEPROM** de 64 bytes donde, opcionalmente, se pueden almacenar datos que no se pierden al desconectar la alimentación.

La zona de memoria RAM se halla dividida en dos bancos (banco 0 y banco 1) de 128 bytes cada uno como se muestra en la figura 2.4.

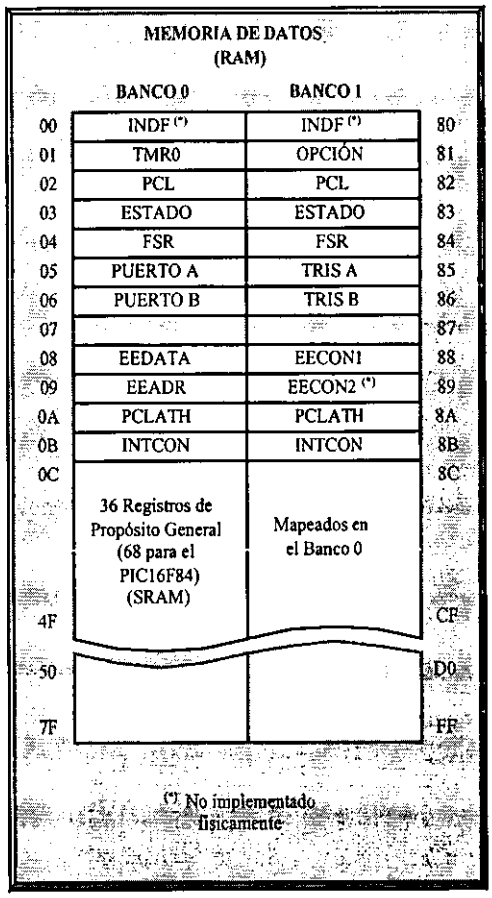


Figura 2.4 Distribución de los bancos de memoria.

En el PIC16C84 sólo se hallan implementadas físicamente las 48 primeras posiciones de cada banco, mientras que en PIC16F84 son 80, de las cuales las 12 primeras están reservadas a los Registros de Función Específica, que son los encargados del control del procesador y sus recursos. Algunos de estos registros se hallan repetidos en la misma dirección de los dos bancos, para simplificar su acceso (INDF, ESTADO, FSR, PCLATH e INTCON). La posición apuntada por la dirección 07H y la apuntada por la 87H no son operativas. Los 36 registros restantes de cada banco (68 para el PIC16F84) se destinan a los Registros de Propósito General y en realidad sólo son operativos los 36 (68) del banco 0 porque los del banco 1 se mapean sobre el banco 0, es decir, cuando se apunta a un registro del banco 1, se accesa al mismo del banco 0.

Direccionamiento de la memoria de datos.

La memoria de datos de la gama media se organiza en un máximo de 4 bancos, cada uno de los cuales puede constar de hasta 128 posiciones de tamaño byte. Para seleccionar el banco y la posición a accesar existen los siguientes dos modos de direccionamiento:

1° Direccionamiento directo.

Los 7 bits de menos peso del Código de Operación de la instrucción proporcionan la dirección de la posición de un banco. Los bits RP1 y RP0 del registro ESTADO <6:5>[7], seleccionan el banco, como se muestra en la figura 2.5.

[7] La nomenclatura <>, <:> es adoptada por el fabricante para denotar ya sea un bit o un rango de bits específicos.

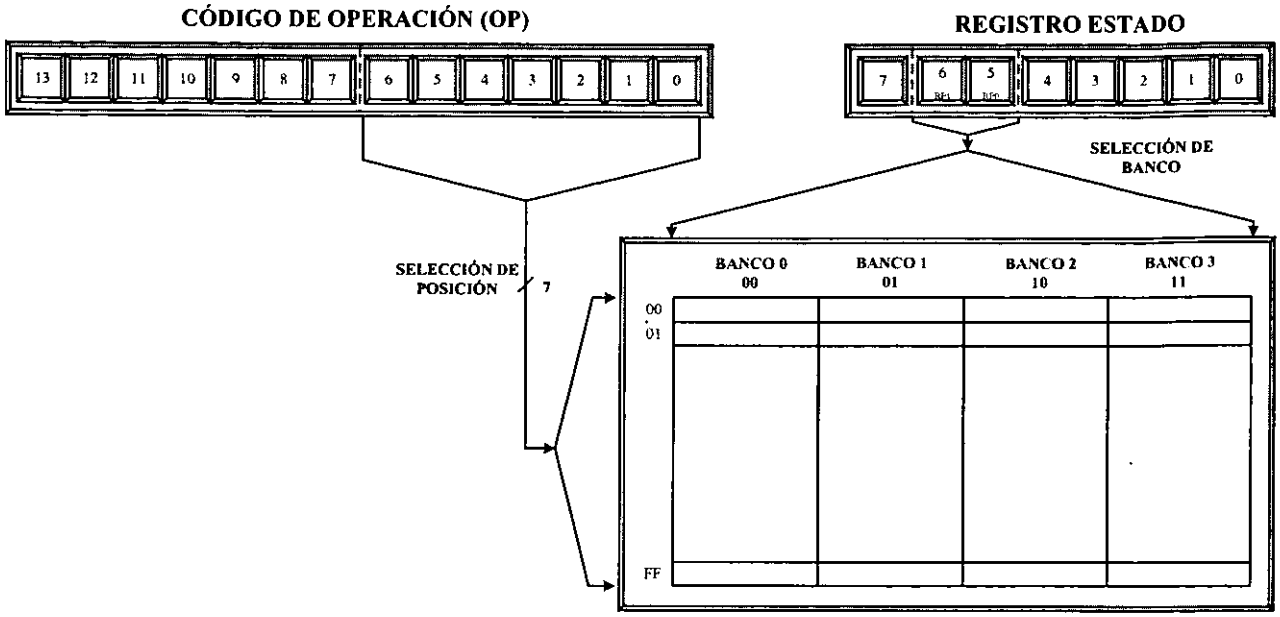


Figura 2.5 Direccionamiento directo.

En el caso del PIC16X84 sólo se usa el bit RP0 al tener implementados únicamente dos bancos.

2° Direccionamiento indirecto.

En este caso el operando de la instrucción hace referencia al registro INDF, que ocupa la posición 0H del área de datos. En realidad el registro INDF no está implementado físicamente por lo que no se podrá accesar a él, y cuando se le hace referencia se accesa a la posición que apunta el registro FSR, que se halla situado en la posición 4H del banco 0. Los 7 bits de menos peso de FSR <6:0> seleccionan la posición y su bit de más peso junto con el bit IRP del registro ESTADO <7>, seleccionan el banco, como se ve en la figura 2.6.

Si se intenta leer el registro INDF siempre encontrará 00H, y si intenta escribir en él se producirá un NOP, es decir una instrucción que no hace nada.

Como sólo hay dos bancos en el PIC16X84 en este modo de direccionamiento, el estado del bit IRP se ignora.

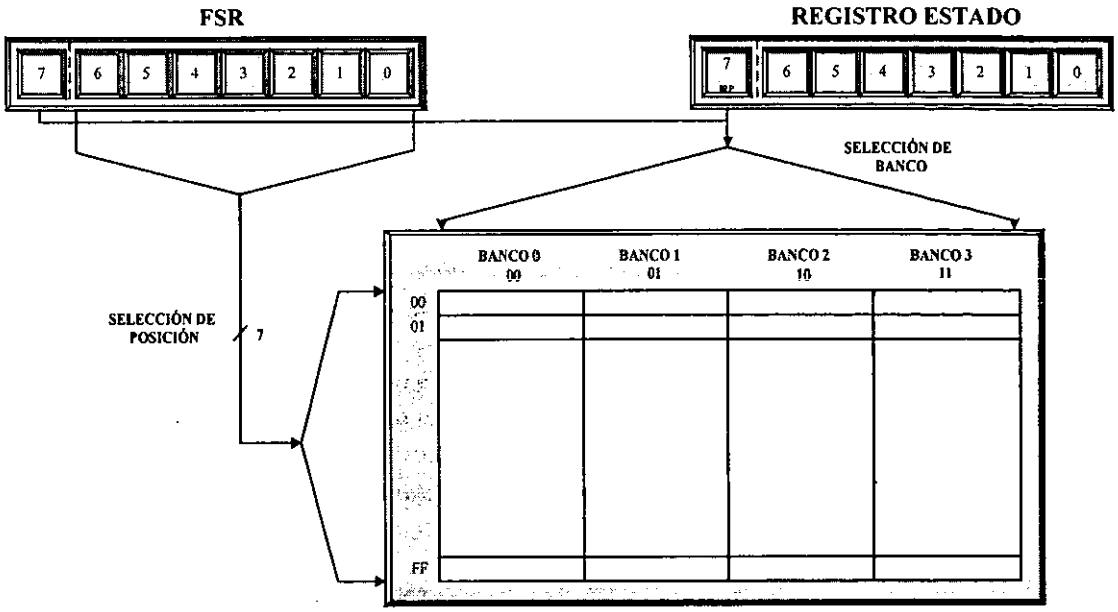


Figura 2.6 Direccionamiento indirecto.

Memoria de programa.

La arquitectura de los PIC de la gama media admite un mapa de memoria de programa capaz de contener 8192 (8K) instrucciones de 14 bits cada una. Este mapa se divide en páginas de 2048 posiciones. Para direccionar 8 K posiciones se necesitan 13 bits como se muestra en la tabla 2.1, que es la longitud que tiene el Contador de Programa. Sin embargo el PIC16X84 sólo tiene implementadas 1 K posiciones, por lo que ignora los 3 bits de más peso del PC.

VALOR HEXADECIMAL	EQUIVALENTE DECIMAL	EQUIVALENTE BINARIO
000	0	0 00 00 0000 0000
3FF	1023	0 00 11 1111 1111
1FFF	8191	1 11 11 1111 1111

Tabla 2.1. Bits de direccionamiento.

Bits de más peso ignorados del PC

El contador de programa, pila y registros PCL y PCLATH.

Tal como se muestra en la figura 2.7, el rango de direcciones que cubre el PIC16X84 en su memoria de programa llega desde la 0000H-03FFH, o sea, un total de 1024 posiciones.

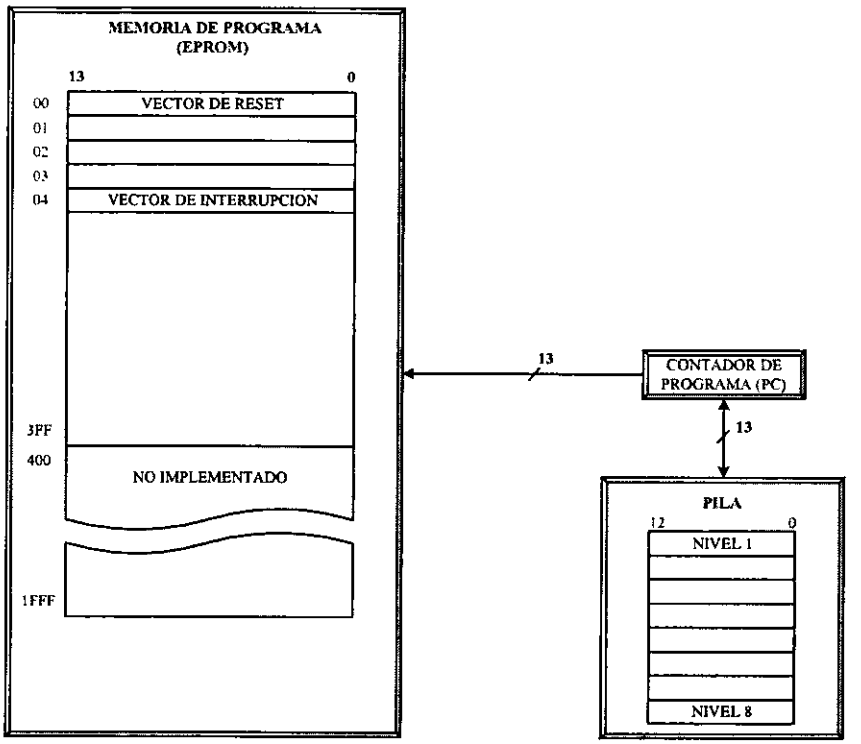


Figura 2.7 Memoria de programa del PIC16X84.

En el PC se ignoran los tres bits de más peso de tal forma que apuntar a la dirección 33H es lo mismo que hacerlo a la 433H, 833H, C33H, 1033H, 1433H o a la 1C33H.

Al igual que todos los registros específicos que controlan la actividad del procesador, el PC está implementado sobre un par de posiciones de la memoria RAM. Cuando se escribe el PC como resultado de una operación de la ALU, los 8 bits de menos peso del PC residen en el registro PCL, que ocupa repetido, la posición 02H de los dos bancos de la memoria de datos. Los bits de más peso del PC <12:8>, residen en los 5 bits de menos peso del registro PCLATH como se muestra en la figura 2.8, que ocupa la posición 0AH de los dos bancos de la memoria RAM como se estudió en la figura 2.4.

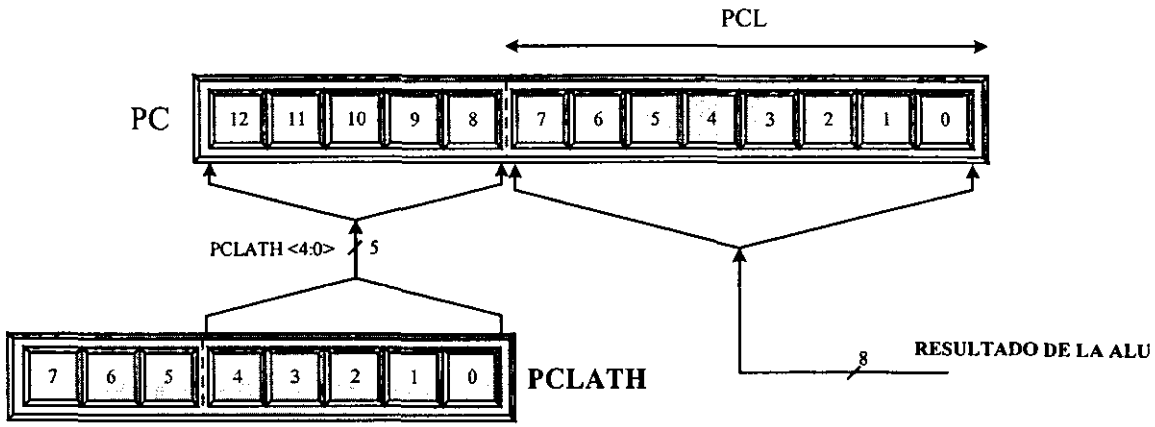


Figura 2.8 Carga del PC con el resultado de la ALU.

En las instrucciones GOTO y CALL de la gama media los 11 bits de menos peso del PC provienen del Código de Operación (OP) y los otros 2 de los bits PCLATH <4:3>. Como se muestra en la figura 2.9.

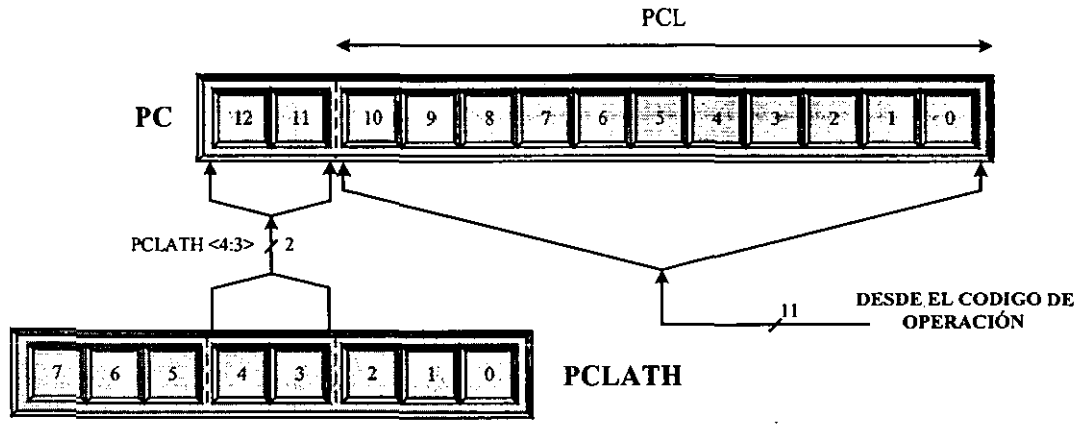


Figura 2.9 Carga del PC durante las instrucciones GOTO y CALL.

Con los 11 bits que se cargan en el PC desde el código de las instrucciones GOTO y CALL a través del decodificador de instrucciones, se puede direccionar una página de 2 K de la memoria. Los bits restantes del PC <12:11> tienen la misión de apuntar una de las cuatro páginas de memoria y, en los modelos de PIC que alcanzan ese tamaño dichos bits proceden de PCLATH <4:3>.

La PILA es la zona aislada de las memorias de instrucciones y datos. Tiene una estructura LIFO, en la que el último valor guardado es el primero que sale. Tiene ocho niveles cada uno con 13 bits. Funciona como un "búffer circular"[8], de manera que el valor que se obtiene al realizar el noveno desempilado es igual al que se obtuvo en el primero.

La instrucción CALL y las interrupciones originan la carga del contenido del PC en el nivel superior o cima de la PILA. El contenido del nivel superior se saca de la PILA al ejecutar las instrucciones RETURN, RETLW y RETFIE. El contenido de registro PCLATH no es afectado por la entrada o salida de información de la PILA.

[8] Búffer circular: Sincronizador interpuesto entre dos acumuladores distintos, de manera que las operaciones siguen su curso mientras se transfieren datos del acumulador intermedio al secundario o interno.

Los vectores RESET e INTERRUPCIÓN se analizarán en la sección 2.2.4.

Memoria EEPROM de datos.

El PIC16X84 contiene una EEPROM de datos de 16 bytes. Esta memoria no forma parte del espacio direccionable normal, y sólo es accesible para lectura/escritura a través de dos registros: EEDATA en la dirección 08H del banco 0, en el que se depositan los datos que se leen o se escriben, y EEADR en la dirección 09H del banco 0, en el que se carga la dirección a accesar de la EEPROM de datos. Las 64 posiciones de un byte ocupan las direcciones de un mapa que comienza en 00H y termina en 3FH, por eso los 2 byte de más peso del registro EEADR siempre valen 0.

El registro EECON1 en la dirección 88 del banco 1 tiene funciones de control de las operaciones en la EEPROM. La distribución de sus bits se muestra en la figura 2.10, y la tabla 2.2 muestra la función de cada uno de éstos.

Figura 2.10 Distribución de los bits del registro EECON1.

BIT	FUNCIÓN
Bit 0 = RD (Read Data)	Lectura de datos. 1 = Habilita la lectura de datos. 0 = Estado normal (cambia automáticamente).
Bit 1 = WR (Write Data)	Escritura de datos. 1 = Habilita la escritura de datos. 0 = Estado normal (cambia automáticamente)
Bit 2 = WREN (WRite ENable)	Permiso de escritura. 1 = Permite la escritura de la EEPROM. 0 = Prohibe la escritura de la EEPROM.
Bit 3 = WRERR (WRite ERRor)	Señalizador de error de escritura. 1 = Cuando una operación de escritura ha registrado errores. 0 = La operación de escritura se ha completado correctamente.
Bit 4 = EEIF (EEPROM Interrupt Flag)	Señalizador de fin de operación de escritura. 1 = La operación de escritura se ha realizado correctamente (se pone a 0 por software). 0 = La operación de escritura no se ha completado.

Tabla 2.2 Descripción y función de los bits del registro EECON1.

Esta memoria EEPROM no emplea ningún recurso de alimentación externo, y funciona en todo el rango de alimentación permitido para el PIC16X84. Su duración típica de programación es de 10 ms, que resulta muy larga para la velocidad del procesador. Por este motivo existen varios bits en el registro EECON1 para supervisar la completa y correcta programación.

El registro EECON2 en realidad no está implementado físicamente; al leerlo todos sus bits son 0. Sólo se emplea como un dispositivo de seguridad durante el proceso de escritura de la EEPROM, para evitar las interferencias en el largo intervalo de tiempo que precisa su desarrollo.

2.2.2. Registros de configuración y control.

En esta sección describiremos los registros internos de la memoria RAM, aunque esta descripción se haga individualmente, existen vínculos entre registros de acuerdo con la arquitectura, como se analizó anteriormente con los registros INDF, PCL, PCLATH, FSR EECON1, EECON2, EEADR y EEDATA que ya no se incluyen en este capítulo.

El registro TMR0 por su dependencia con otros dispositivos dentro de la propia arquitectura y para comprender mejor su interacción entre ellos, se analizarán como parte de la función específica de temporización.

El registro ESTADO.

Este registro ocupa la dirección 03H tanto del banco 0 como del banco 1. Sus bits tienen 3 funciones distintas:

1ª Se encargan de avisar las incidencias del resultado de la ALU.

2ª Indican el estado de RESET.

3ª Seleccionan el banco de memoria a accesar.

En la figura 2.11 se muestra el diagrama de distribución de los bits del registro ESTADO.

REGISTRO ESTADO

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R	R	R/W	R/W	R/W
IRP	RP1	RP0	TO#	PD#	Z	DC	C

Figura 2.11 Estructura interna del registro ESTADO.

Los bits TO# y PD# indican el estado del procesador en algunas condiciones y no se pueden escribir, son de sólo lectura (R) a diferencia del resto que se pueden tanto leer como escribir (R/W). Por este motivo la instrucción *clrf ESTADO* deja el contenido de dicho registro con el valor *000uu1uu* (*u* = no cambia). Sólo se ponen a cero los tres bits de más peso, el bit Z (Zero – Cero) se pone a 1 y los restantes no alteran su valor.

La tabla 2.3 muestra la nomenclatura y función de cada uno de los bits del registro ESTADO y la tabla 2.4 muestra los valores que toman los bits TO# y PD# después de producirse un RESET.

BIT	FUNCIÓN
Bit 0 = C (Carry)	Acarreo en el bit de más peso (8° Bit). 1 = Acarreo en la suma y no en la resta. 0 = Acarreo en la resta y no en la suma. Cambia con las instrucciones ADDWF, SUBWF y ADDLW.
Bit 1 = DC (Digit Carry)	Acarreo en el 4° bit de menos peso. 1 = Acarreo en la suma y no en la resta. 0 = Acarreo en la resta y no en la suma. Cambia con las instrucciones ADDWF, SUBWF y ADDLW , y está orientado a la aritmética en BCD.
Bit 2 = Z (Zero)	1 = El resultado de una operación es 0. 0 = El resultado es diferente de 0.
Bit 3 = PD# (Power Down)	1 = Tras conectar V_{DD} o ejecutar CLRWDT. 0 = Al ejecutar la instrucción SLEEP.
Bit 4 = TO# (Time Out)	1 = Tras conectar V_{DD} o ejecutar CLRWDT o SLEEP. 0 = Al rebasar (desbordamiento) el tiempo del WDT.
Bit 5 y 6 = RP0 y RP1	Selección de la página de memoria de programa (para PIC's con 4 bancos de memoria): 00 = Banco 0 10 = Banco 2 01 = Banco 1 11 = Banco 3
Bit 7 = IRP	Selección de bancos para direccionamiento indirecto. Este bit junto con el de más peso del registro FSR, sirven para determinar el banco de la memoria de datos seleccionado. 0 = Bancos 0 y 1 1 = Bancos 2 y 3 En el PIC16X84 al disponer de sólo dos bancos, no se usa este bit y debe programarse como 0.

Tabla 2.3. Descripción y función de los bits del registro ESTADO.

TO#	PD#	Estado tras el RESET
0	0	Fin del WDT del modo SLEEP.
0	1	Fin del WDT en el modo normal.
1	0	MCLR# activada durante el modo SLEEP
1	1	Conexión de V_{DD}
u	u	MCLR# se mantiene en 0

Tabla 2.4 Estados de los bits TO# y PD# tras un RESET

El registro OPCIÓN.

La función principal de este registro es controlar el TMR0 y el Divisor de Frecuencia (*Pre-scaler*), ocupa la posición 81H en la memoria de datos, que equivale a la dirección 01H del banco 1. La figura 2.12 muestra la distribución de los bits del registro OPCIÓN. La tabla 2.5 muestra la nomenclatura y función de cada uno de los bits del registro OPCIÓN.

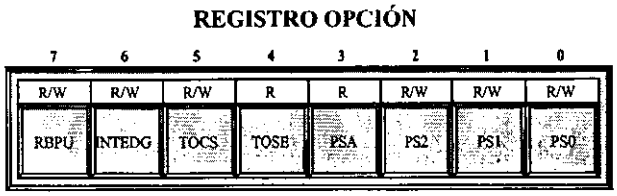


Figura 2.12 Estructura interna del registro OPCIÓN.

BIT	FUNCIÓN
Bits 0-2 = PS0, PS1 y PS2 (PreScaler Bit)	Valor con el que actúa el divisor de frecuencia.

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

BIT	FUNCIÓN
Bit 3 = PSA (PreScaler Assigment)	Asignación del divisor de frecuencia. 1 = El divisor de frecuencia se le asigna al WDT. 0 = El divisor de frecuencia se le asigna al TMR0.
Bit 4 = T0SE (Timer 0 Clock Source Select)	Tipo de flanco en el TOCKI. 1 = Incremento del TMR0 cada flanco descendente. 0 = Incremento del TMR0 cada flanco ascendente.
Bit 5 = T0CS (Timer 0 Clock Edge Select)	Define el tipo de reloj para el TMR0. 1 = Pulsos introducidos a través de TOCKI (contador). 0 = Pulsos de reloj interno Fosc/4 (temporizador).
Bit 6 = INTEDG (Interrupt Edge)	Tipo de Flanco para la interrupción. 1 = Flanco ascendente. 0 = Flanco descendente.
Bit 7 = RBPU# (RB Pull-Up)	Conexión de resistencias Pull-Up en el puerto B. 1 = Desactivadas. 0 = Activadas.

Tabla 2.5 Funciones de los bits del registro OPCIÓN.

El registro de interrupciones INTCON.

Puesto que los microcontroladores PIC de la gama media admiten interrupciones, requieren un registro encargado de su regulación y se encuentra implementado en la dirección 0B. Los tres bits de menos peso del registro INTCON son señalizadores En la figura 2.13 se muestra la distribución de los bits del registro INTCON y la tabla 2.6 muestra la nomenclatura y función de los bits del registro INTCON.

REGISTRO INTCON

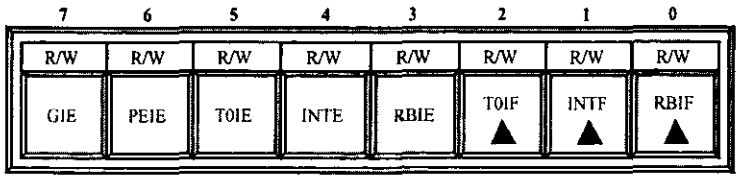


Figura 2.13 Distribución de los bits del registro INTCON.

BIT	FUNCIÓN
Bit 0 = RBIF (RB Interrupt Flag)	Señalizador de cambio de estado en las terminales RB4-RB7 del puerto B. 1 = Pasa a 1 cuando cambia el estado de alguna de las terminales. 0 = No ha cambiado el estado de alguna de la terminales.
Bit 1 = INTF (INTerrupt Flag)	Señalizador de activación de la terminal RB0/INT. 1 = Al activarse RB0/INT. 0 = RB0/INT no se ha activado.
Bit 2 = T0IF (Timer 0 Interrupt Flag)	Señalizador de desbordamiento del TMR0. 1 = Cuando ha ocurrido el desbordamiento. 0 = Indica que el TMR0 no se ha desbordado.
Bit 3 = RBIE (RB Interrupt Enable)	Activación de interrupción por cambio de estado en las terminales RB4-RB7 del puerto B. 1 = Permite esta interrupción. 0 = Prohibe esta interrupción.
Bit 4 = INTE (Interrupt Enable)	Activación de la interrupción en la terminal RB0/INT. 1 = Permite la interrupción al activarse RB0/INT. 0 = Prohibe esta interrupción.
Bit 5 = T0IE (Timer 0 Interrupt Enable)	Permiso de interrupción por desbordamiento del TMR0. 1 = Permite la interrupción al desbordarse el TMR0. 0 = Prohibe esta interrupción.

(Continúa)

BIT	FUNCIÓN
Bit 6 = PEIE (PEripheral Interrupt Enable)	Activación de la interrupción de periféricos (comparador). 1 = Permite la interrupción de periféricos. 0 = Prohibe esta interrupción.
Bit 7 = GIE (Global Interrupt Enable)	Activación global de interrupciones. 1 = Permite la ejecución de todas las interrupciones (separadamente de sus bits individuales). 0 = Prohibido el permiso de interrupciones.

Tabla 2.6 Funciones de los bits del registro INTCON.

PUERTO A y TRIS A.

El PUERTO A dispone de 5 bits. Las líneas RA0 a RA3 están dispuestos conforma a la figura 2.14. Todas ellas están provistas como salidas con un búffer[9] CMOS normal, y como entradas aceptan niveles TTL[10].

El sentido de trabajo de todas las líneas del PUERTO A lo controla el registro TRIS A, en el que un bit a 0 activa a la línea correspondiente como salida, y un bit a 1 la activa como entrada, después de un RESET, todos los bits del registro TRIS A toman el valor de 1.

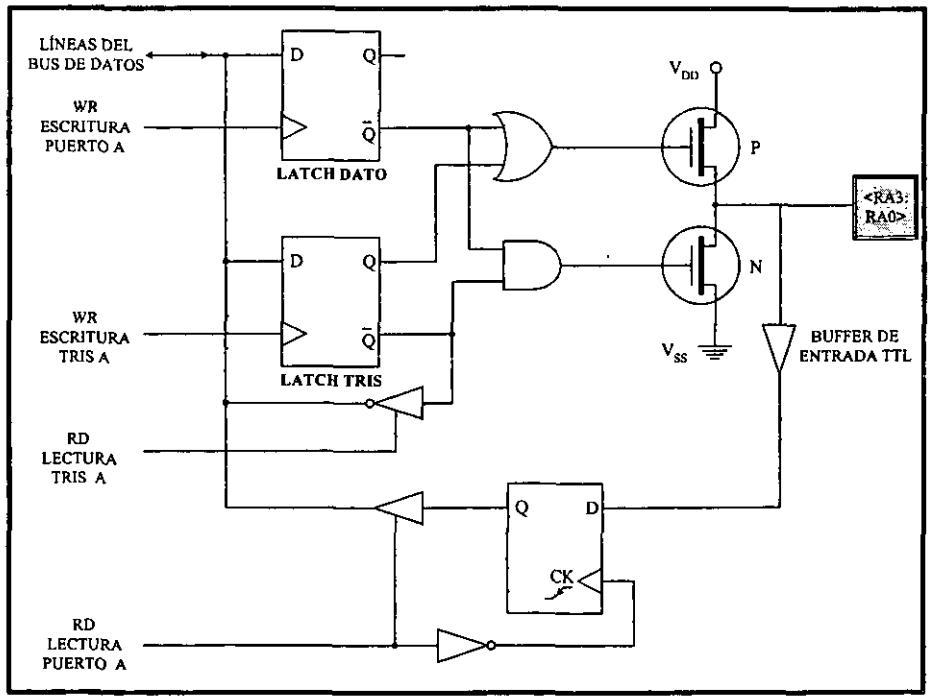


Figura 2.14 Conexión interna de las terminales <RA3:RA0> del PUERTO A con las líneas de control y el bus de datos del procesador.

[9] Búffer: Dispositivo destinado a separar o aislar dos etapas o circuitos de manera que no se produzcan entre éstos reacciones perjudiciales.

[10] TTL (Transistor to Transistor Logic): Lógica de Transistor a Transistor.

Cuando se lee una línea del PUERTO A se recoge el nivel lógico que tiene en ese momento. Las líneas cuando actúan como salidas pasan a través de un LATCH[11], lo que significa que sus terminales sacan el nivel lógico que se haya cargado por última vez en el registro PUERTO A. La escritura de un puerto implica que primero se lea el PUERTO A, luego se modifique el valor y finalmente se escriba en el LATCH de salida.

Cuando se saca un nivel lógico por una línea del PUERTO A, primero se deposita en la línea correspondiente del bus interno de datos y se activa la señal WRITE (WR), lo que origina el almacenamiento de este nivel en el LATCH DATO. En esta situación el LATCH TRIS debería contener un 0 para que actuase como salida.

Con estos valores la puerta OR y AND tendrían un 0 en sus salidas. Estos valores producen la conducción del transistor P-MOS y al bloqueo del transistor N-MOS. Así, la terminal de I/O queda conectada a V_{DD} y tiene un nivel alto. La línea de salida conserva su valor hasta que no se reescriba en el LATCH.

Si una línea actúa como entrada, el nivel lógico depositado en ella desde el exterior pasa a la línea correspondiente del bus interno de datos cuando se activa la señal READ (RD) y se hace conductor el dispositivo triestado que les une. Al programarse en el LATCH TRIS la línea del puerto como entrada, los dos transistores MOS de salida quedan bloqueados y la línea en alta impedancia. Hay que tomar en cuenta que cuando se lee una línea de entrada se obtiene el estado actual que tenga su terminal correspondiente y no el valor que se haya almacenado en el LATCH DATO.

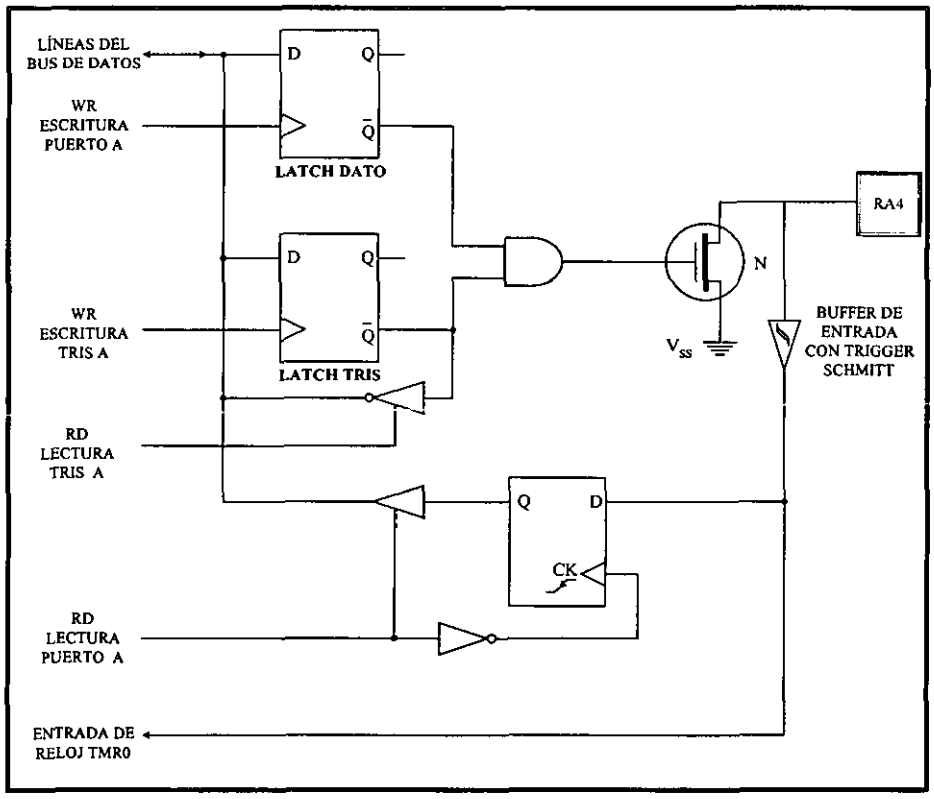


Figura 2.15 Diagrama de conexión de la línea RA4 del Puerto A.

[11] LATCH: Dispositivo electrónico con disposición de compuertas AND – NOT retroalimentados usados comúnmente para almacenamiento temporal.

Como puede verse en la figura 2.15 la línea RA4 del PUERTO A tiene una estructura algo diferente, en este caso la salida es de tipo drenador abierto, y la entrada está provista de un Schmitt-trigger[12] misma que está conectada con la entrada externa del temporizador TMR0.

PUERTO B y TRIS B.

Consta de 8 líneas bidireccionales de I/O <RB7:RB0>, cuya información se almacena en el registro PUERTO B, que ocupa la dirección 06H del banco 0 de la memoria de datos. El registro de configuración TRIS B ocupa la misma dirección en el banco 1 como se analizó en la figura 2.4.

La línea RB0/INT tiene dos funciones, además de ser una línea de I/O actúa como terminal de petición de una interrupción externa, cuando se autoriza esta función mediante la programación del registro INTCON.

A todas las líneas de este puerto se les puede conectar una resistencia *pull-up*[13] a través de la configuración del bit RBPU# = 0 del registro OPTION.

El sentido de trabajo de todas las líneas del PUERTO B lo controla el registro TRIS B, en el que un bit a 0 activa a la línea correspondiente como salida, y un bit a 1 la activa como entrada, después de un RESET, todos los bits del registro TRIS B toman el valor de 1 y se desactivan las resistencias *pull-up*.

Las cuatro líneas <RB7:RB4>, pueden programarse para generar una interrupción si alguna de ellas cambia su estado lógico. En la figura 2.16 se muestra el esquema de conexión entre las terminales <RB7:RB4> y las líneas correspondientes del bus de datos interno.

El estado de las terminales <RB7:RB4> configuradas como entradas se compara con el valor anterior que tenían y que se había almacenado en un latch durante la última lectura del PUERTO B. El cambio de estado en alguna de esas líneas origina una interrupción y la activación del señalizador RBIF.

La línea RB6 también se utiliza para la grabación serie de la memoria de programa y sirve para soportar la señal se reloj. La línea RB7 constituye la entrada de los datos en serie.

[12] Las compuertas Schmitt-trigger son dispositivos que se utilizan para convertir señales imperfectas, lentas o con ruido en señales digitales bien definidas, rápidas y sin ruido.
[13] Carga *pull-up*: En electrónica, conexión del positivo de la fuente de alimentación y una terminal de I/O a través de una resistencia de alto valor.

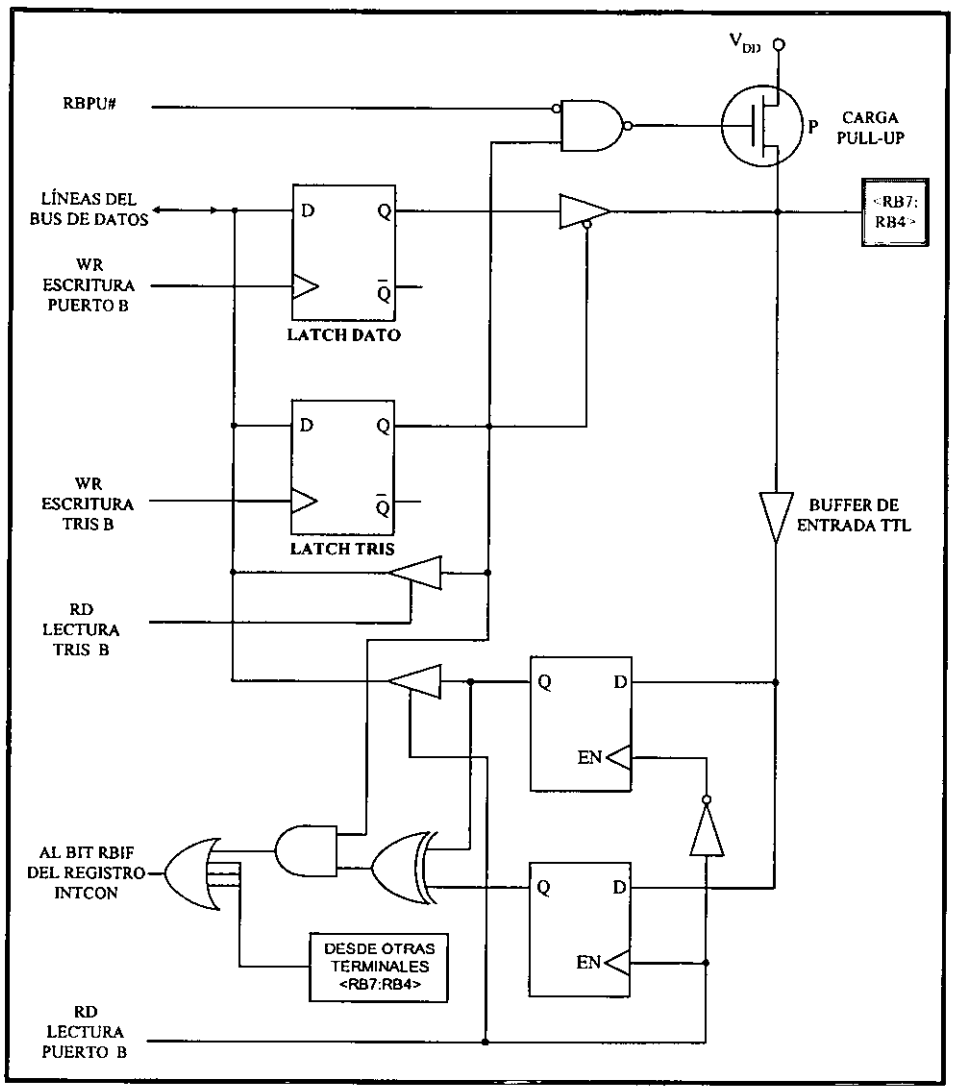


Figura 2.16 Diagrama de conexión de las terminales <RB7:RB4> del PUERTO B.

La figura 2.17 muestra el diagrama de conexión de las líneas <RB3:RB0>. La línea punteada indica que la terminal RB0/INT cuenta además con un búffer Schmitt Trigger para la función de interrupción externa.

En la tabla 2.7 se presenta un resumen de las características y funciones de las terminales del PUERTO B, en donde $ST^{(1)}$ es el búffer Schmitt-trigger habilitado cuando la terminal es configurada para la función de interrupción externa y $ST^{(2)}$ es el búffer Schmitt-trigger habilitado cuando la terminal es usada para la programación serial del microcontrolador.

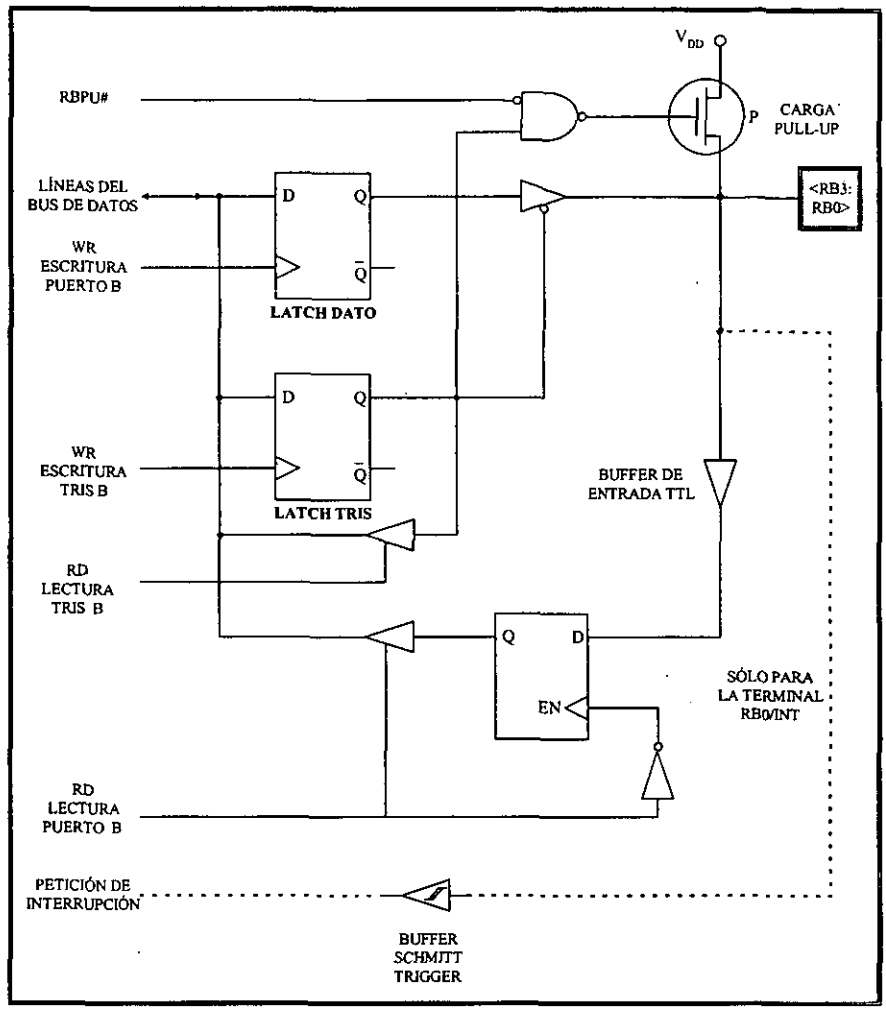


Figura 2.17 Diagrama de conexión de las terminales <RB3:RB0> del PUERTO B.

NOMBRE	BIT	TIPO DE BUFFER	FUNCIÓN
RB0/INT	Bit 0	TTL/ST	Terminal de I/O o entrada de interrupción externa. Carga interna pull-up programable por software.
RB1	Bit 1	TTL	Terminal de I/O. Carga interna pull-up programable por software.
RB2	Bit 2	TTL	Terminal de I/O. Carga interna pull-up programable por software.
RB3	Bit 3	TTL	Terminal de I/O. Carga interna pull-up programable por software.
RB4	Bit 4	TTL	Terminal de I/O (interrupción con cambio de estado lógico). Carga interna pull-up programable por software.
RB5	Bit 5	TTL	Terminal de I/O (interrupción con cambio de estado lógico). Carga interna pull-up programable por software.
RB6	Bit 6	TTL/ST(2)	Terminal de I/O (interrupción con cambio de estado lógico). Carga interna pull-up programable por software. Reloj de programación serial.
RB7	Bit 7	TTL/ST(2)	Terminal de I/O (interrupción con cambio de estado lógico). Carga interna pull-up programable por software. Datos de programación serial.

Tabla 2.7 Resumen de funciones de las terminales del PUERTO B.

La palabra de configuración.

Se trata de una posición reservada de la memoria de programa situada en la dirección 2007H y accesible únicamente durante el proceso de grabación. Al escribirse el programa de la aplicación es necesario grabar el contenido de esta posición de acuerdo con las características del sistema.

En la figura 2.18 se muestra la distribución y asignación de los 14 bits de la Palabra de Configuración de los PIC16X8X y en la tabla 2.8 se describen las funciones de cada bit.

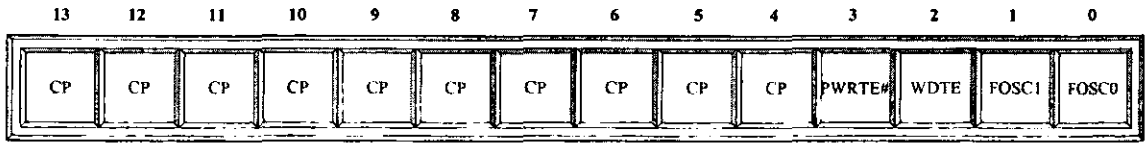


Figura 2.18 Distribución de los bits de la Palabra de Configuración del PIC16X8X.

BIT	FUNCIÓN		
Bit 0-1 = FOSC0, FOSC1 (Frecuency OSCilator)	Selección del Oscilador utilizado:		
	FOSC1	FOSC0	Oscilador
	0	0	LP
	0	1	XT
	1	0	HS
	1	1	RC
Bit 2 = WDTE (WatchDog Timer Enable)	Activación del WDT 1 = WDT Activado. 0 = WDT Desactivado.		
Bit 3 = PWRTE# (PoWeR-up Timer Enable)	Activación del Temporizador Power-up 1 = Desactivado. 0 = Activado.		
Bit 4-13: CP (Code Protect)	Bits de protección del Código de Programa. 1 = No protegida. 0 = Protegida.		

Tabla 2.8 Función de los bits de la Palabra de Configuración.

El temporizador *Power-up* retrasa 72 ms la puesta en marcha o RESET que se produce al conectar la alimentación al PIC, para garantizar la estabilidad de la tensión aplicada.

Cuando los bits de protección del Código de Programa están en 0 el programa no se puede leer, evitando copias. Además evita que se pueda accesar la EEPROM de datos y, finalmente, si se modifica el bit CP de 0 a 1, se borra completamente la EEPROM.

Una de las diferencias que tiene el PIC16C84 con el PIC16F84 es la actuación del bit PWRTE de la palabra de configuración, que tiene invertida su función en ambos. En el primero cuando este bit = 1 el temporizador de *Power-up* está activado, mientras que sucede lo contrario en el segundo.

2.2.3. Temporización.

Una de las labores más habituales en los programas de control de dispositivos suele ser el de determinar intervalos concretos de tiempo, el cual recibe el nombre de temporizador (*timer*). También suele ser frecuente contar los impulsos que se producen en el exterior del sistema, y el elemento destinado a este fin se denomina contador que es de tipo ascendente/descendente.

Si las labores del temporizador o contador las asignáramos al programa principal quitarían mucho tiempo al procesador con su consecuente detrimento en actividades más importantes. Por esta razón se diseñan recursos específicamente orientados a estas funciones.

Los PIC 16FX8X poseen un temporizador/contador de 8 bits, llamado TMR0 que tiene dos funciones:

1ª Como contador de eventos, que están representados por los impulsos que se aplican a la terminal RA4/TOCKI. Al llegar al valor FFH se desborda el contador, con el siguiente impulso pasa al valor 00H, activando con esto un señalizador y/o provocando una interrupción.

2° Como temporizador, cuando se carga en el registro TMR0 un valor inicial y se incrementa con cada ciclo de instrucción (Fosc/4) hasta que se desborda, es decir pasa de FFH a 00H, activando con esto un señalizador y/o provocando una interrupción.

La figura 2.19 ilustra las dos funciones del temporizador TMR0.

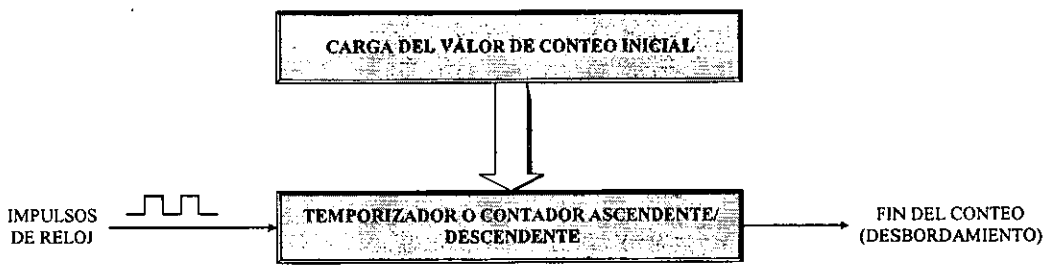


Figura 2.19 Esquema simplificado de un temporizador/contador.

Para que el TMR0 funcione como contador de impulsos aplicados a la terminal TOCKI hay que poner a 1 el bit T0CS, que ocupa el bit 5 del registro OPCIÓN. En esta situación el registro TMR0 se incrementa con cada flanco activo. El tipo de flanco activo se elige programando el bit T0SE, que ocupa el bit 4 del registro OPCIÓN; si T0SE = 1 el flanco activo es descendente, y si T0SE = 0, es ascendente. Cuando se desea que el TMR0 funcione como temporizador el bit T0CS = 0.

Los PIC 16XF8X disponen de dos temporizadores, el TMR0 y el Temporizador Perro Guardián (WDT – Watch Dog Timer). El primero actúa como principal y sobre el recae el control de tiempos y el conteo de impulsos. El otro vigila que el programa no permanezca en un ciclo infinito y para ello cada cierto tiempo comprueba si el programa se esta ejecutando normalmente. En caso contrario, si el control esta detenido en un ciclo infinito el WDT genera una señal de RESET.

El TMR0 se comporta como un Registro de Función Especial ubicado en la dirección 01H del banco 0 de la memoria de datos.

A menudo el TMR0 y el WDT precisan controlar largos intervalos de tiempo y necesitan aumentar la duración de los impulsos que les incrementa. Para esto dispone de un circuito programable denominado Divisor de Frecuencia, que divide la frecuencia utilizada en diversos rangos.

Para programar el comportamiento del TMR0, el WDT y el Divisor de Frecuencia se utilizan algunos bits del registro OPCIÓN y de la Palabra de Configuración. En la figura 2.20 se proporciona un esquema simplificado de la arquitectura del circuito de control de tiempos usado en los PIC 16X8X.

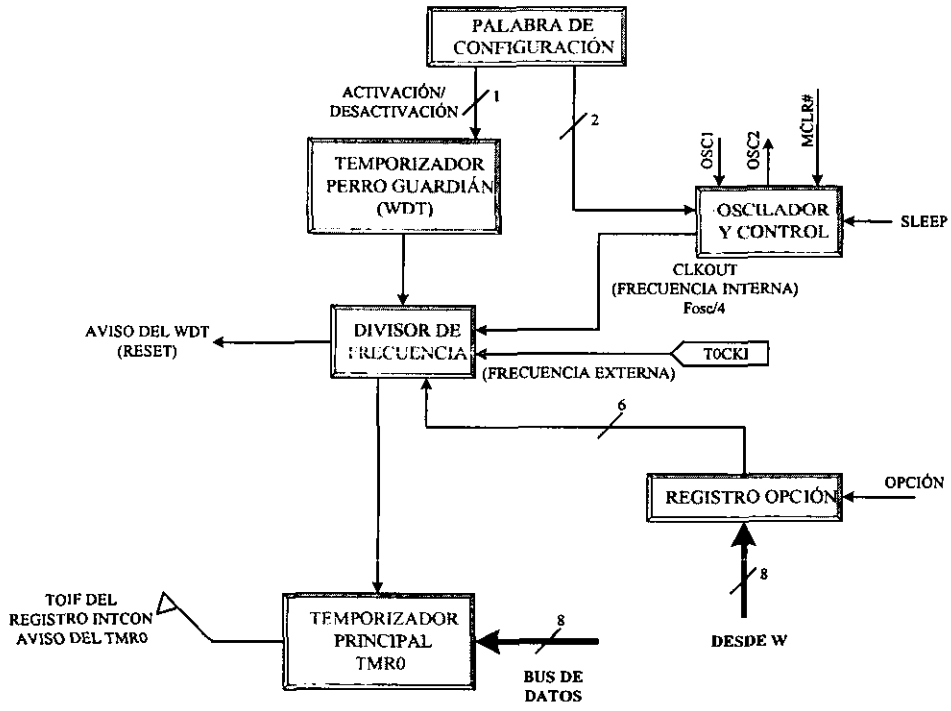


Figura 2.20 Esquema simplificado del control de tiempos en la arquitectura de los PIC16X8X.

El Divisor de Frecuencia puede usarse con el TMR0 o con el WDT. Con el TMR0 actúa como *Pre*-divisor *(Pre-scaler),* es decir, los impulsos pasan primero por el Divisor y luego se aplican al TMR0 una vez aumentada su duración. Con el WDT actúa después realizando la función de *Post-divisor (Post-scaler).* Los impulsos que divide por un rango el Divisor de Frecuencia, pueden provenir de la señal de reloj interna Fosc/4 o de los que se aplican en la terminal T0CKI.

TMR0 puede ser leído y escrito en cualquier momento al estar conectado al bus de datos. Funciona como un contador ascendente de 8 bits. Cuando funciona como temporizador conviene cargarle con el valor de los impulsos que se quiere temporizar, pero expresados en complemento a 2. De esta manera, al desbordarse se activa el señalizador T0IF que ocupa el bit 2 del registro INTCON y/o se produce una interrupción.

Para calcular los tiempos a medir con el TMR0 se utilizan las siguientes formulas prácticas:

Temporización = 4 * Tosc * (Valor cargado en el TMR0) * (Rango del divisor).
Valor a cargar en el TMR0 = Temporización/4 * Tosc * (Rango del divisor).

En cualquier momento se puede leer el valor que contiene TMR0, sin detener su conteo.

En la figura 2.21 se muestra el esquema de funcionamiento del TMR0. Observe que hay un bloque que retrasa 2 ciclos el conteo para sincronizar el momento del incremento producido por la señal aplicada en T0CKI con el que se producen los impulsos internos del reloj. Cuando se escribe en el TMR0 se retrasan dos ciclos su reincremento y se pone a 0 el divisor de frecuencia.

El bit PSA selecciona la asignación del Divisor de Frecuencia al TMR0 o al WDT.

El WDT es un contador interno de 8 bits que origina un RESET cuando se desborda. Su control de tiempos es independiente del TMR0 y está basado en una simple red RC. Su actuación es opcional y puede bloquearse para que no funcione programando el bit WDTE que ocupa el bit 3 de la Palabra de Configuración.

La temporización nominal con la que se halla programado el WDT es de 18 ms, pero utilizando el Divisor de Frecuencia puede aumentarse hasta alcanzar los 2.3 segundos.

Para evitar que se desborde el WDT y genere un RESET, hay que recargar o refrescar su cuenta antes de que llegue a su desbordamiento. Este refresco, que en realidad consiste en ponerlo a 0 para iniciar la temporización, se consigue por software con las instrucciones *CLRWDT* y *SLEEP*. La instrucción *CLRWDT* borra al WDT y reinicia su cuenta. A diferencia la instrucción *SLEEP*, además de borrar WDT detiene al sistema y lo mete a un estado de resposo o bajo consumo. Si no se desactiva el WDT al entrar en el modo SLEEP, al completar su conteo provocará un RESET y sacará al microcontrolador del modo de bajo consumo.

En el registro ESTADO existe un bit denominado TO# que pasa a valer 0 después del desbordamiento del WDT.

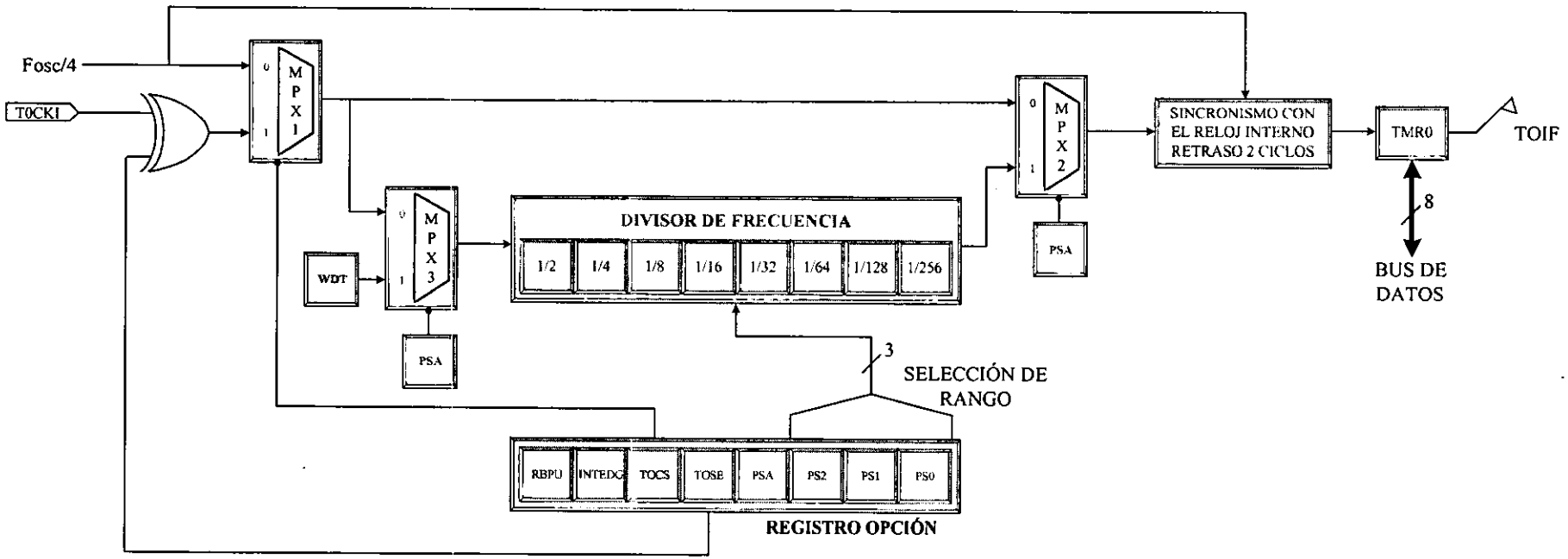


Figura 2.21 Esquema general del funcionamiento del TMR0.

2.2.4. *Interrupciones y reset.*

Las llamadas a subrutinas mediante la instrucción *Call* son desviaciones del flujo de control del programa originadas por instrucciones, por lo que se consideran síncronas[14]. Se producen cada vez que se ejecuta dicha instrucción.

Las interrupciones son desviaciones del flujo de control del programa originadas asíncronamente[15] por diversos sucesos que no se hallan bajo supervisión de las instrucciones. Dichos sucesos pueden ser externos al sistema, como la generación de un flanco o nivel activo en una terminal del microcontrolador, o bien internos, como el desbordamiento de un contador o temporizador.

El comportamiento del microcontrolador ante la interrupción es similar al de la instrucción *CALL* de llamada a subrutina. En ambos casos se detiene la ejecución del programa en curso, se guarda la dirección actual del Contador de Programa (PC) en la Pila y se carga el PC con una dirección, que en el caso de *CALL* viene acompañando a la propia instrucción, y en el caso de una interrupción es una dirección **reservada** de la memoria de programa EEPROM, llamada **Vector de Interrupción**.

En los PIC16X8X el Vector de Interrupción se encuentra situado en la dirección 04H, en donde comienza la Rutina de Servicio de Interrupción (RSI). En general, en dicho vector se suele colocar una instrucción de salto incondicional (*GOTO*), que traslada el flujo de control a la zona de la memoria de programa EEPROM destinada a contener la rutina de atención a la interrupción.

La RSI suele comenzar guardando en la memoria de datos algunos registros específicos del procesador. Concretamente aquellos que la RSI va a emplear y a alterar su contenido. Antes del retorno al programa principal se recuperan los valores guardados y se restaura completamente el estado del procesador. Algunos procesadores guardan estos registros en la Pila, pero los PIC no disponen de instrucciones para meter (*push*) y sacar (*pop*) información de la Pila, utilizando para este fin Registros de Propósito General de la memoria de datos.

Los PIC16X8X pueden ser interrumpidos por cuatro causas diferentes, pero todas ellas desvían el flujo de control a la dirección 04H, por lo que otra de las operaciones iniciales de la RSI es averiguar cuál de las posibles causas ha sido la responsable de la interrupción en curso. Para ello se exploran los señalizadores de las fuentes de interrupción.

Otro detalle importante en la RSI de los PIC16X8X es que éstos poseen un bit GIE (Global Interrupt Enable – Habilitador Global de Interrupción) que ocupa la posición 7 del registro INTCON, cuando dicho bit vale 0 prohibe todas las interrupciones y al comenzar la RSI se pone automáticamente a 0, con objeto de no atender nuevas interrupciones hasta que se termine la que ha comenzado. Es el retorno final de la interrupción que devuelve el valor de GIE nuevamente a 1 para volver a tener en cuenta las interrupciones. Dicho retorno de interrupción se realiza mediante la instrucción *RETFIE*.

---

[14] Los sistemas síncronos son aquellos en los que las variables de estado interno o las variables de entrada no actúan directamente sobre el sistema, sino que lo hacen en los instantes en que éste recibe impulsos procedentes de un generador.

[15] Los sistemas asíncronos son aquellos en los que las variables de entrada actúan de forma directa sobre el sistema. Los cambios de estado pueden afectarse en cualquier instante de tiempo.

Antes del retorno conviene borrar el señalizador de la causa de interrupción que se ha atendido, porque si bien los señalizadores se ponen a 1 automáticamente en cuanto se produce la causa que indican, la puesta a 0 se hace por programa.

Causas de interrupción.

Los PIC116X8X tienen cuatro causas o fuentes posibles de interrupción:

1ª Activación de la terminal RB0/INT.
2ª Desbordamiento del temporizador TMR0.
3ª Cambio de estado de una de la terminales <RB7:RB4> del PUERTO B.
4ª Finalización de la escritura en la EEPROM de datos.

Cuando ocurre cualquiera de los cuatro sucesos indicados se origina una petición de interrupción, que si se acepta y se atiende, comienza depositando el valor del PC actual en la Pila poniendo el bit GIE = 0 y cargando en el PC el valor 04H, que es el Vector de Interrupción desde donde se desvía el flujo de control.

Cuando GIE = 0 no se acepta ninguna de las interrupciones. Si GIE = 1 solo se aceptan aquellas fuentes de interrupción cuyo bit de permiso también se lo consienta, es decir, cada fuente de interrupción tiene otro bit de permiso, que según su valor permite o prohibe la realización de dicha interrupción, además cada fuente de interrupción dispone de un señalizador.

Para conocer qué causa ha provocado la interrupción se exploran los señalizadores, tres de los cuales se ubican en el registro INTCON y el cuarto (EEIF) se halla en el bit 4 del registro EECON1.

Los señalizadores deben ponerse a 0 por programa antes del retorno de la interrupción y son operativos aunque la interrupción este prohibida con su bit de permiso correspondiente. En la figura 2.22 se muestra el esquema de la lógica de control que origina la interrupción.

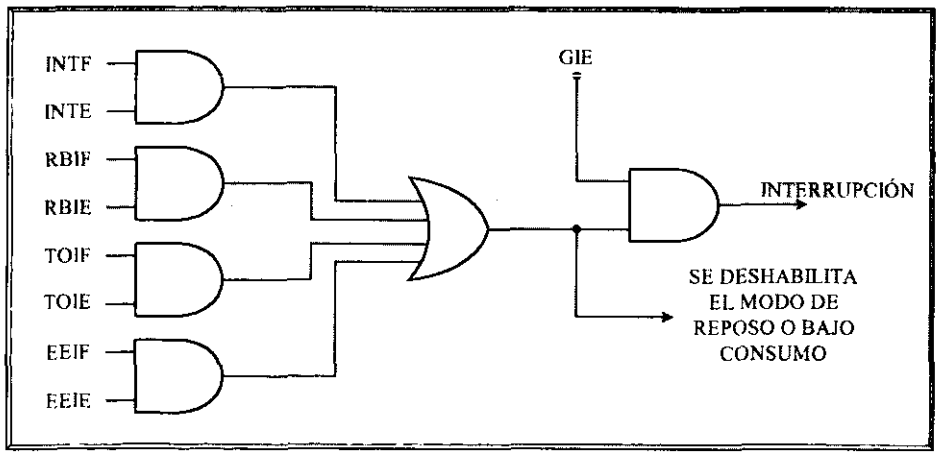


Figura 2.22 Lógica de control para la generación de una interrupción en los PIC16X8X.

Interrupción externa INT.

Esta fuente de interrupción es sumamente importante para atender acontecimientos externos en tiempo real. Cuando ocurre alguno de ellos activa la terminal RB0/INT y se hace una petición de interrupción. Entonces, de forma automática, el bit INTF = 1 y si el bit de permiso INTE = 1, se autoriza el desarrollo de la interrupción.

Mediante el bit 6, llamado INTEDG, del registro OPCIÓN se puede seleccionar cuál será el flanco activo en RB0/INT. Si se desea el ascendente se escribe un 1 en dicho bit, y si se desea que sea el descendente se escribe un 0.

El procesador explora el señalizador INTF al final del primer ciclo de reloj de cada ciclo de instrucción.

Si INTE = 1 y se aplica un flanco activo en RB0/INT, el señalizador INTF se pone a 1 automáticamente y se solicita una interrupción, que es aceptada cuando GIE = 1. Antes de regresar al programa principal hay que borrar INTF, puesto que en caso contrario al ejecutar la instrucción de retorno *RETFIE* se volvería a desarrollar el mismo proceso de interrupción.

Interrupción por desbordamiento del TMR0.

Cuando el TMR0 se desborda y pasa del valor FFH al 00H, el señalizador T0IF se pone automáticamente a 1. Si además, el bit de permiso de la interrupción del TMR0 T0IE = 1 y el bit GIE = 1, se produce la interrupción.

Si no se recarga el TMR0 cuando se desborda, sigue contando desde 00H a FFH. En cualquier momento se puede leer y escribir este registro, pero cada vez que se escribe se pierden dos ciclos de reloj para la sincronización.

Cuando se carga inicialmente TMR0 con el valor N_{10}, cuenta 256 – N impulsos, siendo el tiempo que tarda en hacerlo el que expresa la siguiente fórmula:

$$Temporización = 4 * Tosc * (256 - N_{10}) * Rango\ del\ Divisor\ de\ Frecuencia.$$

Interrupción por cambio de estado de una de la terminales <RB7:RB4> del PUERTO B

Esta interrupción esta diseñada específicamente para detectar la pulsación de una tecla correspondiente a un teclado matricial que se explora con cuatro líneas de I/O. Para esta función se destinan las líneas <RB7:RB4> del PUERTO B, que cada vez que cambia el estado lógico de una de ellas fuerza al señalizador RBIF a ponerse a 1, y si los bits de permiso RBIE = 1 y GIE = 1 se autoriza la interrupción.

Interrupción por finalización de la escritura en la EEPROM de datos.

El tiempo típico que tarda en desarrollarse una operación de escritura en la EEPROM de datos de los PIC16X8X es de 10 ms, que es considerable comparado con la velocidad a la que el procesador ejecuta instrucciones. Para asegurarse que se ha completado la escritura y puede continuarse con el flujo de control del programa es aconsejable manejar la interrupción que se origina al finalizar la escritura, que pone automáticamente el señalizador EEIF a 1, y se autoriza siempre que los bits de permiso EEIE = 1 y GIE = 1.

Cuando se describió el proceso de escritura de le EEPROM de datos se indicó que se usaba un registro *no real* para asegurar la misma. Se trataba del registro EECON2, en el que se grababan dos valores, el 55H y el AAH. Durante la escritura de este registro debe prohibirse la aceptación de interrupciones a fin de salvaguardar la operación de escritura, por eso en ese módulo se pone GIE = 0.

En los PIC16C84 y PIC16F8X se puede leer y escribir la EEPROM de datos aunque se haya protegido el código. En los PIC16CR8X, que disponen de memoria ROM para el código existen dos bits para el código de protección: uno dedicado a la ROM de código y el otro para la EEPROM de datos.

Causas de reinicialización o reset.

Los PIC16X8X tienen cinco causas que provocan la reinicialización del sistema, que consiste en cargar al PC con el valor 00H (Vector RESET) y poner el estado de los bits de los Registros Específicos con un valor predeterminado.

1ª Conexión de la alimentación (POR – Power On Reset).
2ª Activación de la terminal MCLR# (Master Clear Reset) en operación normal.
3ª Activación de la terminal MCLR# en el modo SLEEP[16].
4ª Desbordamiento del WDT en operación normal.
5ª Desbordamiento del WDT en el modo SLEEP.

En la tabla 2.9 se presenta el estado lógico que adquieren los bits de los Registros de Función Especial de la memoria de datos cuando se presenta un RESET por una de las cinco causas posibles y en la figura 2.23 se muestra el esquema electrónico de los PIC16X8X para la generación del RESET. La terminal MCLR# dispone de un filtro interno para eliminar los ruidos y los impulsos muy pequeños.

[16] SLEEP: En electrónica se entiende como un estado de reposo o bajo consumo de energía.

REGISTRO	DIRECCIÓN	CONEXIÓN DE LA ALIMENTACIÓN	DESBORDAMIENTO WDT MODO NORMAL	DESBORDAMIENTO WDT MODO SLEEP	MCLR# MODO NORMAL	MCLR# MODO SLEEP
W	----	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
INDF	00H	----	----	----	----	----
TMR0	01H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PC	02H	0000H	0000H	PC + 1	0000H	0000H
ESTADO	03H	0001 1xxx	0000 1uuu	uuu0 0uuu	000u uuuu	0001 0uuu
FSR	04H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PUERTO A	05H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PUERTO B	06H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TRIS A	85H	---1 1111	---1 1111	---u uuuu	---1 1111	---1 1111
TRIS B	86H	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
OPCIÓN	81H	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
EEDATA	08H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
EEADR	09H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
EECON1	88H	---0 0000	---0 ?000	---u uuuu	---0 ?000	---0 ?000
EECON2	89H	----	----	----	----	----
PCLATH	0AH	---0 0000	---0 0000	---u uuuu	---0 0000	---0 0000
INTCON	0BH	0000 000x	0000 000u	uuuu uuuu	0000 000u	0000 0000

u = No cambia x = Indeterminado ---- = No implementado ? = Depende de otras condiciones

Tabla 2.9 Valor que adquieren los bits de los FSR´s y el W después de producirse un RESET.

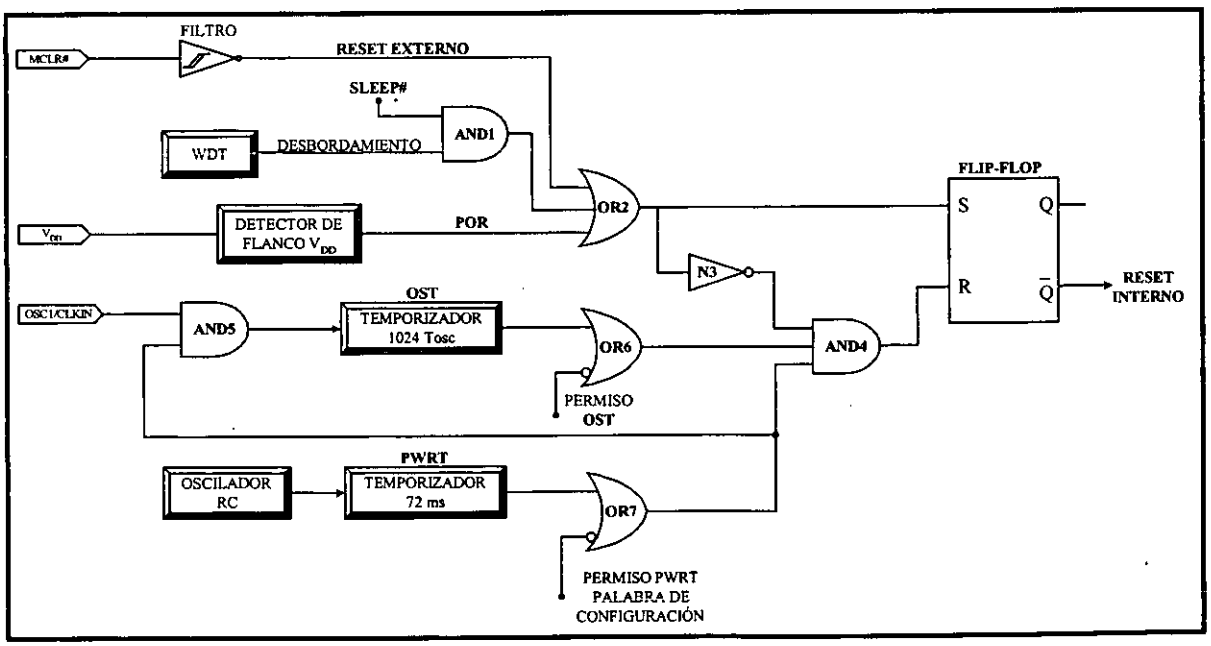


Figura 2.23 Diagrama de conexión para la generación interna del RESET.

El temporizador PWRT# (Power Up Timer – Temporizador de Arranque) activa una salida al cabo de un cierto tiempo tras la conexión de la alimentación, que se aplica a la entrada de la AND4, encargada de controlar el reset del Flip-Flop[17] que controla la generación interna del RESET del sistema. Sólo es válida la salida de PWRT si el bit de permiso de éste está activo a nivel bajo (PWRTE# = 0), dado que también se aplica a la OR7 de la figura 2.23. El bit PWRTE# reside en la Palabra de Configuración en la posición 3, y como ya se mencionó una de las pocas diferencias que tiene el PIC16C84 con el PIC16F84 es que el primero tiene como nivel activo el bit PWRTE# en 1, mientras que este último lo tiene en 0.

El RESET del Flip-Flop final se produce cuando la AND4 produce un nivel lógico alto, lo que requiere que sus tres entradas tengan este mismo nivel lógico, lo que supone:

a) Que no haya peticiones de RESET y la compuerta OR2 tenga su salida en un nivel lógico bajo.
b) Que si está activo PWRT finalice su retardo de 72 ms.
c) Que también finalice la temporización del oscilador OST, que retarda 1.024 * Tosc, tras completarse el retardo de PWRT.

El temporizador PWRT# proporciona un retardo fijo de 72 ms y sus impulsos de reloj los genera un oscilador RC propio. Este tiempo garantiza la estabilización del voltaje de alimentación V_{DD}.

El temporizador OST (Oscilator Start-Up Timer – Temporizador Inicio de Oscilador) proporciona un retardo de 1.024 * Tosc (Período de los impulsos aplicados a la terminal OSC1/CKLIN). Sirve para asegurar que el cristal de cuarzo o resonador cerámico empleado en los osciladores tipo XT, LP o HS esté estabilizado y en marcha. OST comienza a funcionar cuando termina el retardo de PWRT# debido a la conexión de la salida OR7 con la AND5 de la figura 2.23.

La activación de la entrada SET del Flip-Flop se consigue cuando se activa la terminal MCLR#, cuando se desborda el WDT o cuando se detecta un flanco ascendente en la terminal V_{DD} (POR – Power On Reset).

En la figura 2.24 se muestra un cronograma de las principales señales que participan en la generación del RESET y en el que se aprecia la secuencia de los retardos $T_{PWRT\#}$ y T_{OST}.

[17] Un circuito **Flip-Flop** puede mantener un estado binario en forma indefinida hasta que recibe la dirección de una señal de entrada para cambiar su estado.

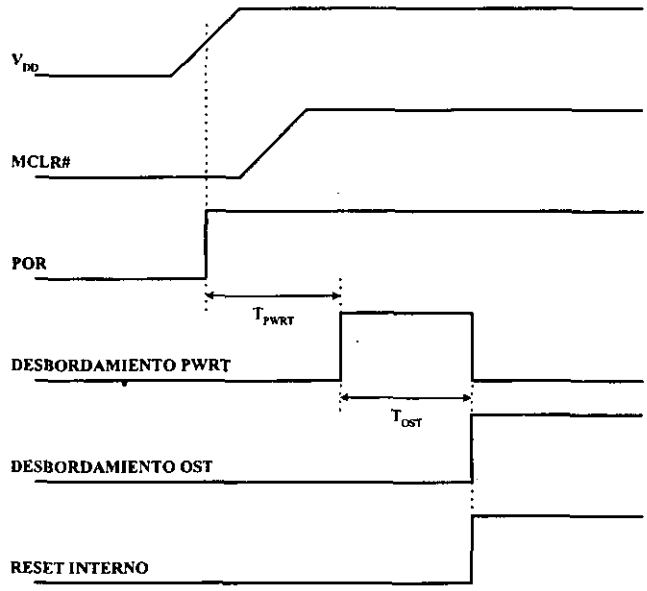


Figura 2.24 Cronograma de las principales señales que intervienen en el RESET para uno de los posibles casos en el que *MCLR#* no cambia de estado con V_{DD}.

En el registro ESTADO hay dos bits que indican las condiciones en las que se ha originado el RESET. Se trata de TO# (Timer Out) y PD# (Power Down) y se explicaron en la tabla 2.4.

Se produce un fallo en la alimentación cuando V_{DD} desciende por debajo del valor mínimo, sin llegar a cero, y luego se recupera a este fallo se le conoce como *Brown – Out* (Apagón). En esta situación es preciso provocar un RESET. Para generar un RESET en un PIC16X8X cuando hay un fallo de alimentación hay que colocar un circuito externo de protección, como los que se muestran en la figura 2.25.

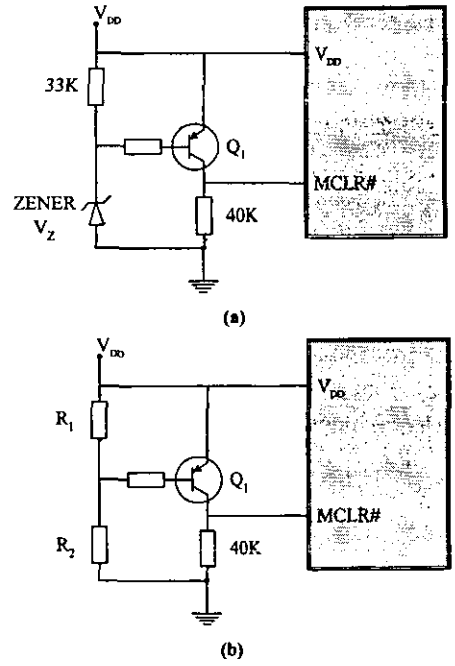


Figura 2.25 Circuitos externos de protección.

2.2.5. El estado sleep.

Este modo de funcionamiento de los PIC está caracterizado por el reducido consumo de energía que requiere y está recomendado en aquellas aplicaciones en las que hay largos períodos de espera hasta que se produzca algún suceso asíncrono, como la pulsación de una tecla. En dichos períodos el procesador está inactivo.

El consumo típico del PIC es de 2 mA, aproximadamente, reduciéndose a menos de 10μA en el modo SLEEP, lo que permite alimentarle con una pequeña pila durante casi dos años.

Para entrar en el modo SLEEP hay que ejecutar las instrucciones *SLEEP*. En el modo SLEEP la terminal T0CKI se conecta a V_{DD} o a V_{SS}, para eliminar la entrada de impulsos externos al TMR0. Por otra parte, como se detiene el oscilador principal que genera los impulsos Tosc, también se detiene el TMR0. Las terminales de I/O mantienen el estado anterior al modo SLEEP y las que no se hallan conectadas a periféricos y actúan como entradas de alta impedancia se aconseja conectarlas a VDD o a VSS para evitar posibles fugas de corriente. La terminal MCLR# debe conectarse a nivel lógico alto. Sin impulsos de reloj el procesador se detiene y deja de ejecutar instrucciones hasta que se reactive y salga del modo SLEEP.

Si el WDT continúa activo en el modo SLEEP, al entrar en este modo se borra, pero sigue funcionando. Los bits del Registro ESTADO PD# y TO# toman los valores 0 y 1 respectivamente.

Para salir del modo SLEEP existen tres alternativas:

1ª Activación externa de MCLR# para provocar un RESET.
2ª Desbordamiento del WDT si quedó operativo en el modo SLEEP.
3ª Generación de una interrupción. En este caso, como el TMR0 está detenido sólo pueden producirse los otros tres tipos de interrupción.

Cuando se reactiva el PIC se desarrolla la secuencia del oscilador OST, que retarda 1.024 Tosc para estabilizar la frecuencia de trabajo, y luego pasa a ejecutar la siguiente instrucción a *SLEEP* (PC + 1).

Los bits TO# y PD# se emplean para conocer la causa del RESET que reactiva al sistema. PD# = 0 cuando se ejecuta la instrucción SLEEP. TO# = 0 cuando se desborda el WDT.

2.3. Repertorio de instrucciones.

Descripción detallada en orden alfabético.

A continuación se presentan las 35 instrucciones de los PIC de la gama media en forma tabular y en orden alfabético. Esta descripción puede ser de utilidad cuando se esta haciendo un programa para encontrar alguna propiedad de interés.

ADDLW	**ADD Literal to W**
Sintaxis	ADDLW *k*
Codificación	11 111x kkkk kkkk
Palabras, ciclos	1, 1
Operación	W + *k* → W
Bit de estado	C, DC, Z
Descripción	Añade el contenido de W al literal *k*, y almacena el resultado en W.
¡Atención!	Esta instrucción no existe en los PIC 16C5X

ADDWF	**ADD W to F**
Sintaxis	ADDWF *f,d*
Codificación	0001 11df ffff 00 0111 dfff ffff
Palabras, ciclos	1, 1
Operación	W + *f* → *f* si *d* = 1 o W + *f* → W si *d* = 0
Bit de estado	C, DC, Z
Descripción	Añade el contenido de W al contenido de *f*, y almacena el resultado en W si *d* = *0*, y en *f* si *d* = *1*.

ANDLW	**AND Literal and W**
Sintaxis	ANDLW *k*
Codificación	1110 kkkk kkkk 11 1001 kkkk kkkk
Palabras, ciclos	1, 1
Operación	W AND *k* → W
Bit de estado	Z
Descripción	Efectúa un AND lógico entre el contenido de W y el literal *k*, y almacena el resultado en W.

ANDWF	**AND W with F**
Sintaxis	ANDWF *f,d*
Codificación	0001 01df ffff 00 0101 dfff ffff
Palabras, ciclos	1, 1
Operación	W AND *f* → *f* si *d* = 1 o W AND *f* → W si *d* = 0
Bit de estado	Z
Descripción	Efectúa un AND lógico entre el contenido de W y el contenido de f y coloca el resultado en W si *d* =*0*, y en *f* si *d* = *1*.

BCF	**Bit Clear F**
Sintaxis	BCF *f,b*
Codificación	0100 bbbf ffff 01 00bb bfff ffff
Palabras, ciclos	1, 1
Operación	$0 \rightarrow b(f)$
Bit de estado	Ninguno
Descripción	Pone a cero el bit número *b* de *f*.

BSF	**Bit Set F**
Sintaxis	BSF *f,b*
Codificación	0101 bbbf ffff 01 01bb bfff ffff
Palabras, ciclos	1, 1
Operación	$1 \rightarrow b(f)$
Bit de estado	Ninguno
Descripción	Pone a uno el bit número *b* de *f*.

BTFSC	**Bit Test, Skip if Clear**
Sintaxis	BTFSC *f,b*
Codificación	0110 bbbf ffff 01 10bb bfff ffff
Palabras, ciclos	1, 1 o 2 (ver descripción)
Operación	Salto si $b(f) = 0$
Bit de estado	Ninguno
Descripción	Si el bit número *b* de *f* es nulo, la instrucción que sigue a ésta se ignora y se trata como un NOP. En este caso, y sólo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.

BTFSS	**Bit Test, Skip if Set**
Sintaxis	BTFSS *f,b*
Codificación	0111 bbbf ffff 01 11bb bfff ffff
Palabras, ciclos	1, 1 o 2 (ver descripción)
Operación	Salto si $b(f) = 1$
Bit de estado	Ninguno
Descripción	Si el bit número *b* de *f* está a 1, la instrucción que sigue a ésta se ignora y se trata como un NOP. En este caso, y sólo en este caso, la instrucción BTFSS precisa dos ciclos para ejecutarse.

CALL	Subroutine Call
Sintaxis	CALL *k*
Codificación	1001 kkkk kkkk 10 0kkk kkkk kkkk
Palabras, ciclos	1, 2
Operación	En el caso de los 16C5X: PC + 1 → pila, *k* → PC (0 –7), 0 → PC (8), PA2 a PA0 → PC (9 – 11). En el caso de los 16C64, 71, 74 y 84: PC + 1 → pila, *k* → PC (0 – 10), PCLATH(3, 4) → PC (11, 12).
Bit de estado	Ninguno
Descripción	Salvaguarda la dirección de vuelta en la pila y después llama a la subrutina situada en la dirección cargada en el PC.
¡Atención!	El modo de cálculo de la dirección difiere según la familia PIC utilizada. También hay que posicionar correctamente PA2, PA1 y PA0 (PIC 16C5X) o el registro PCLATH (en los demás PIC) antes de ejecutar la instrucción CALL.

CLRF	Clear F
Sintaxis	CLRF *f*
Codificación	0000 011f ffff 00 0001 1fff ffff
Palabras, ciclos	1, 1
Operación	00 → *f*
Bit de estado	Z
Descripción	Pone el contenido de *f* a cero y activa el bit Z del registro de estado.

CLRW	Clear W register
Sintaxis	CLRW
Codificación	0000 0100 0000 00 0001 0xxx xxxx
Palabras, ciclos	1, 1
Operación	00 → W
Bit de estado	Z
Descripción	Pone el registro W a cero y activa el bit Z del registro de estado.

CLRWDT	Clear Watchdog Timer
Sintaxis	CLRWDT
Codificación	0000 0000 0100 00 0000 0110 0100
Palabras, ciclos	1, 1
Operación	00 → WDT y 0 → predivisor del temporizador
Bit de estado	1 → TO y 1 → PD
Descripción	Pone a cero el registro contador del temporizador watchdog, así como el predivisor

COMF	Complement F
Sintaxis	COMF *f,d*
Codificación	0010 01df ffff 00 1001 dfff ffff
Palabras, ciclos	1, 1
Operación	/*f* → *f* si *d* = 1 o /*f* → W si *d* = 0
Bit de estado	Z
Descripción	Hace el complemento de *f* bit a bit. El resultado se almacena de nuevo en *f* si *d* = *1*, y en W si *d* = *0* (en este caso, *f* no varía).

DECF	Decrement F
Sintaxis	DECF *f,d*
Codificación	0000 11df ffff 00 0011 dfff ffff
Palabras, ciclos	1, 1
Operación	*f* – 1 → *f* si *d* = 1 o *f* – 1 → W si *d* = 0
Bit de estado	Z
Descripción	Decrementa el contenido de *f* en una unidad. El resultado se almacena de nuevo en *f* si *d* = *1*, y en W si *d* = *0* (en este caso, *f* no varía).

DECFSZ	Decrement F, Skip if Zero
Sintaxis	DECFSZ *f,d*
Codificación	0010 11df ffff 00 1011 dfff ffff
Palabras, ciclos	1, 1 (2)
Operación	*f*–1 → *f* si *d* = 1 o *f* – 1 → W si *d* = 0 y salto si *f* – 1 = 0
Bit de estado	Ninguno
Descripción	Decrementa el contenido de *f* en una unidad. El resultado se almacena de nuevo en *f* si *d* = *1*, y en W si *d* = *0* (en este caso *f* no varía). Si el resultado es nulo, se ignora la siguiente instrucción y, en este caso, esta instrucción dura dos ciclos

GOTO	Salto Incondicional
Sintaxis	GOTO *k*
Codificación	101k kkkk kkkk 10 1kkk kkkk kkkk
Palabras, ciclos	1, 2
Operación	En el caso de los 16C5X: *k* → PC (0 – 8), PA2, PA1, PA0 → PC (9 – 11) En el caso de los 16C64, 71, 74 y 84: *k* → PC (0 – 10), PCLATH (3, 4) → PC (11, 12).
Bit de estado	Ninguno
Descripción	Llama a la subrutina situada en la dirección cargada en el PC.
¡Atención!	El modo de cálculo de la dirección difiere según la familia de PIC utilizada. También hay que posicionar correctamente PA2, PA1 y PA0 (PIC 16C5X) o el registro PCLATH (el los demás PIC) antes de ejecutar la instrucción GOTO.

INCF	Increment F
Sintaxis	INCF *f,d*
Codificación	001 10df ffff 00 1010 dfff ffff
Palabras, ciclos	1, 1
Operación	$f+1 \rightarrow f$ si $d=1$ o $f+1 \rightarrow$ W si $d=0$
Bit de estado	Z
Descripción	Incremente el contenido de *f* en una unidad. El resultado se almacena de nuevo en *f* si *d = 1*, y en W si *d = 0* (en este caso, f no varía).

INCFSZ	Increment F, Skip if Zero
Sintaxis	INCFSZ *f,d*
Codificación	0011 11df ffff 00 1111 dfff ffff
Palabras, ciclos	1, 1 (2)
Operación	$f+1 \rightarrow f$ si $d=1$ o $f+1 \rightarrow$ W si $d=0$ y salto si $f+1=0$
Bit de estado	Ninguno
Descripción	Incrementa el contenido de *f* en una unidad. El resultado se almacena de nuevo en *f* si *d = 1*, y en W si *d =0* (en este caso, *f* no varía). Si el resultado es nulo, se ignora la siguiente instrucción y, en ese caso, esta instrucción dura dos ciclos.

IORLW	Inclusive OR Literal with W
Sintaxis	IORLW *k*
Codificación	1101 kkkk kkkk 11 1000 kkkk kkkk
Palabras, ciclos	1, 1
Operación	W OR $k \rightarrow$ W
Bit de estado	Z
Descripción	Efectúa un OR lógico inclusivo entre el contenido de W y el literal *k*, y almacena el resultado en W.

IORWF	Inclusive OR W with F
Sintaxis	IORWF *f,d*
Codificación	0001 00df ffff 11 0100 dfff ffff
Palabras, ciclos	1, 1
Operación	W OR $f \rightarrow f$ si $d=1$ o W OR $f \rightarrow$ W si $d=0$
Bit de estado	Z
Descripción	Efectúa un OR lógico inclusivo entre el contenido de *f*, y almacena el resultado en *f* si *d = 1*, y en W si *d = 0*.

MOVF	**Move F**
Sintaxis	MOVF f,d
Codificación	0010 00df ffff 00 1000 dfff ffff
Palabras, ciclos	1, 1
Operación	$f \rightarrow f$ si $d = 1$ o $f \rightarrow$ W si $d = 0$
Bit de estado	Z
Descripción	Desplaza el contenido de f a f si d = 1 ó a W si d = 0.
¡Atención!	El desplazamiento de *f* a *f*, que a priori parece inutil, permite comprobar el contenido de *f* con respecto a cero, ya que esta instrucción actúa sobre el bit Z.

MOVLW	**Move Literal to W**
Sintaxis	MOVLW *k*
Codificación	1100 kkkk kkkk 11 00xx kkkk kkkk
Palabras, ciclos	1, 1
Operación	$k \rightarrow$ W
Bit de estado	Ninguno
Descripción	Carga W con el literal *k*.

MOVWF	**Move W to F**
Sintaxis	MOVWF *f*
Codificación	0000 001f ffff 00 0000 1fff ffff
Palabras, ciclos	1, 1
Operación	W $\rightarrow f$
Bit de estado	Ninguno
Descripción	Carga *f* con el contenido de W

NOP	**No Operation**
Sintaxis	NOP
Codificación	0000 0000 0000 0000 0xx0 0000
Palabras, ciclos	1, 1
Operación	Ninguna
Bit de estado	Ninguna
Descripción	Sólo consume tiempo de máquina (en este caso, un ciclo) como cualquier otro NOP.

OPTION	Load Option Register
Sintaxis	OPTION
Codificación	0000 0000 0010 00 0000 0110 0010
Palabras, ciclos	1, 1
Operación	W → OPTION
Bit de estado	Ninguno
Descripción	Carga el registro OPTION con el contenido de W.
¡Atención!	Esta instrucción no debe utilizarse en otros circuitos que no sean los PIC 16C5X. No obstante, es correctamente interpretada por los circuitos 16C64, 71, 74 y 84, con el fin de asegurar una compatibilidad ascendente.

RETFIE	Return From Interrupt
Sintaxis	RETFIE
Codificación	00 0000 0000 1001
Palabras, ciclos	1, 2
Operación	Pila → PC, 1→ GIE
Bit de estado	Ninguno
Descripción	Carga el PC con el valor que se encuentra en la parte superior de la pila, asegurando así la vuelta de la interrupción. Pone a 1 el bit GIE, con el fin de autorizar de nuevo que se tengan en cuenta las interrupciones.
¡Atención!	Esta instrucción dura dos ciclos. Esta instrucción no existe en los PIC 16C5X.

RETLW	Return Literal to W
Sintaxis	RETLW *k*
Codificación	1000 kkkk kkkk 11 01xx kkkk kkkk
Palabras, ciclos	1, 2
Operación	*k* → W, pila → PC
Bit de estado	Ninguno
Descripción	Carga W con el literal *k*, y después carga el PC con el valor que se encuentra en la parte superior de la pila, efectuando así un retorno de subrutina.
¡Atención!	Esta instrucción dura dos ciclos.

RETURN	Return from Subroutine
Sintaxis	RETURN
Codificación	00 0000 0000 0000
Palabras, ciclos	1, 2
Operación	Pila → PC
Bit de estado	Ninguno
Descripción	Carga el PC con el valor que se encuentra en la parte superior de la pila, efectuando así una vuelta de subrutina. Se trata de una instrucción RETLW simplificada.
¡Atención!	Esta instrucción dura dos ciclos. Esta instrucción no existe en los PIC 16C5X

RLF	Rotate Left F through Carry
Sintaxis	RLF *f,d*
Codificación	0011 01df ffff 00 1101 dfff ffff
Palabras, ciclos	1, 1
Operación	Ver descripción
Bit de estado	C
Descripción	Rotación de un bit a la izquierda del contenido de *f*, pasando por el bit de acarreo C. Si *d = 1* el resultado se almacena en *f*, si *d = 0* el resultado se almacena en W.

RRF	Rotate Right F through Carry
Sintaxis	RRF *f,d*
Codificación	0011 00df ffff 00 1100 dfff ffff
Palabras, ciclos	1, 1
Operación	Ver descripción
Bit de estado	C
Descripción	Rotación de un bit a la derecha del contenido de *f*, pasando por el bit de acarreo C. Si *d = 1* el resultado se introduce en *f*, si *d = 0* el resultado se almacena en W.

SLEEP	Sleep
Sintaxis	SLEEP
Codificación	0000 0000 0011 00 0000 0110 0011
Palabras, ciclos	1, 1
Operación	0 → PD, 1 → TO, 00 → WDT, 0 → predivisor del WDT
Bit de estado	TO, PD
Descripción	Pone el circuito en modo sleep con parada del oscilador.
¡Atención!	Consulte los capítulos dedicados a cada tipo de circuito para ver las posibilidades exactas y las consecuencias de la entrada del circuito en modo SLEEP.

SUBLW	Substract W from Literal
Sintaxis	SUBLW *k*
Codificación	11 110x kkkk kkkk
Palabras, ciclos	1, 1
Operación	*k* – W → W
Bit de estado	C, DC, Z
Descripción	Sustrae el contenido de W del literal *k*, y almacena el resultado en W. La sustracción se realiza en complemento a dos.
¡Atención!	Esta instrucción no existe en los PIC 16C5X.

SUBWF	Substract W from F
Sintaxis	SUBWF *f,d*
Codificación	0000 10df ffff 00 0010 dfff ffff
Palabras, ciclos	1, 1
Operación	*f* – W → *f* si *d* = 1, o *f* – W → W si *d* = 0.
Bit de estado	C, DC, Z
Descripción	Sustrae el contenido de W del contenido de *f*, y almacena el resultado en W si *d* = *0* y en *f* si *d* = *1*. La sustracción se realiza en complemento a dos.

SWAPF	Swap F
Sintaxis	SWAPF *f,d*
Codificación	0000 10df ffff 00 0010 dfff ffff
Palabras, ciclos	1, 1
Operación	*f*(0 - 3) → *f*(4 – 7) y *f*(4 – 7) → *f*(0 – 3) resultado → dependiendo de *d*.
Bit de estado	Ninguno
Descripción	Intercambia los cuatro bits de mayor peso con los cuatro bits de menor peso de *f*, y almacena el resultado en *f* si *d* = *1*, o en W si *d* = *0*.

TRIS	Load TRIS Register
Sintaxis	TRIS *f*
Codificación	0000 0000 0fff 00 0000 0110 0fff
Palabras, ciclos	1, 1
Operación	W → registro TRIS del puerto *f*
Bit de estado	Ninguno
Descripción	Craga el contenido de W en el registro TRIS del puerto *f*.
¡Atención!	Esta instrucción no debe utilizarse en otros circuitos que no sean los PIC 16C5X. No obstante, es correctamente interpretada por los circuitos 16C64, 71, 74 y 84, con el fin de asegurar una compatibilidad ascendente.

XORLW	**Exclusive OR Literal with W**
Sintaxis	XORLW *k*
Codificación	1111 kkkk kkkk 11 1010 kkkk kkkk
Palabras, ciclos	1, 1
Operación	W XOR *k* → W
Bit de estado	Z
Descripción	Efectúa un OR lógico exclusivo entre el contenido de W y el literal *k*, y almacena el resultado en W.

XORWF	**Exclusive OR W with F**
Sintaxis	XORWF *f,d*
Codificación	0001 10df ffff 00 0110 dfff ffff
Palabras, ciclos	1, 1
Operación	W XOR *f* → *f* si *d* = 1 o W XOR *f* → W si *d* = 0
Bit de estado	Z
Descripción	Efectúa un OR lógico exclusivo entre el contenido de W y el contenido de *f*, y almacena el resultado en *f* si *d* = *1* y en W si *d* = *0*.

CAPITULO 3

EL SISTEMA DE ENTRENAMIENTO

El objetivo del sistema de entrenamiento es consolidar los conocimientos teórico-prácticos en el manejo de los microcontroladores PIC, y servir como elemento de soporte a todo aquel que desee aprender el manejo de dicho dispositivo.

El sistema de entrenamiento que se propone lo conforman un conjunto de módulos independientes entre sí, lo que permite el intercambio y fácil manejo de sus componentes, para que el estudiante pueda realizar los experimentos que se proponen en prácticas que se verán en el Capitulo 5 y proyectos sencillos desarrollados por él mismo. Está diseñado de este modo porque, por ejemplo, en caso de llegar a dañarse alguno de sus componentes no es necesario poner fuera de servicio todo el sistema. Además, un sistema modular tiene la ventaja de reducir el espacio necesario para el desarrollo de prácticas. Los módulos son compactos y con dispositivos accesibles en costo y disponibilidad en el mercado electrónico local.

Una de las necesidades más frecuentes en electrónica digital es recibir información de entrada procedente de interruptores, teclados, fotoceldas, etc. o controlar dispositivos como LED, zumbadores, lámparas, relevadores, motores y otras cargas que operan a partir de fuentes AC o DC de alto voltaje y consumen altas corrientes.

Los diseños reales utilizan diversos periféricos que se conectan a las terminales del microcontrolador que soportan las líneas de I/O. Para este sistema de entrenamiento se utilizan únicamente periféricos digitales.

En las secciones que siguen conoceremos las técnicas que se utilizan comúnmente para llevar a cabo la interconexión de las entradas y salidas del microcontrolador

3.1. Módulos de entradas digitales

Como primer paso vamos a describir los periféricos de entrada más fáciles y usados en los proyectos, los digitales. Si son de entrada introducen un nivel lógico alto o bajo por la línea que se les conecta.

Una forma muy común de proporcionar información en un sistema digital es utilizando interruptores. La función genérica de interruptor es bloquear o permitir el paso de corriente. Los interruptores vienen en una gran variedad de configuraciones y presentaciones. En la figura 3.1 se muestran algunas de ella y sus símbolos.

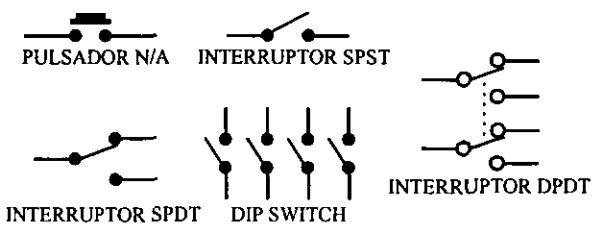


Figura 3.1 Símbolos de diferentes tipos de interruptores.

3.1.2. Módulo de interruptores (dipswitch)

Los interruptores tienen dos estados permanentes y hay que accionarlos para cambiar de uno a otro. El interruptor admite el estado abierto y el estado cerrado. La figura 3.4 muestra las dos formas de conectar interruptores SPST (un polo, un tiro).

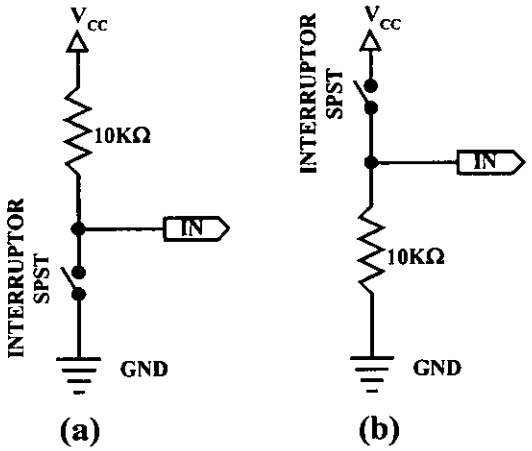


Figura 3.4 Esquema de conexión de interruptores SPST como entradas digitales.

En el esquema de la figura 3.4(a) por la línea de entrada al microcontrolador (IN) se introduce un nivel lógico alto cuando el interruptor está abierto, mientras que cuando se cierra aplica un nivel lógico bajo. La configuración del esquema 3.4(b) funciona al revés.

En nuestro sistema se utilizan interruptores tipo dipswitch los cuales vienen en grupos de 4, 8, 10 o más interruptores miniatura tipo SPST independientes, como se ve en la figura 3.5.

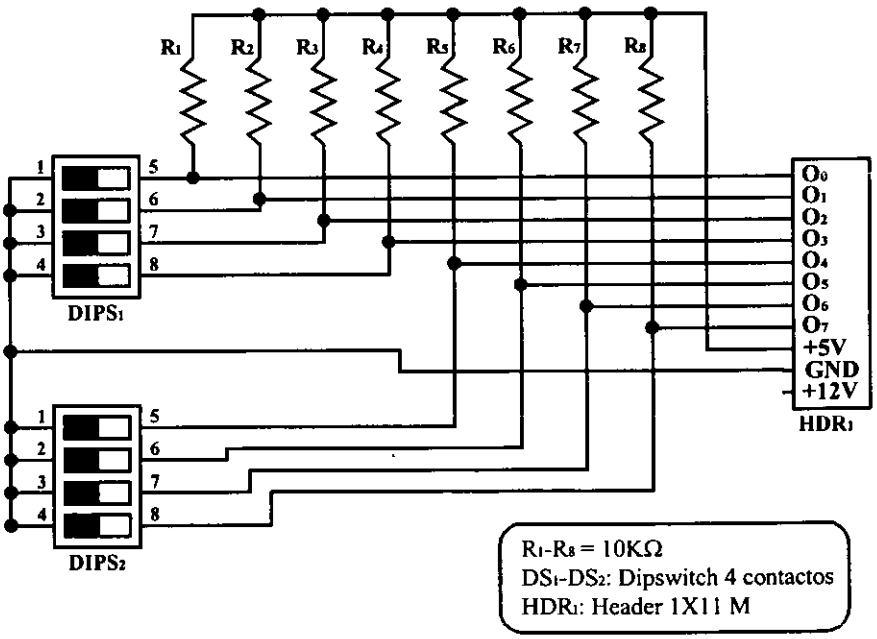


Figura 3.5 Diagrama de módulo con dipswitch.

3.1.3. Módulo de teclado matricial

Desde el punto de vista eléctrico, cada tecla de un teclado es un mecanismo idéntico a un pulsador N/A. La aportación del teclado consiste en la configuración de las teclas para que necesiten pocas líneas de entrada en la detección de la que se ha presionado.

Para disminuir las líneas necesarias para detectar la tecla pulsada, éstas se agrupan de forma matricial en filas y columnas. Con esta configuración un teclado matricial de 16 teclas sólo precisa de 8 líneas del PIC para su gestión. Si cada tecla actuara como un pulsador individual se precisaría de 16 líneas de I/O del microcontrolador para gestionarlas.

En la figura 3.6 se muestra el diagrama esquemático de un teclado matricial.

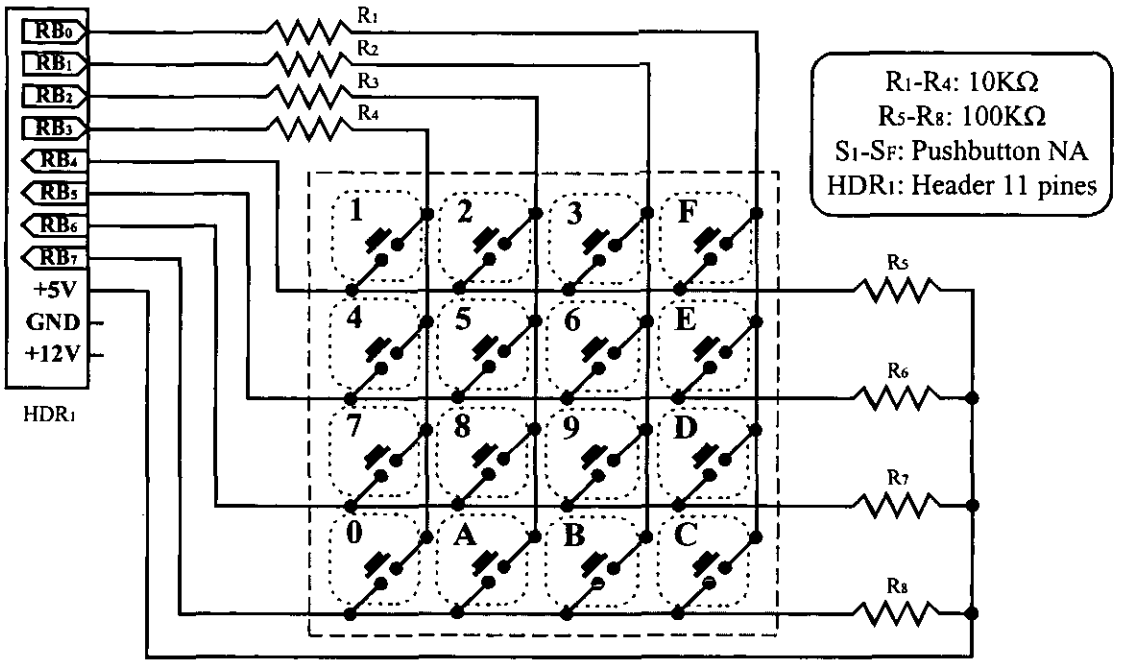


Figura 3.6 Conexión de un teclado matricial a un puerto de 8 líneas (Módulo de teclado matricial)

En la figura 3.6 las 4 líneas menos significativas del puerto B <RB0:RB3> se configuran como salidas que aplican un patrón de estados lógicos a las cuatro columnas del teclado. Las cuatro líneas más significativas <RB4:RB7> están configuradas como entradas y reciben los niveles lógicos que tienen las filas del teclado.

El programa que controla el funcionamiento del teclado saca un cero, secuencialmente, por una de las cuatro líneas de salida <RB0:RB3> que se aplica a las columnas, al mismo tiempo que lee el nivel lógico introducido por las filas en las líneas <RB4:RB7>. Si ninguna de las teclas de la columna por la que se introduce el nivel bajo está pulsada, se leerá un nivel alto en las 4 filas, pasándose a activar la siguiente columna.

Si se aplica en una columna un nivel bajo y al leer las filas una de ellas se encuentra en nivel bajo, se deduce que la tecla asociada a dicha fila y dicha columna se halla presionada. Así se determina la tecla que se presiona en cada momento.

Cada tecla tiene asociado un código binario que se muestra en la tabla 3.1 y que se corresponde con los 4 bits que sacan las líneas <RB0-RB3> y los 4 recibidos por <RB4-RB7>.

Los códigos de exploración de las teclas varían según los modelos y la colocación de las mismas. El programa debe realizar la exploración cada cierto tiempo, y suele ser de un valor aproximado de 20 ms.

También es el software el encargado de realizar el tratamiento oportuno cuando se pulsan varias teclas a la vez, eliminar los rebotes[18], implementar funciones de repetibilidad, etc.

Tecla	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	CODIGO
1	1	1	1	0	0	1	1	1	E7
2	1	1	1	0	1	0	1	1	EB
3	1	1	1	0	1	1	0	1	ED
F	1	1	1	0	1	1	1	0	EE
4	1	1	0	1	0	1	1	1	D7
5	1	1	0	1	1	0	1	1	DB
6	1	1	0	1	1	1	0	1	DD
E	1	1	0	1	1	1	1	0	DE
7	1	0	1	1	0	1	1	1	B7
8	1	0	1	1	1	0	1	1	BB
9	1	0	1	1	1	1	0	1	BD
D	1	0	1	1	1	1	1	0	BE
0	0	1	1	1	0	1	1	1	77
A	0	1	1	1	1	0	1	1	7B
B	0	1	1	1	1	1	0	1	7D
C	0	1	1	1	1	1	1	0	7E

Tabla 3.1 Códigos binarios y hexadecimales que corresponden a cada tecla del teclado matricial.

3.2. Módulos de salidas digitales.

La aplicación principal de un sistema con microcontrolador es precisamente controlar el funcionamiento de los diferentes componentes del sistema. Por ejemplo, pensemos en un sistema que controle el nivel de un líquido en un tanque, el sistema debe poder controlar el funcionamiento de una bomba, activándola para suministrar el líquido, cuando este haya disminuido por debajo de un nivel de referencia y desactivando la bomba para que el líquido no se desborde. Además el sistema puede proporcionar información al operador o usuario sobre el nivel del líquido en cada momento por medio de un indicador.

En las secciones que siguen se detalla el funcionamiento de algunos de los dispositivos de salida tanto de control, (relevadores, optoacopladores, etc.) como de monitoreo (LED's, displays, etc), agrupando uno o más de estos dispositivos en módulos independientes para que el estudiante comprenda las diferentes técnicas de control con microcontroladores PIC.

[18] Los rebotes son generados al conmutar un interruptor, y son voltajes transitorios que pueden provocar estados no deseados en los circuitos a los que están conectados.

3.2.1. Módulo de monitores (LED´s)

Los diodos emisores de luz o LED´s se utilizan frecuentemente en los circuitos digitales como monitores lógicos y para transmitir información de un circuito a otro por vía óptica. Un LED encendido representa, normalmente, un estado alto y un LED apagado un estado bajo. En la figura 3.7 se muestra el símbolo de los diodos LED. También se indica la forma de utilizarlo como monitor lógico.

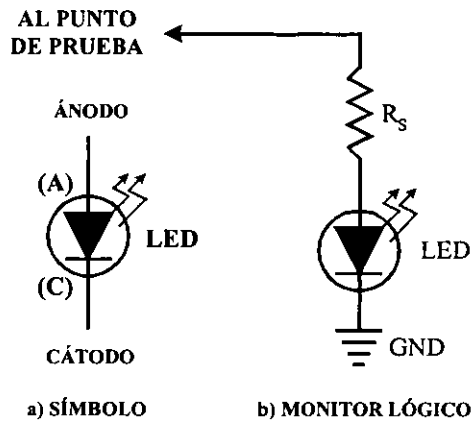


Figura 3.7 Símbolo y conexión de un LED como monitor lógico.

Los LED's son diodos que emiten luz (roja, amarilla, verde, etc.) cuando se polarizan en forma directa, es decir, cuando el ánodo es positivo y el cátodo es negativo.

En los LED's circulares la base posee una parte plana. La terminal situada de ese lado corresponde al cátodo. En los LED's rectangulares el cátodo se identifica por una marca o bisel en uno de sus bordes. En LED nuevos, el cátodo es la terminal más larga y la de mayor área cuando se observa hacia el interior de la cápsula.

La cantidad de luz emitida por un LED es directamente proporcional a la corriente que circula por el mismo. Esta corriente nunca debe ser superior al valor máximo especificado por el fabricante. Para evitar que esto suceda los LED's deben protegerse mediante una resistencia limitadora de corriente conectada en serie, como se muestra en la figura 3.8.

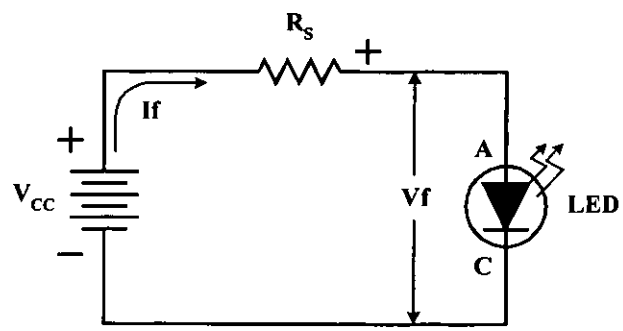


Figura 3.8 Circuito equivalente para calcular Rs.

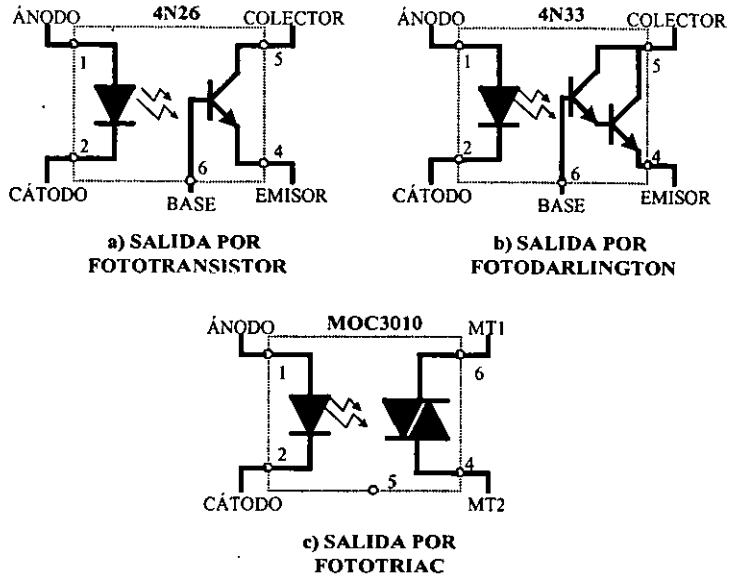


Figura 3.13 Configuraciones internas de los tipos más comunes de optoacopladores.

Para nuestro sistema se eligió utilizar dos tipos de optoacopladores, a transistor para cargas de CD y a triac para cargas de CA. El diagrama esquemático se da en la figura 3.14.

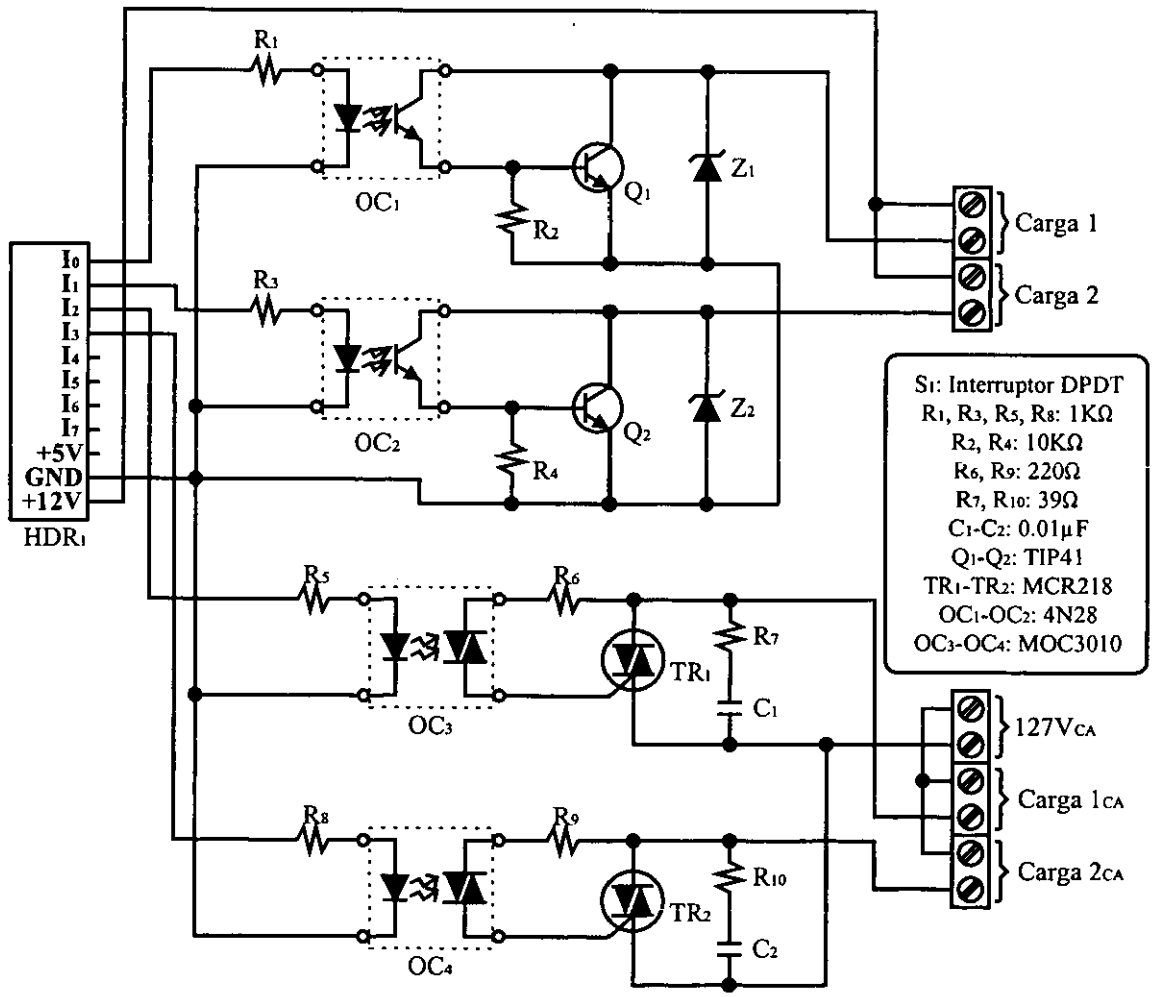


Figura 3.14 Módulo de optoacoplador.

3.2.4. Modulo de relevador

Un relevador es un dispositivo electromecánico muy utilizado en aplicaciones de control. Lo constituyen una bobina y varios contactos, unos normalmente abiertos (NA) y otros normalmente cerrados (NC) como se muestra en la figura 3.15.

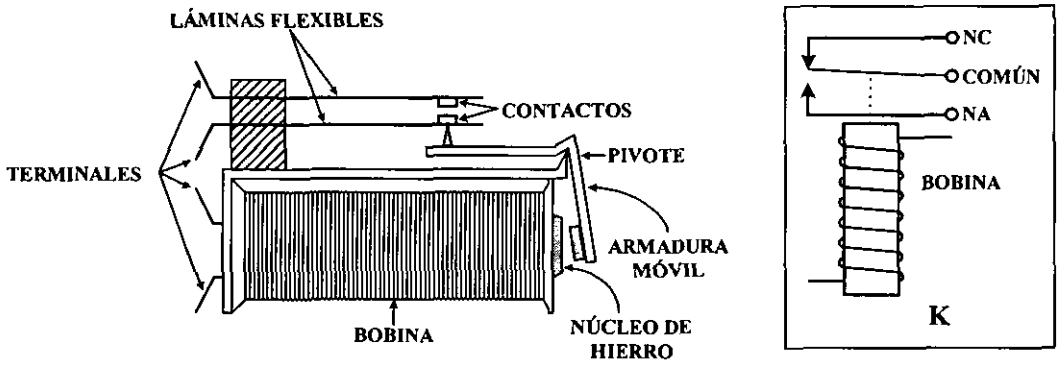


Figura 3.15 Aspecto físico y símbolo de un relevador.

Cuando se aplica un voltaje a la bobina, circula a través de ella una corriente, la cual crea un campo electromagnético que cambia el estado original de los contactos: los normalmente abiertos se cierran y los normalmente cerrados se abren. Cuando se suspende la corriente, los contactos vuelven a sus posiciones originales.

Los relevadores se utilizan en los circuitos digitales para aislar salidas que no tienen gran capacidad de corriente de cargas de potencia que operan a altos voltajes y/o consumen altas corrientes.

En la figura 3.16 se muestra la forma de excitar la bobina de un relevador mediante una salida que maneja niveles TTL utilizando un transistor de propósito general como elemento impulsor o *driver*. El diodo D1 elimina los picos de voltaje producidos por la bobina del relevador durante su operación.

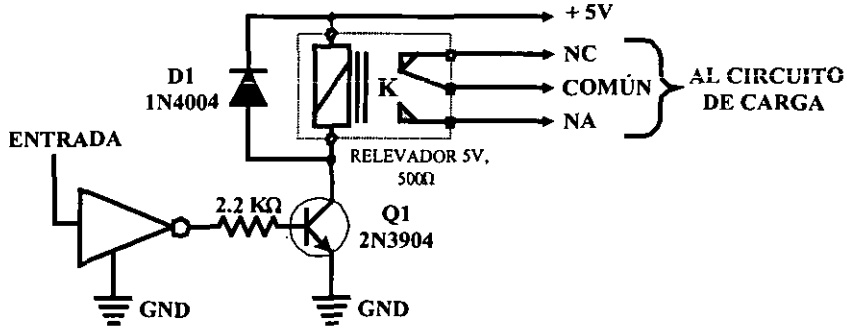


Figura 3.16 interfaz digital de relevador con transistor.

Cuando la salida del microcontrolador es alta, el transistor conduce y energiza la bobina del relevador (K). Como resultado, el contacto normalmente abierto (NA) se cierra y el normalmente cerrado (NC) se abre. Este efecto puede utilizarse para conectar o desconectar una carga externa, por ejemplo un motor.

ESTA TESIS NO SALE DE LA BIBLIOTECA

Cuando la salida del microcontrolador es baja, el transistor no conduce y no se energiza la bobina. En consecuencia, los contactos NA y NC permanecen en sus posiciones originales o retornan a estas.

En el sistema de entrenamiento se usarán 2 relevadores con sus respectivos circuitos de interfaz a transistor. El diagrama de dicho módulo se ve en la figura 3.17.

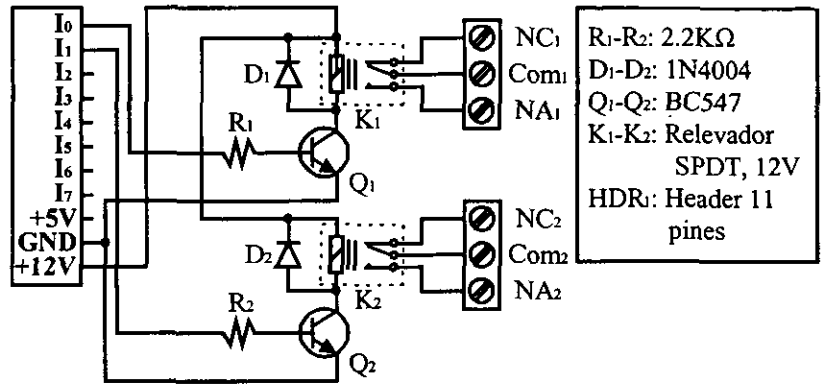


Figura 3.17 Módulo de relevador.

3.2.5. Módulo de pantalla (LCD)

Cuando se trabaja en diseño de circuitos electrónicos es frecuente encontrarse con la necesidad de visualizar un mensaje, que tiene que ver con el estado del sistema a controlar, con instrucciones para el operario, o si es un instrumento de medida, mostrar el valor registrado. En la mayoría de los casos, recurrimos a los *displays* de siete segmentos, pero estos además de no mostrar caracteres alfanuméricos ni ASCII, tienen un elevado consumo de corriente.

Uno de los métodos más eficientes de presentar información es el uso de una pantalla de cristal líquido (LCD). La LCD ofrece, entre otras características, un bajo consumo de potencia, ocupan muy poco espacio y son muy fáciles de interfazar con sistemas digitales.

La LCD opera bajo un principio diferente al de los displays con LED's. Cada segmento está hecho de un fluido viscoso que normalmente es transparente, pero se opaca (aparece oscuro) cuando se energiza mediante un pequeño voltaje de corriente alterna de baja frecuencia.

El voltaje alterno de excitación es generalmente una onda cuadrada de 3 a 15V de amplitud y de 25 a 60 Hz de frecuencia. Se aplica entre el pin de acceso al segmento (a, b, c, etc.) y un pin especial llamado *backplane*, que sustituye el terminal común (ánodo o cátodo) de los displays LED convencionales.

En la figura 3.18 se ilustra la estructura interna y el principio de funcionamiento de una pantalla de cristal líquido. En contraste con los LED's, la LCD no generan luz sino que simplemente controlan la luz incidente. La clave de su operación es el fluido especial llamado cristal líquido colocado entre dos láminas transparentes.

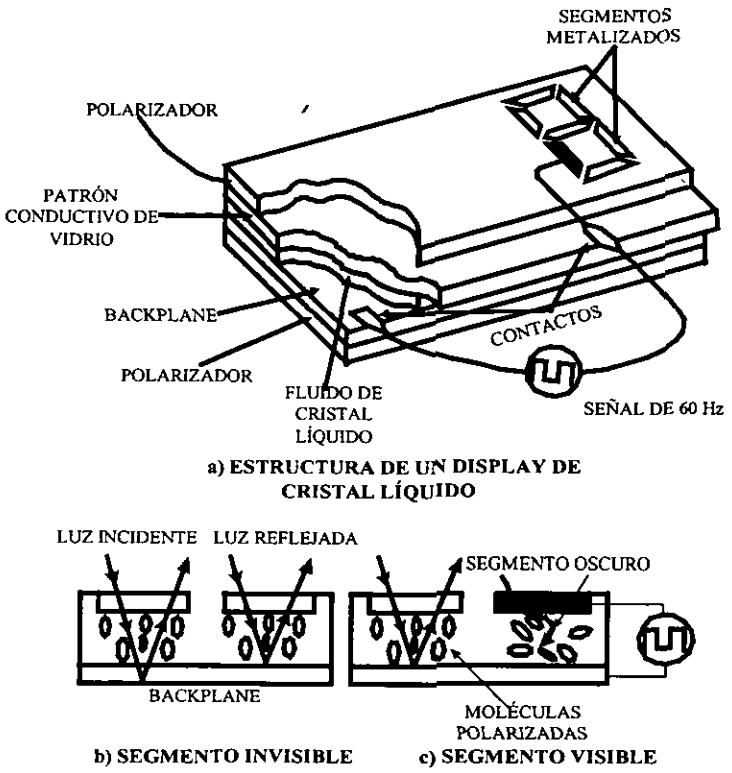


Figura 3.18 Funcionamiento de un *display* LCD.

Sobre la lámina superior se forman los segmentos del display, los cuales se metalizan para permitir que puedan ser controlados externamente. La lámina inferior o *backplane* actúa como una superficie reflectora de luz y también está metalizada.

En condiciones normales, las moléculas de cristal líquido están alineadas o polarizadas. Cuando incide luz en el sistema, esta pasa a través de las moléculas del fluido, se refleja en el *backplane* y retorna a la superficie sin sufrir cambio alguno. Como resultado, el segmento permanece brillante y aparece invisible al ojo humano.

Cuando se aplica un voltaje alterno entre el segmento y el *backplane*, las moléculas se dispersan y absorben la luz incidente, es decir, no la dejan pasar y por tanto el *backplane* no la refleja. Como resultado el segmento aparece oscuro. El mismo principio se aplica para hacer visible cualquier otro segmento y visualizar así números, letras, etc.

La LCD se utiliza extensamente en relojes, calculadoras, instrumentos y otras aplicaciones digitales. Su mayor ventaja es el bajo consumo de corriente y pueden ser leídos en presencia de luz brillante. Sin embargo presentan algunos inconvenientes.

Estos dispositivos no pueden ser leídos en la oscuridad. Por esta razón, algunos *displays* de este tipo incluyen una lámpara incandescente miniatura. Son muy sensibles a las bajas temperaturas; son muy delicados y tienden a ser lentos.

Partiendo de la sencilla pantalla de siete segmentos, la electrónica ha evolucionado rápidamente en este sector, creando nuevos dispositivos de mayor complejidad, capaces de presentar dinámicamente muchos datos a partir de informaciones obtenidas de microprocesadores y microcontroladores.

Dentro de esta nueva generación sobresalen los módulos con LCD programables. Estos dispositivos incluyen toda la lógica de control necesaria para visualizar dinámicamente caracteres alfanuméricos y símbolos especiales en una pantalla LCD de matriz de puntos incorporada.

Los mensajes se visualizan en la pantalla LCD una vez que se introduzcan los correspondientes códigos ASCII de cada uno de los caracteres a visualizar. Además, un extenso juego de caracteres predefinidos en fábrica permiten usar nuevos caracteres y símbolos definidos por el usuario. A pesar de que el número de modelos comerciales es muy grande, las líneas necesarias para su conexión y control son prácticamente las mismas.

A continuación haremos una descripción general de estos módulos desde un punto de vista operativo.

Los módulos LCD se encuentran en diferentes presentaciones: 2 líneas de 16 caracteres cada una (2x16), 2x20, 4x20, 4x40, etc. La forma de utilizarlos y sus interfases son similares, por eso, los conceptos vistos aquí se pueden emplear en cualquiera de ellos. En este trabajo, nos enfocaremos en un display de 2x16, ya que es de bajo costo, se consigue fácilmente y tiene un tamaño suficiente para la mayoría de las aplicaciones.

Algunos de estos dispositivos tienen luz posterior o *backlight*, para mejorar su visualización, está se maneja a través de dos pines que normalmente se conectan a +5V y a tierra. Para evitar que se presenten altas temperaturas, debido a la luz posterior, estos pines se deben manejar de manera pulsante (encendiendo y apagando), con una frecuencia aproximada a los 60 Hz. Otra opción mucho más sencilla es utilizar una resistencia de 10 ohms a ½ watt para alimentar el positivo del *backlight*.

Los pines de conexión de estos módulos incluyen un bus de datos de 8 bits, un pin de habilitación (E), un pin de selección (RS) que indica si el dato es una instrucción o un carácter del mensaje a desplegar y un pin que indica si se va a escribir o leer en el módulo LCD (R/W). En la tabla 3.2 se describe la función de cada uno de ellos.

Terminal	Símbolo	*Nombre y función*
1	Vss	Tierra, 0V.
2	Vdd	Alimentación +5V.
3	Vo	Ajuste de voltaje de contraste
4	RS	(Register Select) Selección Dato/Control
5	R/W	(Read/Write) Lectura/Escritura en LCD
6	E	(Enable) Habilitación
7	D0	(Data Bus) Bus de datos.
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	

Tabla 3.2. Función de los pines del módulo LCD.

Según la operación que se desee realizar sobre el módulo LCD, los pines de control E, RS y R/W deben tener un estado determinado. Además, debe tener en el bus de datos un código que indique un carácter para mostrar en la pantalla o una instrucción de control. En la figura 3.19 se muestra el diagrama de tiempos que se debe cumplir para manejar el módulo.

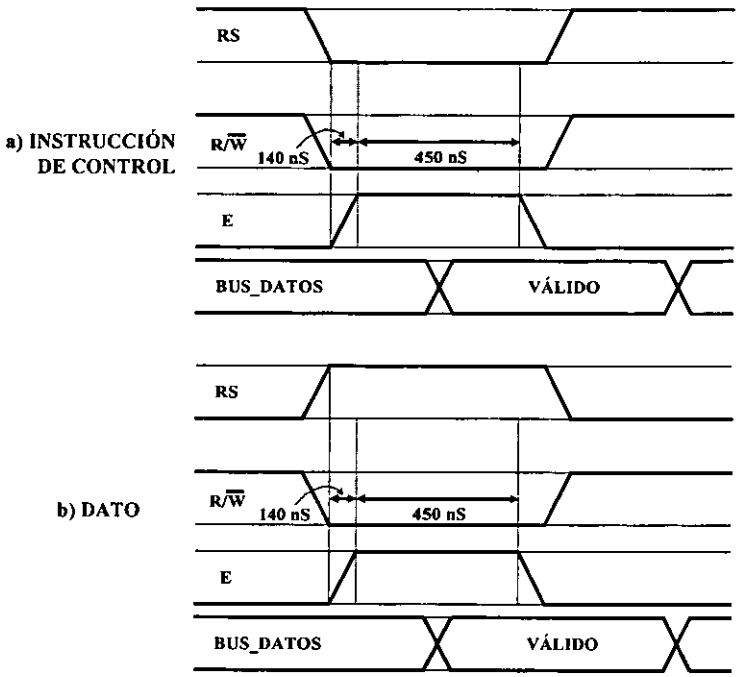


Figura 3.19. Diagrama de tiempos del módulo LCD.

La interfaz entre el microcontrolador y el display de cristal líquido se puede hacer con el bus de datos trabajando a 4 u 8 bits. Las señales de control trabajan de la misma forma en cualquiera de los dos casos, la diferencia se establece en el momento de iniciar el sistema, ya que existe una instrucción que permite establecer dicha configuración.

Los caracteres que se envían al módulo del display se almacenan su memoria RAM. Existen direcciones de memoria RAM, cuyos datos son visibles en la pantalla y otras que no lo son, estas ultimas se pueden utilizar para guardar caracteres que luego se desplazan hacia la parte visible. En la figura 3.20 se muestran las direcciones de memoria visibles y no visibles, que conforman las dos líneas de caracteres del módulo.

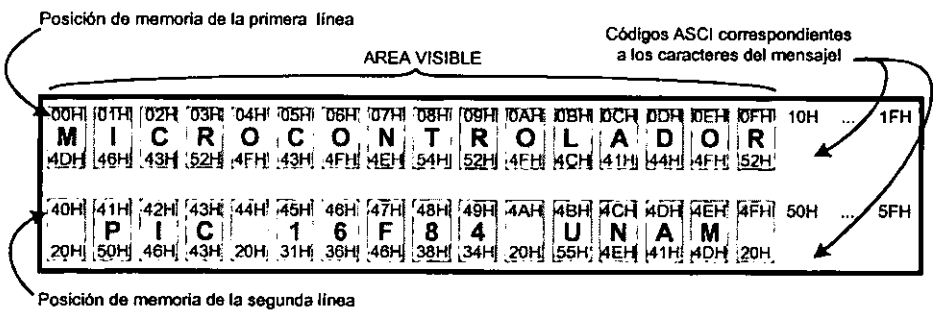


Figura 3.20. Mapa de memoria del módulo LCD

Es importante notar que solo se pueden mostrar caracteres ASCII de 7 bits, por lo tanto algunos caracteres especiales no se pueden ver (se debe tener a la mano una tabla de los caracteres ASCII para conocer los datos que son prohibidos). Por otra parte, se tiene la opción de crear caracteres especiales (creados por el programador), y almacenados en la memoria RAM que posee el módulo. En el apendice A se encuentra la tabla de los caracteres ASCII que pueden mostrarse en este tipo de displays.

Finalmente, en la figura 3.21 se muestra el diagrama esquemático del modulo con pantalla LCD. El dipswitch sirve para conmutar entre el funcionamiento a 8 y 4 bits.

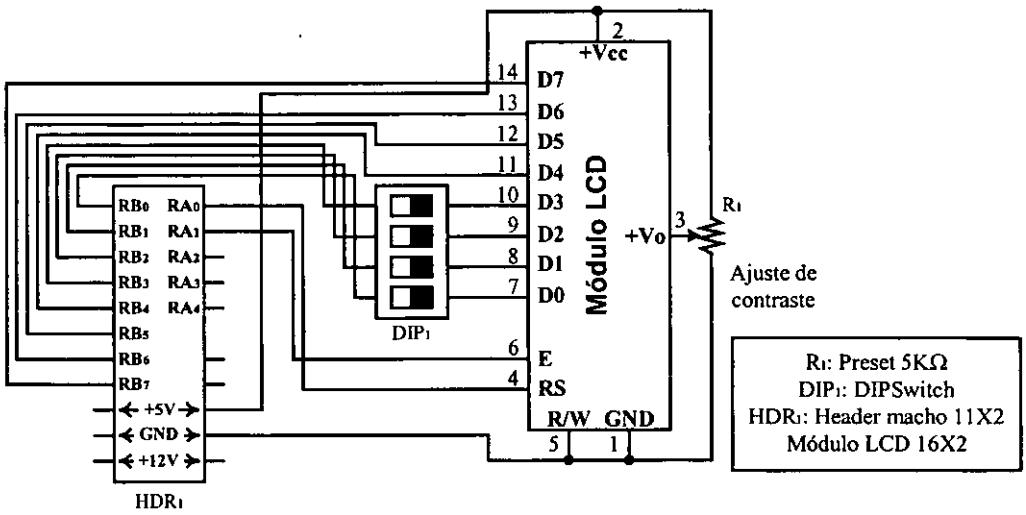


Figura 3.21 Modulo con pantalla LCD

El módulo LCD responde a un conjunto especial de instrucciones, estas deben ser enviadas por el microcontrolador o sistema de control al display, según la operación que se requiera. En la tabla 3.3 se muestran las instrucciones del módulo.

Instrucción	Señal de control		Dato / Dirección								Descripción
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Borrar pantalla	0	0	0	0	0	0	0	0	0	1	Limpia todo display y retorna el cursor a la posición inicio (dirección 0)
Cursor a casa	0	0	0	0	0	0	0	0	1	*	Retorna el cursor a la posición de inicio (dirección 0). También retorna el display, desplazándolo a la posición original. Los de la RAM DD permanecen sin cambio.
Seleccionar modo	0	0	0	0	0	0	0	1	I/D	0	Configura la dirección de movimiento del cursor (I/D=0: decrementa; I/D=1: incrementa) y si se desplaza o no el display. Esta operación es realizada durante operaciones de lectura y escritura.
Encender/apagar la pantalla	0	0	0	0	0	0	1	D	C	B	Configura el estado ON/OFF de todo el display (D=0: apagar; D=1: encender), el cursor (C=0: mensaje fijo en la pantalla; C=1. Desplaza el mensaje en la pantalla) y el parpadeo del caracter en la posición del cursor (B=0: no parpadea el caracter; B=1: parpadea el caracter señalado por el cursor)
Desplazar cursor/pantalla	0	0	0	0	0	1	S/C	R/L	*	*	Mueve el cursor (S/C=0: mueve cursor; S/C=1: desplaza pantalla) y desplaza el display (R/L=0: desplazamiento a la izquierda; R/L=1: desplazamiento a la derecha) sin cambiar los contenidos de la RAM DD.
Activar función	0	0	0	0	1	DL	N	F	*	*	Configura el tamaño de la interfaz (DL=1: datos de 8 bits, DL=0: datos de 4 bits), el número de líneas del display (N=0: 1 línea; N=1: 2 líneas) y la fuente del carácter (F).
CG RAM	0	0	0	1	Dirección generador de RAM						Ajusta la dirección del generador de caracteres. El dato CG RAM es enviado y recibido después de este ajuste.
DD RAM	0	0	1	Direcciones de datos RAM							Ajusta la dirección de la RAM DD. La dirección es enviada y recibida después de este ajuste.
Bandera de ocupado	0	0	BF	AC							Lectura de la bandera Busy Flag, indicando qué operaciones internas están siendo realizadas, y lectura de los contenidos del contador de direcciones.
Escritura CG RAM/DD RAM	1	0	Escritura de Dato								Escribe datos en la RAM DD o en la RAM CG.
Lectura CG RAM/DD RAM	1	0	Lectura de Dato								Lectura de datos desde la RAM DD o la RAM CG

Tabla 3.3. Conjunto de instrucciones de las LCD.

3.2.6. Módulo con motores paso a paso (PAP)

Los motores paso a paso son un tipo especial de motores que permiten el avance de su eje en ángulos muy precisos y por pasos en las dos posibles direcciones de movimiento, izquierda o derecha. Aplicando a ellos una determinada secuencia de señales digitales, avanzan por pasos hacia un lado u otro y se detienen exactamente en una determinada posición.

Cada paso tiene un ángulo muy preciso, determinado por la construcción del motor, lo que permite realizar movimientos exactos sin necesidad de un sistema de control por lazo cerrado.

A un motor paso a paso se le puede ordenar, por medio del control, que avance 5 o 10 pasos hacia adelante, luego un determinado número de pasos hacia atrás o simplemente que no gire. Este sistema ha simplificado enormemente la implementación de automatismos y las aplicaciones de la robótica.

Los motores paso a paso presentan grandes ventajas con respecto a la utilización de servomotores debido a que se pueden manejar digitalmente sin realimentación, su velocidad se puede controlar fácilmente, tienen una larga vida, son de bajo costo, la interfaz es sencilla y su mantenimiento es mínimo debido a que no tienen escobillas.

El funcionamiento de los motores paso a paso se basa en el simple principio de atracción y repulsión que ocurre entre los polos magnéticos. Como ya sabemos un imán tiene dos polos llamados Norte y Sur o N y S.

El principio básico del magnetismo establece que polos iguales se repelen y polos diferentes se atraen. En la figura 3.22 tenemos la ilustración de un motor paso a paso imaginario con dos bobinas y un rotor formado por un imán.

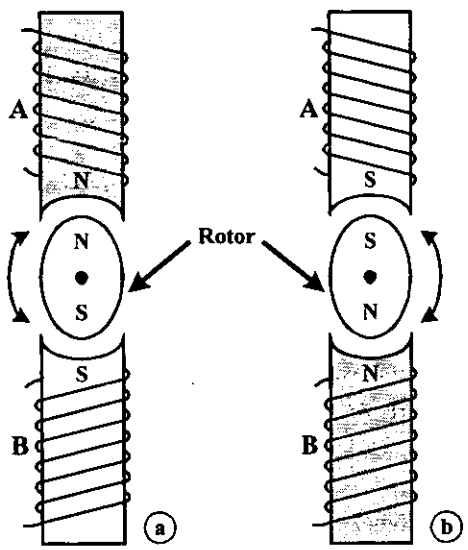


Figura 3.22 Principio básico del motor paso a paso.

Si aplicamos corriente a las bobinas A y B de tal manera que se formen electroimanes con las polaridades indicadas (a), el rotor gira hasta alcanzar la posición de reposo (b).

En la figura 3.23 tenemos la aproximación a un motor real utilizando cuatro bobinas mediante las cuales podemos hacer girar el rotor en ángulos de 90°. Al cambiar la polaridad de

las bobinas del estator se presenta el efecto de repulsión y atracción por parejas de polos con los polos del imán produciendo el giro por pasos.

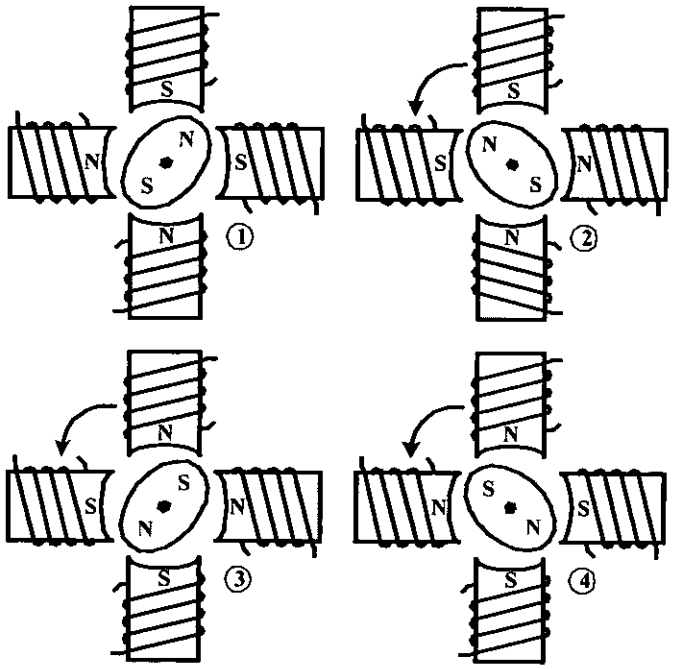


Figura 3.23 Principio de funcionamiento de un motor paso a paso.

Para lograr un movimiento mucho más suave, los motores paso a paso se fabrican aumentando el número de polos del estator y se les practican una serie de ranuras tanto en el rotor como en el estator. Así se logran movimientos que van desde 1.8° por paso. Los grados de avance por paso son una de las características más importantes en este tipo de motores y generalmente está indicada en su carcaza o cuerpo.

TIPOS DE MOTORES PASO A PASO

Según su construcción, hay tres tipos de motores PAP de imán permanente, de reluctancia variable e híbridos. En los primeros, su rotor es un imán permanente que está ranurado en toda su longitud y el estator está formado por una serie de bobinas enrolladas alrededor de un núcleo o polo. Su funcionamiento se basa en el principio explicado de atracción y repulsión de polos magnéticos.

En los de reluctancia variable el rotor está fabricado por un cilindro dentado de hierro y el estator está formado por bobinas que crean los polos magnéticos. Como este tipo no tiene un imán permanente, su rotor gira libremente cuando las bobinas no tiene corriente lo que puede ser inconveniente en un momento dado si hay una carga que presione el eje. Este tipo de motor puede trabajar a mayor velocidad que el anterior. Los híbridos combinan las dos características anteriores lográndose un alto rendimiento a buena velocidad.

En cuanto a la forma de conexión de las bobinas del estator, los motores PAP se dividen en dos tipos: unipolares y bipolares. En los unipolares hay cuatro bobinas y pueden tener cinco, seis y hasta ocho terminales. En el primer caso existe un cable que es común a las 4 bobinas, para el de 6 cables se tiene un común para cada pareja de bobinados y en el de 8 cables cada bobina es independiente. Los bipolares tienen dos bobinas sin toma media, es decir, tienen cuatro terminales.

Los motores unipolares tienen la ventaja de operar con una sola fuente, mientras que los motores bipolares requieren polaridad positiva y negativa, haciéndose necesario utilizar circuitos en puente.

MODOS DE OPERACIÓN

Los motores PAP, tanto unipolares como bipolares, pueden trabajar en dos modos de operación, de paso completo y de medio paso.

En el primer caso, con cada secuencia el rotor gira un determinado ángulo determinado por la fabricación del motor. En el modo de medio paso, cada secuencia produce un giro en grados correspondiente a la mitad de su paso normal. En las tablas 3.4 y 3.5 se muestra la secuencia de señales que se debe aplicar al motor en cada uno de los casos.

Q4	Q3	Q2	Q1
1	1	0	0
0	1	1	0
0	0	1	1
1	0	0	1

Tabla 3.4 Paso completo

Q4	Q3	Q2	Q1
1	0	0	1
1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1
0	0	0	1

Tabla 3.5 Medio paso

Además del sentido de giro y la posición determinados por la secuencia, también se puede controlar la velocidad de los motores PAP, dentro de cierto rango, variando la frecuencia de los pulsos aplicados a las bobinas.

CONTROL DE LOS MOTORES PAP

Siendo un microcontrolador un sistema en el que se puede enviar a través de un puerto, una secuencia de señales digitales ordenadas por un programa, resulta el dispositivo ideal para controlar uno o varios motores PAP.

Antes de saber las técnicas de programación para el control de un motor, debemos conocer cómo se implementa la interfaz adecuada para manejarlo. Esta interfaz debe tener la capacidad de manejo de corriente y de velocidad para excitar las bobinas.

Existen dos formas o modos de interfaz. Una con circuitos integrados especializados que requiere de pocas líneas de control por parte del microcontrolador y otra donde se maneja cada bobina con un bit independiente en un puerto de salida.

En este trabajo solamente abordaremos la interfaz con transistores ya que es una interfaz robusta, económica y sencilla de implementar y que no requiere ningún cuidado en el manejo de los componentes, cumpliendo con esto uno de los objetivos de la implementación de este sistema.

Para el módulo utilizaremos un motor unipolar de 6 terminales, con un paso de 7.5° (48 pasos por vuelta). Para el circuito de interfaz se utilizan 4 transistores TIP122, que es un darlington con protección para el manejo de cargas inductivas (tiene un diodo internamente). En la figura 3.24 puede verse el diagrama esquemático del módulo en donde además puede verse la manera de acoplar los transistores que conforman el circuito de interfaz.

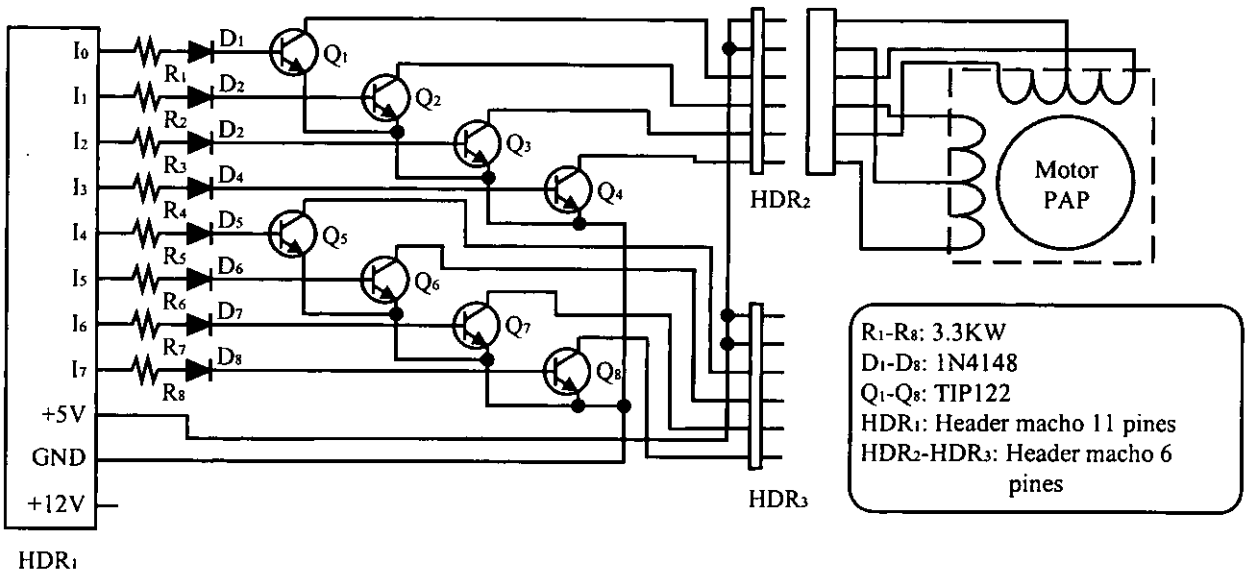


Figura 3.24 Módulo con motor paso a paso.

3.3. Módulos bidireccionales.

Le llamamos módulos bidireccionales a los circuitos que tienen la posibilidad de funcionar tanto como dispositivos de entrada como de salida de señales digitales ya sea de forma alternativa o simultáneamente.

En nuestro caso, se incluyen en esta sección el Módulo de interface serial RS-232 el cual se vale de un circuito especializado: el MAX232 y la Tarjeta principal que es el componente más importante de este trabajo, ya que es en donde se alojará el microcontrolador y permitirá llevar a cabo diversas prácticas y experimentos.

3.3.1. Módulo de comunicaciones RS-232

El puerto serial de las computadoras, conocido también como puerto RS-232, es muy útil ya que permite la comunicación no sólo con otras computadoras, sino también con otros dispositivos tales como el mouse, impresoras y por supuesto, microcontroladores.

Existen dos formas de intercambiar información binaria: la paralela y la serial. La comunicación paralela transmite todos los bits de un dato de manera simultánea y tiene la ventaja de que la transferencia es rápida, pero la desventaja es que necesita una gran cantidad de hilos o líneas, situación que eleva los costos y se agrava cuando las distancias que separan los equipos entre los cuales se hace el intercambio de datos es muy grande, debido a la capacitancia entre los conductores, la cual limita el correcto intercambio de datos a unos pocos metros.

La comunicación serial por su parte, transmite un bit a la vez, por lo cual es mucho más lenta, pero posee la ventaja de necesitar un menor número de líneas para la transferencia de la información y las distancias a las cuales se puede realizar el intercambio es mayor; a esto se suma que mediante dispositivos como los modem, la comunicación se puede extender prácticamente a cualquier lugar.

Comunicación serial síncrona y asíncrona.

Existen dos tipos de comunicación serial que son la comunicación síncrona y la asíncrona.

En la comunicación síncrona, además de una línea sobre la que se transfieren los datos, se necesita otra que contenga pulsos de reloj que indiquen cuando un dato es válido; la duración del bit está determinada por la duración del pulso de sincronía.

En la comunicación asíncrona, los pulsos de reloj no son necesarios y se acude a otros mecanismos para realizar la lectura/escritura de los datos; la duración de cada bit está determinada por la velocidad con la cual se realiza la transferencia de datos. *Para nuestro sistema de entrenamiento sólo trabajaremos con la comunicación asíncrona.*

La figura 3.25 muestra la estructura de un caracter que se transmite de forma asíncrona. Normalmente, cuando no se realiza ninguna transferencia de datos, la línea del transmisor es pasiva (*idle*) y permanece en un estado alto.

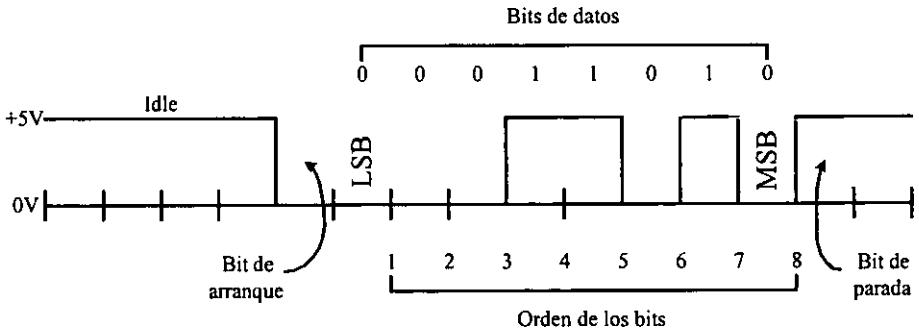


Figura 3.25 Estructura de transmisión asíncrona de un caracter.

Para empezar a transmitir datos, el transmisor coloca esta línea en bajo durante el tiempo de un bit, lo cual se conoce como bit de arranque (*start bit*) y a continuación, empieza a transmitir con el mismo intervalo de tiempo los bits correspondientes al dato (que pueden ser 7 u 8 bits), empezando por el menos significativo (LSB), y terminando con el más significativo (MSB). Al finalizar se agrega el bit de paridad (*parity*), si es que está activada esta opción, y los bits de parada (*Stop*) que pueden ser 1 o 2, en los cuales la línea regresa a un estado alto. Al concluir esta operación el transmisor estará preparado para transmitir el siguiente dato.

El receptor no esta sincronizado con el transmisor y desconoce cuando va a recibir datos. La transición de alto a bajo de la línea del transmisor activa al receptor y éste genera un conteo de tiempo de tal manera que realiza una lectura de la línea medio bit después del evento; si la lectura realizada es un estado alto, asume que la transición ocurrida fue ocasionada por ruido en la línea; si por el contrario, la lectura es un estado bajo, considera como válida la transición y empieza a realizar lecturas secuenciales a intervalos de un bit hasta conformar el dato transmitido.

El receptor puede tomar el bit de paridad para determinar la existencia o no de errores y realizar las acciones correspondientes, al igual que los bits de parada para situaciones similares. Lógicamente, tanto el transmisor como el receptor deberán tener los mismos parámetros de velocidad, paridad, número de bits del dato transmitido y de bits de parada.

En los circuitos digitales, cuyas distancias son relativamente cortas, se pueden manejar transmisiones en niveles lógicos TTL (0 – 5V), pero cuando las distancias aumentan, estas señales tienden a degradarse debido al efecto capacitivo de los conductores y su resistencia eléctrica. El efecto se incrementa a medida que se incrementa la velocidad de transmisión.

Todo esto origina que los datos recibidos no sean iguales a los transmitidos, lo que no se puede permitir en una transferencia de datos. Una de las soluciones más inmediatas en este tipo de situaciones es aumentar los márgenes de voltaje con que se transmiten los datos, de tal manera que las perturbaciones causadas se puedan minimizar e incluso ignorar.

Ante la gran variedad de equipos, sistemas y protocolos que existen surgió la necesidad de un acuerdo que permitiera que los equipos de varios fabricantes pudieran comunicarse entre sí. A principios de los años sesenta se desarrollaron varias normas que pretendían hacer compatibles los equipos, pero en 1962 se publicó la que se convirtió en la más popular: la norma RS-232. Esta norma define la interfaz mecánica, las características, los pines, las señales y os protocolos que debía cumplir la comunicación serial. La norma ha sufrido algunas revisiones, como la RS-232C en 1969 y la EIA/TIA-232E en 1991.

De todas maneras, todas las normas RS-232 cumplen básicamente con los mismos niveles de voltaje, como se puede observar en la figura 3.26:

- Un uno lógico es un voltaje comprendido entre –5V y –15V en el transmisor y entre –3V y –25V en el receptor.
- Un cero lógico es un voltaje comprendido entre 5V y 15V en el transmisor y entre 3V y 25V en el receptor.

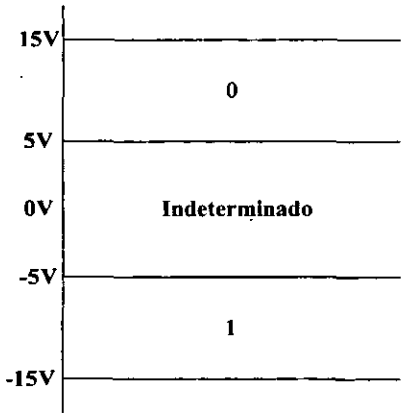
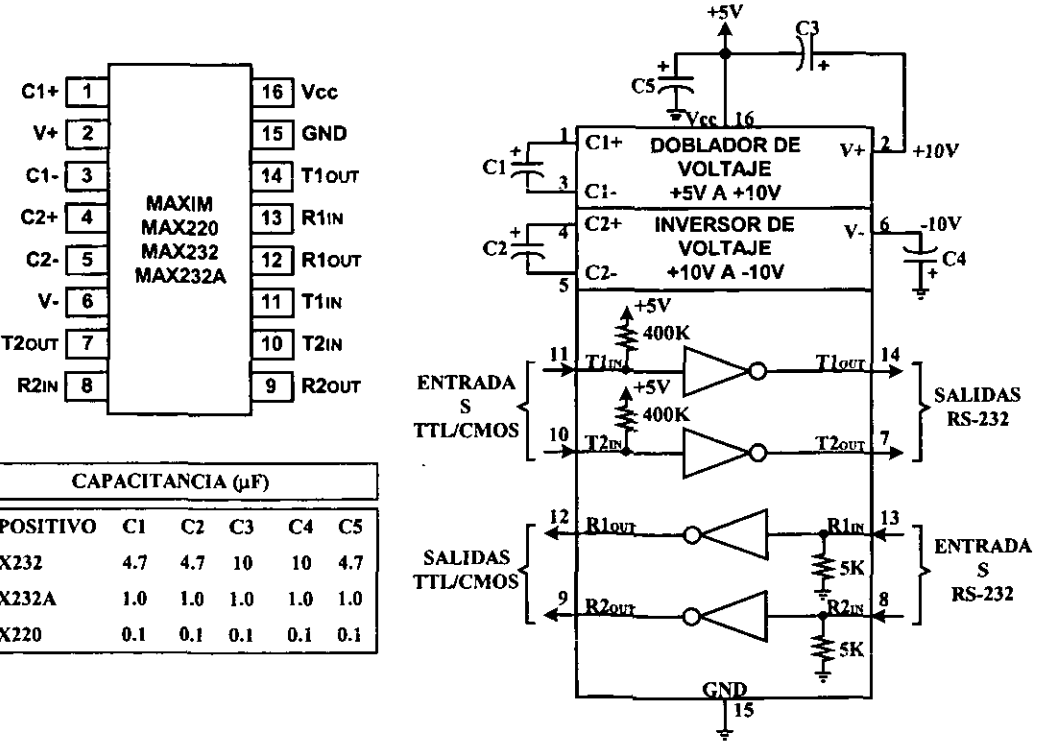


Figura 3.26 Niveles de voltaje RS-232

Por lo tanto deben existir dispositivos que permitan convertir niveles TTL a niveles RS-232 y viceversa. Para esto se han desarrollado alternativas muy útiles, como el circuito integrado MAX232.

Este dispositivo soluciona los problemas de niveles de voltaje cuando se requiere enviar señales digitales sobre una línea RS-232. El MAX232 se usa en aquellas aplicaciones donde no se dispone de fuentes dobles de ±12 volts; por ejemplo, en aplicaciones alimentadas con baterías de una sola polaridad. El MAX232 necesita solamente una fuente de +5V para su operación; un elevador de voltaje interno convierte el voltaje de +5V al de doble polaridad de ±12V.

Como la mayoría de las aplicaciones de RS-232 necesitan de un receptor y un emisor, el MAX232 incluye en un solo empaque 2 parejas completas de transmisor y receptor, como lo ilustra la estructura interna del circuito integrado que se muestra en la figura 3.27.



CAPACITANCIA (µF)					
DISPOSITIVO	C1	C2	C3	C4	C5
MAX232	4.7	4.7	10	10	4.7
MAX232A	1.0	1.0	1.0	1.0	1.0
MAX220	0.1	0.1	0.1	0.1	0.1

Figura 3.27 Diagrama de pines y estructura interna del MAX232

El MAX232 tiene un doblador de voltaje de +5V a +10V y un inversor de voltaje para obtener la polaridad de –10V. El primer convertidor utiliza el capacitor C1 para doblar los +5V de entrada a +10V sobre el condensador C3 en la salida positiva V+. El segundo convertidor usa el capacitor C2 para invertir +10V a –10V en el capacitor C4 de la salida V-. El valor mínimo de estos capacitores los sugiere el fabricante en el recuadro de la misma figura, aunque en la práctica casi siempre se utilizan capacitores de tantalio de 10μF.

En la tabla 3.6 se presentan algunas características de funcionamiento de este circuito integrado.

	LIMITES:		
Fuente de alimentación	-0.3 a +6V		
Voltajes de entrada:			
Tin	-0.3V a (Vcc - 0.3V)		
Rin	±30V		
Voltajes de salida:			
Tout	±15V		
Rout	-0.3V a (Vcc + 0.3V)		
Protección corto	Continua		
Disipación de potencia	842 mW		
	CARACTERISTICAS a Vcc=+5V, C1 – C4 = 0.1μF		
	Min.	Típ.	Máx.
TRANSMISOR			
Voltaje de salida (carga 3KΩ)	±5V	±8V	
Entrada BAJA		1.4V	0.8V
Entrada ALTA	2V	1.4V	
Velocidad		200Kb/seg.	
RECEPTOR			
Rango de entrada			±30V
Entrada BAJA	0.8V	1.3V	
Entrada ALTA		1.8V	2.4V
Resistencia de Entrada	3KΩ	5KΩ	7KΩ

Tabla 3.6 Características Eléctricas del MAX232 (Continuación)

En la figura 3.28 se ilustra el esquema del modulo de interfaz serial RS-232 empleando el MAX232.

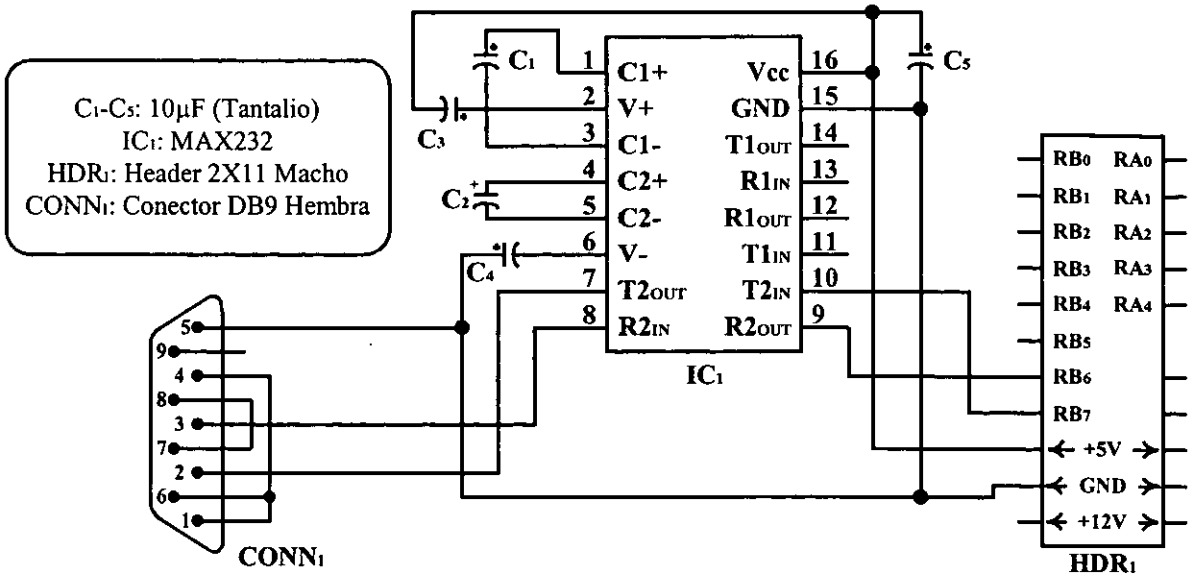


Figura 3.28 Diagrama del Módulo de interfaz serial RS-232

3.3.2. Modulo de grabación de PIC

Cuando se trabaja con microcontroladores, un componente muy importante es el grabador, que es el encargado de escribir el programa, previamente escrito y compilado vía software, en la memoria de programa del microcontrolador. Existen grabadores muy completos capaces de trabajar con muchos modelos de diferentes familias, como los que ofrece Microchip: el Pic Start plus® o el Pro Mate II®, pero su elevado precio los aleja de los usuarios personales, como nosotros los estudiantes.

Afortunadamente y gracias a que Microchip ha puesto a disposición, a través de su página en internet, las características y requerimientos para la grabación de sus microcontroladores es que existen varias versiones de sencillos grabadores, específicos para ciertos modelos de microcontroladores, que gobernados desde una PC se ofrecen por un precio ligeramente superior al de un libro, o también, se pueden encontrar en Internet los diagramas esquemáticos de grabadores puestos a disposición por diseñadores para quien quiera aprender a usar microcontroladores PIC.

Debido a nuestra experiencia y bajo la premisa de que este sistema de entrenamiento sea económico, para este trabajo se ha elegido un sencillo pero versátil grabador cuyo diseñador firma como JDM y que ha nombrado a su grabador como **PIC programmer 2**, también conocido como **LUDIPIPO**, este grabador está diseñado para grabar microcontroladores PIC de las familias: 12C5XX, 24CXX, 16C55X, 16C61, 16C62X, 16C71X, 16C8X, y **16F8X**. El diagrama de este grabador se puede ver en la figura 3.29.

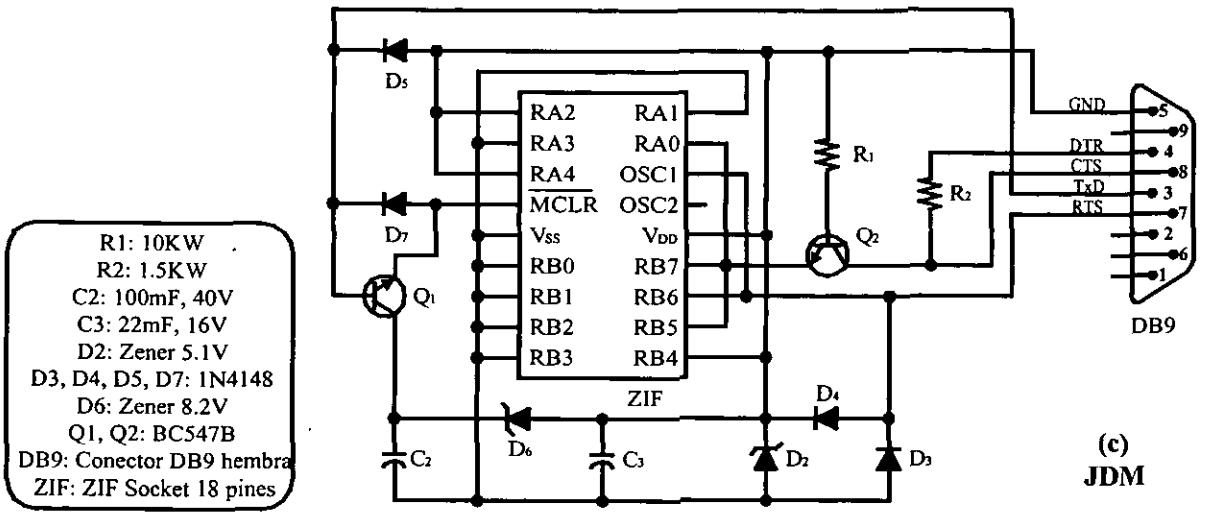


Figura 3.29 Diagrama esquemático del Módulo grabador de PIC (PIC-programmer 2)

La característica que hace más atractivo a este diseño es que no usa fuente de poder externa, puesto que se vale de los niveles de voltaje RS-232, visto en la sección anterior, para polarizar al microcontrolador y proporcionar el voltaje de programación (Vpp).

Las salidas RS232 están limitadas en corriente y esto protege al PIC en caso de que sea mal insertado en el grabador. El voltaje de entrada en el reloj (RB6) esta limitado por los diodos D3 y D4 sin necesidad de usar resistencias para limitar la corriente.

Cuando se efectúa una operación de lectura del microcontrolador el transistor Q2 amplifica el voltaje de salida a niveles RS232 funcionando entonces como base común. R2 esta conectada como pull-up. Q2 también limita el voltaje de entrada al PIC cuando la terminal DTR esta en alto durante una operación de escritura (grabación). Cuando esto sucede el transistor trabaja como un emisor seguidor y el voltaje de entrada es reducido a Vdd-0.7V.

El cambio de DTR a nivel bajo, hará trabajar Q2 invertido y que la amplificación sea solo de 5. La resistencia equivalente es de alrededor de 10KΩ/5=2KΩ. Esto reduce la corriente en la entrada de datos del PIC en conjunto con la resistencia R2.

Q1 es el encargado de conectar o desconectar el voltaje Vpp a la terminal MCLR. La terminal TxD del puerto serial carga el capacitor C2 hasta 13V a través del diodo de la juntura base-colector del transistor Q1. Dicho voltaje es limitado por el diodo zener D6 a ser de aproximadamente 5.1+8.2V=13.3V.

3.4. La tarjeta principal

La tarjeta principal que se muestra en la figura 3.30 es el corazón de nuestro sistema, en ella se inserta el microcontrolador, previamente grabado en el módulo de grabación, y se insertarán las diferentes tarjetas de los módulos que se requieran para llevar a cabo una práctica o un experimento, por lo que se le ha dotado de 4 puertos: 4 headers 11 X 2 hembras en los que se encuentran presentes las señales de todas las líneas de I/O del microcontrolador: <RA0:RA4> y <RB0:RB7> y voltajes de polarización comunes y especiales para el microcontrolador y los módulos (+5V, GND y +12V).

Para mayor comodidad, en esta misma tarjeta se encuentran implementados los circuitos indispensables para el funcionamiento de cualquier microcontrolador: un sencillo circuito de *reset* y el circuito de reloj, con base en un cristal oscilador de 4MHz, suficiente para la mayoría de las aplicaciones. Además, se ha reservado un espacio para una tableta de prototipos (protoboard) y se ha dispuesto un puerto especial para que el alumno tenga la posibilidad de tomar las señales necesarias para realizar proyectos o prácticas con componentes y/o configuraciones distintos a los empleados en los módulos del sistema

Asimismo, para cumplir con los objetivos del diseño, para que este sistema sirva en el entrenamiento de varios estudiantes, al igual que el Módulo de Grabación, la base para la inserción del microcontrolador es de tipo ZIF (Zero Input Force: Cero Fuerza de Inserción) para hacer más sencillo el cambio del chip de un estudiante por el chip de otro sin correr el peligro de dañar físicamente al microcontrolador.

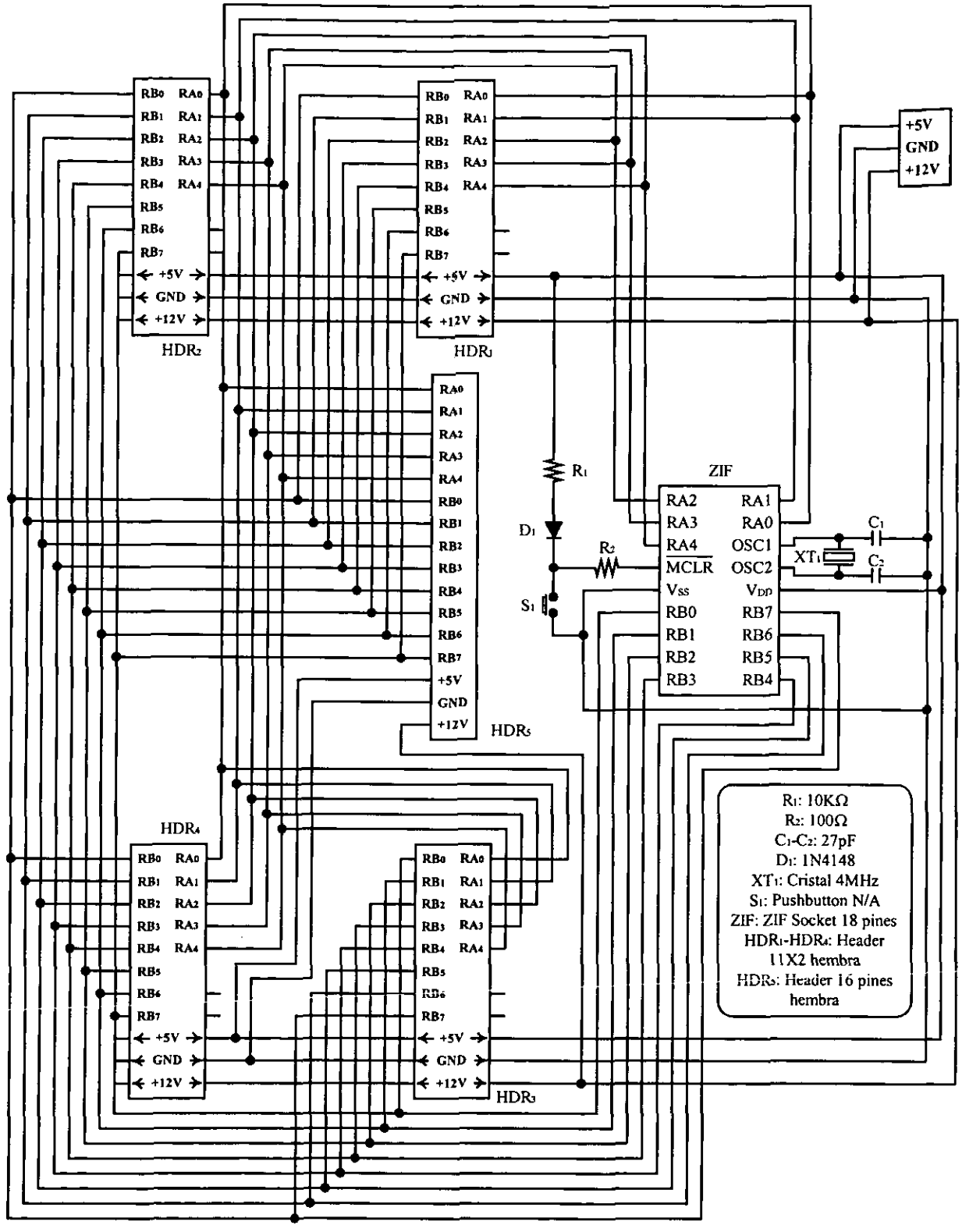


Figura 3.30 Diagrama esquemático de la Tarjeta principal.

3.5. La fuente de poder.

Una fuente de poder transforma la corriente alterna presente en los tomacorrientes (enchufes) de la red de distribución eléctrica, en corriente continua, apta para alimentar los diversos circuitos electrónicos. Debido al tipo de microcontrolador que se va a emplear, es necesario un voltaje estable y preciso ya que si el voltaje fuera menor el sistema podría no funcionar y si el voltaje es mayor se podría dañar al PIC. Por estas razones usaremos una fuente de poder regulada, que se muestra en la figura 3.31, la cual entrega +5V y +12V a 2 Amp.

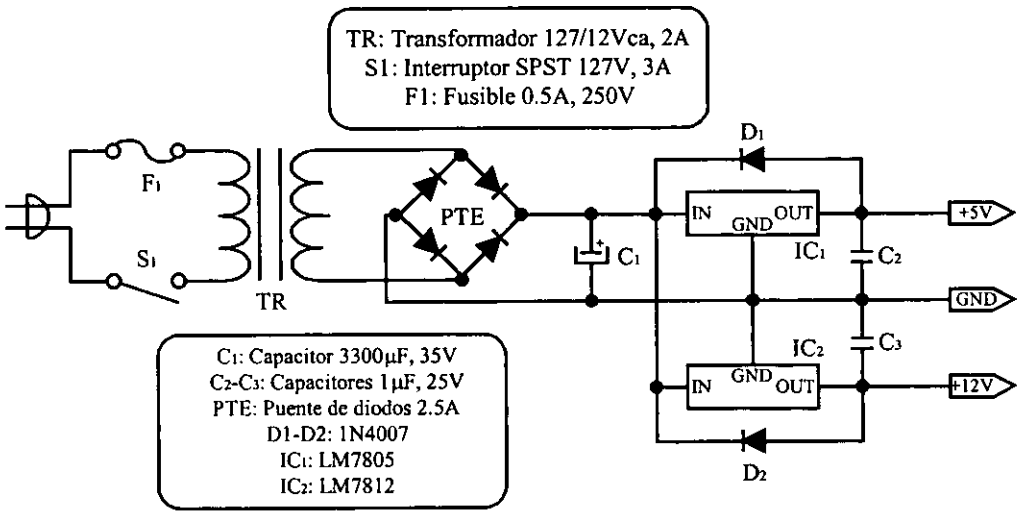


Figura 3.31 Diagrama esquemático de la fuente de poder.

El primer bloque contiene un transformador, que se encarga reducir el voltaje de la línea de alimentación de 117Vca a 15Vca. Enseguida de las terminales de bajo voltaje se encuentra conectado un puente rectificador, el cual se encarga de eliminar los semiciclos negativos de la señal de alterna, entregando una señal continua pulsante. El capacitor conectado a la salida del rectificador se encarga de eliminar las bruscas caídas de la señal, esto es, filtrando la señal, entregando una forma de onda continua con ligeras variaciones. Enseguida se encuentran los reguladores que proporcionan una señal plana de valor constante (+5 y +12 Vcd) independientemente de la carga conectada a sus terminales de salida.

CAPITULO 4

SOFTWARE DEL SISTEMA

El sistema de entrenamiento está formado por el conjunto de herramientas tanto del tipo hardware como software, que se necesitan para desarrollar un proyecto con microcontroladores, a nivel software las principales herramientas del sistema son las siguientes:

- Programa editor.
- Programa ensamblador.
- Programa simulador.
- Programa grabador.

Cada uno de ellos comprende una fase del desarrollo de un proyecto.

En la figura 4.1 se muestra el diagrama de flujo con las fases que compone el desarrollo de un proyecto.

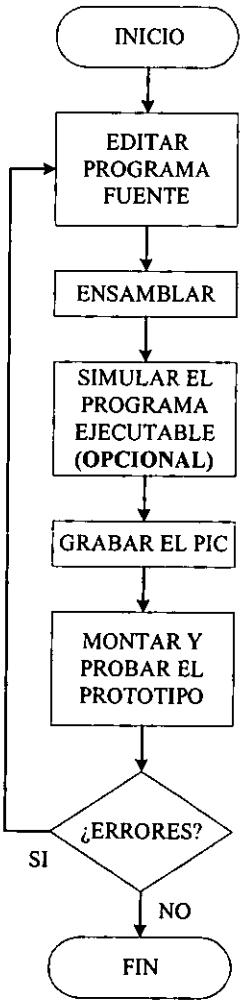


Figura 4.1 Diagrama de flujo del desarrollo de un proyecto.

La simulación es en realidad un proceso complejo y requiere la definición de una gran cantidad de parámetros, por lo que en base a nuestra experiencia se sugiere como opcional.

Para apoyar al diseñador en la creación de proyectos con microcontroladores el fabricante ha desarrollado un programa llamado MPLAB y le ha anexado las siglas IDE (Integrated Development Enviroment - Ambiente Integrado de Desarrollo), que es un software integral y lo compone:

- Un administrador de proyectos.
- Un editor.
- Un ensamblador.
- Un simulador.
- Un grabador.

Dicho software lo proporciona el fabricante Microchip gratuitamente y se puede obtener de su página en Internet en la dirección: www.microchip.com. Actualmente se encuentra la versión 5.11.02 que es la versión con la que trabajaremos.

El software MPLAB permite:

- Ensamblar, compilar y vincular el código fuente.
- Agrupar archivos fuente dentro de proyectos.
- Depurar el código fuente.
- Grabar dispositivos (sólo con los grabadores desarrollados por Microchip).

A través del administrador de proyectos se hacen las siguientes operaciones:

- Crear un proyecto.
- Añadir uno o varios archivos de código fuente a un proyecto.
- Ensamblar o compilar el código fuente.
- Editar el código fuente.
- Reconstruir todos los archivos fuente, o compilar un solo archivo.
- Depurar el código fuente.

El Editor de MPLAB es en realidad un editor de texto como el editor de MS-DOS o el Notepad de Windows y cuya característica especial es trabajar con el formato ASCII y permiten al diseñador escribir el programa, que es el que contiene las instrucciones pero en lenguaje ensamblador propio del microcontrolador. El programa o código fuente está contenido dentro de un archivo fuente.

El ensamblador que contiene el MPLAB se llama MPASM y permite generar varios formatos de código objeto que soportan las herramientas de desarrollo de Microchip así como los grabadores desarrollados por el fabricante sin salir de MPLAB.

El simulador de MPLAB se llama MPLAB-SIM y es un simulador de eventos para las familias de microcontroladores PIC y que, al contrario de un emulador, está diseñado para depurar software. MPLAB-SIM simula las funciones principales así como la mayoría de los estímulos en puertos de las familias de microcontroladores PIC16/17.

MPLAB tiene integrado un programa grabador, que es compatible con los grabadores hardware que manufactura Microchip, como son el Pic Start 16C® , el Pic Start 16B®, el Pic Start Plus® y Pro Mate II® y con algunos grabadores ajenos a la empresa Microchip.

Si se requiere información detallada del empleo de MPLAB se recomienda dirigirse a la página en Internet de Microchip en la dirección www.microchip.com en donde se encuentra el manual completo.

Los requerimientos mínimos para emplear MPLAB son los siguientes:

- PC compatible 486 o superior, (Pentium recomendado).
- Microsoft Windows 3.1x® o Windows 95/98®.
- Pantalla VGA (SVGA recomendada).
- 8 MB de memoria RAM (32 recomendada).
- 20 MB de espacio libre en Disco Duro.
- Ratón u otro dispositivo puntero.

Una vez que se instala y se entra a MPLAB aparece la ventana principal como se muestra en la figura 4.2.

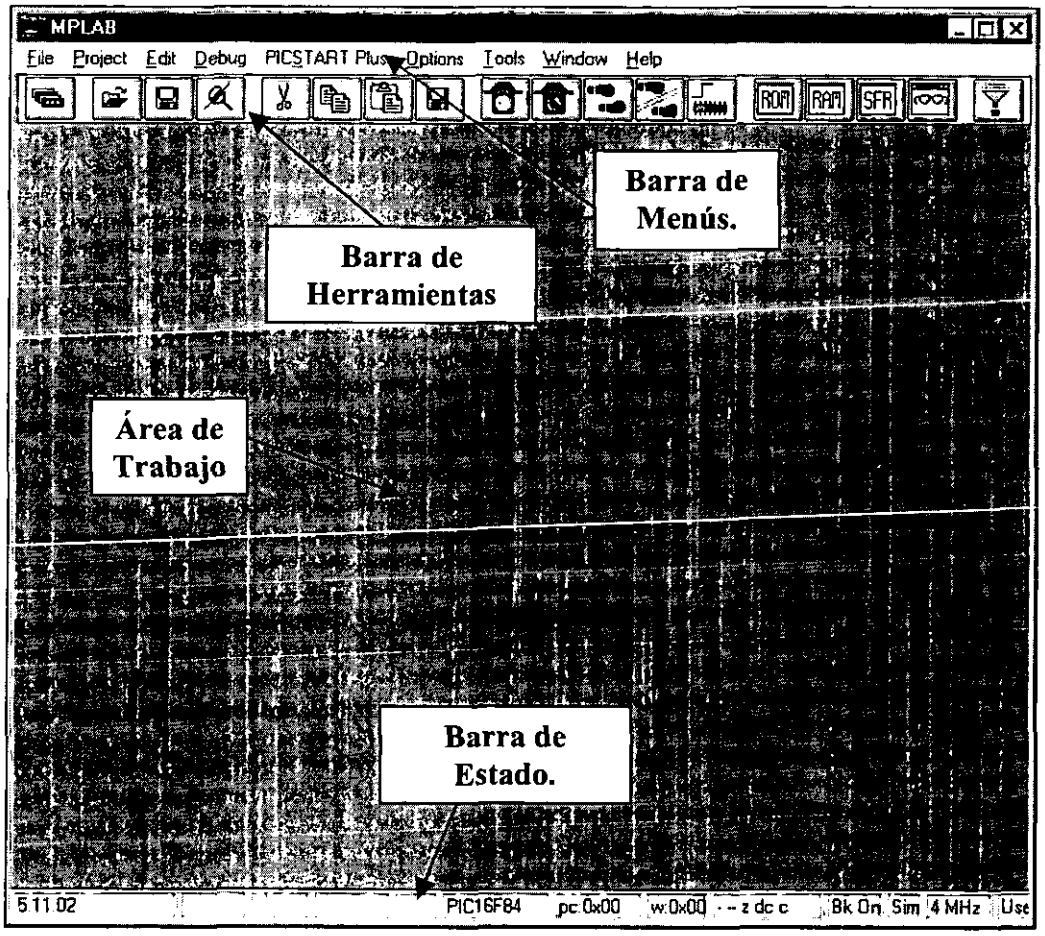


Figura 4.2 Ventana principal del software MPLAB-IDE.

Dentro de esta ventana principal existen cuatro áreas específicas: Barra de Menú, Barra de Herramientas, Área de Trabajo y Barra de Estado. Donde la primera permite el manejo operaciones vinculadas a archivos, proyectos, edición, depuración, grabadores, etc. por ejemplo el menú File habilita las operaciones que se realizan con archivos como Crear un nuevo archivo, Abrir, Renombrar, Importar archivos de otro formato, etc. La Barra de Herramientas contiene iconos que en realidad son accesos rápidos a las funciones más usuales de la barra de Menús, por ejemplo el icono  accesa a las opciones de File y Open directamente, facilitando con esto el abrir un archivo existente. El Área de trabajo es donde se abrirán las ventanas derivadas de la ventana principal, así por ejemplo la ventana del Editor estará contenida dentro de esta área. La barra de Estado nos proporciona información de la configuración de MPLAB, por ejemplo el microcontrolador, la velocidad del oscilador, el estado del Contador de Programa, etc.

El archivo fuente

Antes de iniciar la creación de un archivo fuente es conveniente definir algunos estándares en las extensiones de los archivos establecidos por el fabricante y que se manejan en toda la literatura relacionada con el tema, éstos facilitan la revisión y depuración de un programa, algunos que usaremos son:

- Los archivos del código fuente llevarán la extensión **.ASM*.
- Los archivos de listado llevarán la extensión **.LST*.
- Los archivos de código objeto llevarán la extensión **.OBJ*.
- Los archivos ejecutables en formato Intel Hex tendrán la extensión **.HEX*.
- Los archivos de referencias cruzadas llevarán la extensión **.XRF*.
- Los archivos de errores de ensamblado llevarán la extensión **.ERR*.

El archivo fuente es por definición aquel que contiene el programa o código fuente, y se denomina fuente debido a que de él se deriva todo el proceso de la creación de proyectos con microcontroladores.

Para iniciar la creación de un archivo fuente se pulsa con el ratón sobre la barra menú en File después se selecciona New o bien la combinación de teclas Ctrl + N.

Se mostrará un mensaje como el de la figura 4.3 solicitándole la creación de un nuevo proyecto.

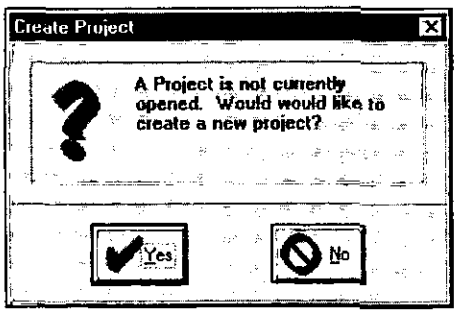


Figura 4.3 Mensaje para la creación de un nuevo proyecto.

Pulse en el botón **NO** y la ventana será como la de la figura 4.4. Ahora se ha abierto la ventana del Editor y se podrá iniciar la escritura del código fuente.

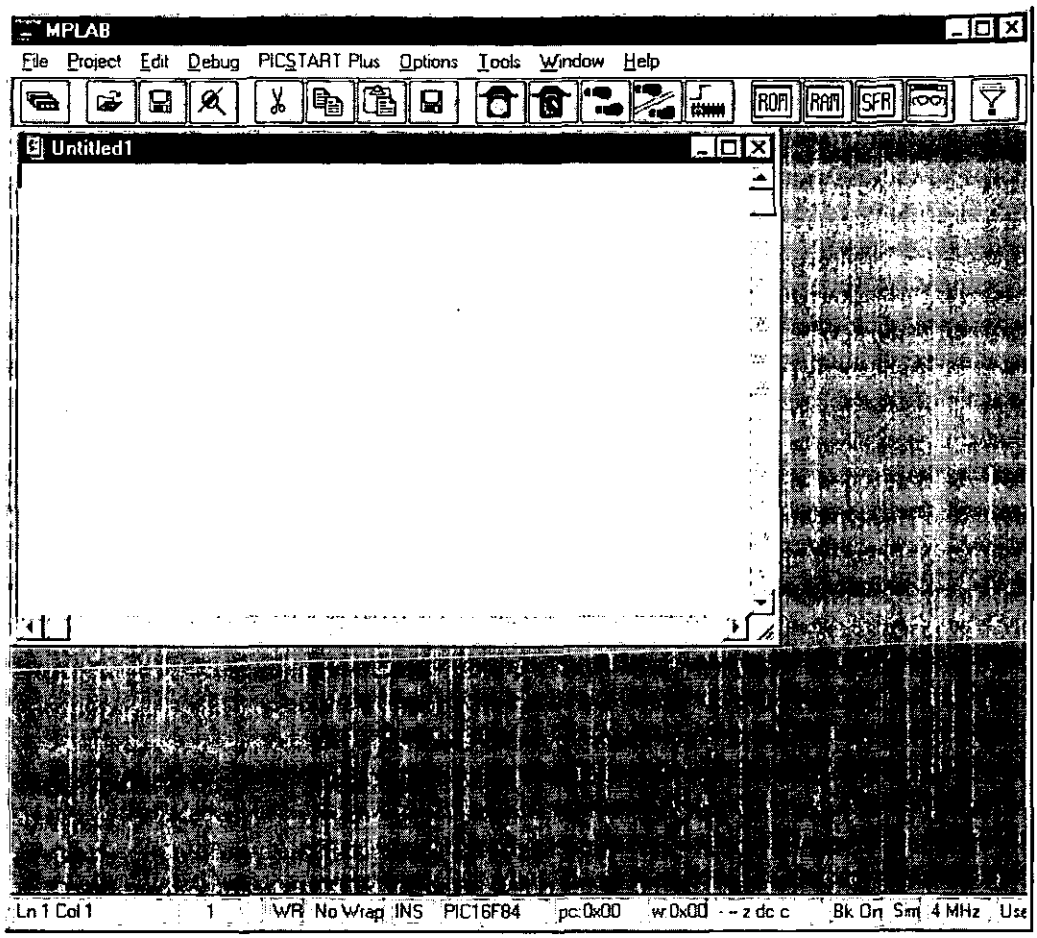


Figura 4.4 Ventana del Editor para la escritura del código fuente.

4.1. Escritura del código fuente

El código fuente está estructurado verticalmente en columnas llamadas campos y horizontalmente en grupos de renglones que conforman áreas específicas, es decir por ejemplo que cualquier texto que comience en la primera columna y se encuentre en el área de programa se considera una etiqueta y es una parte del campo de etiquetas.

Las siguientes tres columnas forman al campo de instrucciones, el campo de datos y el campo de comentarios respectivamente. Los comentarios deben empezar con punto y coma (;) y pueden ir en cualquier campo del código fuente.

De igual forma horizontalmente se nombran las áreas de encabezado, directivas y programa respectivamente y se recomienda dividirlas con asteriscos (*).

A continuación se describen las características de cada campo y de las áreas.

Campo de etiquetas

Las etiquetas son nombres de subrutinas o secciones del código fuente y sirven para nombrar a las partes del programa, así se posibilita que las instrucciones puedan saltar o hacer referencia a esas partes del programa sin necesidad de recordar las direcciones donde están ubicadas.

Muchos ensambladores establecen un límite pequeño al tamaño de las etiquetas así como los caracteres que pueden usarse en ellas. El ensamblador MPASM permite etiquetas de hasta 32 caracteres y pueden ir seguidas de dos puntos (:), espacios o tabuladores. Deben empezar por un carácter alfanumérico o de guión bajo (_) y puede contener cualquier combinación de caracteres alfanuméricos.

En la figura 4.5 se muestra la ventana del archivo OUTPUT.ASM donde **Inicializa**, **Principal**, y **Terminado** son tres etiquetas y para que sean consideradas como tal deben encontrarse ubicadas en el área de programa.

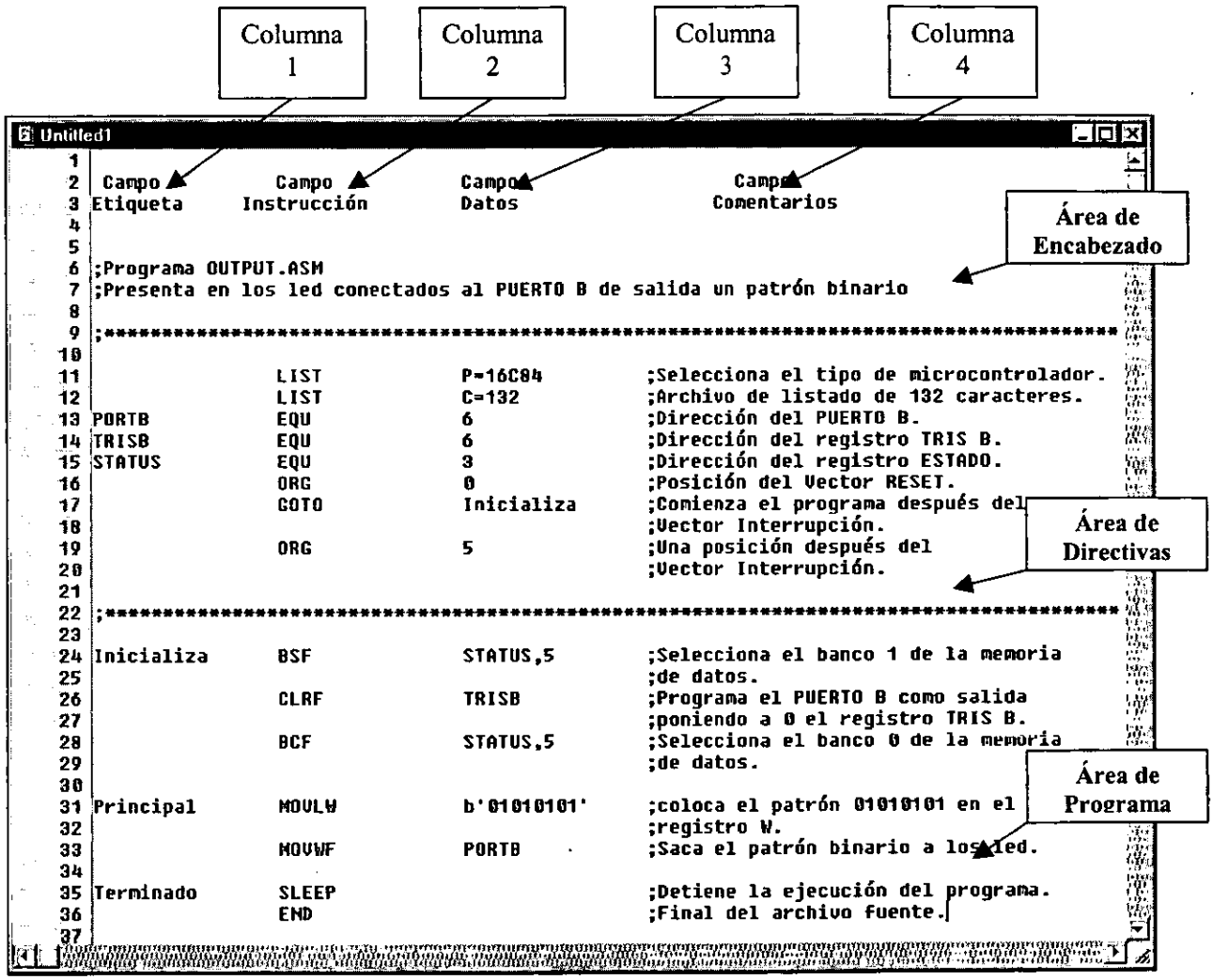


Figura 4.5 Estructura del código fuente del archivo OUTPUT.ASM.

Campo de instrucciones

Corresponde a la segunda columna del código fuente como se mostró en la figura 4.5, y puede contener una instrucción del microcontrolador o una instrucción para el ensamblador, denominada directiva, según el área en que se encuentre.

Campo de datos

Le corresponde la tercera columna y contiene datos u operandos para las instrucciones. En el caso de los microcontroladores PIC, los datos pueden ser un registro, un bit de un registro, una etiqueta, o un número constante. Algunas instrucciones no llevan datos. Si una instrucción necesita múltiples datos, deben separarse por comas (,).

La especificación de la base en que se expresan los datos, números u operandos es opcional, en la tabla 4.1 se muestra un ejemplo de prefijos empleados para expresar una numeración para diferentes bases.

Número	Base	Prefijo	Notación
65	Decimal	d, D	d'65'
65	Binario	b, B	b'01000001'
65	Hexadecimal	0x, 0X	0x41
65	Octal	o, O	o'41'

Tabla 4.1 Notación de numeración para diferentes bases.

Campo de comentarios

El último campo es el de comentarios, y siempre tiene que anteponerse un punto y coma (;). Un comentario puede colocarse en cualquier campo del código fuente.

Los comentarios describen la tarea que realizan determinadas instrucciones o subrutinas. Sin comentarios se dificulta entender la operación del código fuente.

Área de encabezado

Primeramente, al inicio de todo archivo fuente, en el área de encabezado, se suelen escribir una serie de comentarios que indican el nombre del mismo, así como una descripción sobre el trabajo que va a realizar el microcontrolador.

Área de directivas

Las expresiones que veremos a continuación son directivas para el ensamblador. Pueden variar de un ensamblador a otro según el fabricante, aunque suelen ser bastante similares entre sí.

Una directiva es un comando escrito en el código fuente para realizar un control directo, o bien para ahorrar tiempo al momento de ensamblar. El resultado de incorporar directivas se puede ver en el archivo **.LST* que se genera después de ensamblar el archivo.

La directiva LIST dispone de diferentes opciones que permiten elegir el tipo de procesador a emplear (P), número de caracteres por línea (C), tamaño de los tabuladores (B), base de numeración por omisión (R), niveles de mensajes de salida (W), entre otros.

Las siguientes directivas que encontramos en el código fuente de la figura 4.5 son las **EQU**. Éstas asignan valores a las etiquetas deseadas. Así por ejemplo PORTB tiene asignado el valor 6 que corresponde a la dirección de la memoria de datos del PUERTO B ubicado en el banco 0, STATUS la dirección 3 y TRISB el valor 6, correspondiente a la dirección de la memoria de datos del registro TRIS B ubicado en el banco 1.

La siguiente directiva ORG indica al ensamblador donde debe comenzar a colocar las instrucciones en la memoria de programa. Es decir, indica el ORiGen para todo el código que sigue. La dirección de inicio es la posición 0, debido a que la familia de microcontroladores PIC de gama media, después de encendido o bien de un RESET siempre ejecutan la instrucción ubicada en la dirección 00H, y se denomina Vector Reset.

La dirección 04H es el Vector Interrupción. Si se genera una interrupción el microcontrolador ejecuta la instrucción que se encuentra aquí. Es bueno por lo tanto dejar libre la dirección 04H por si más adelante se desea añadir la capacidad del manejo de interrupciones al programa y comenzar en la dirección 05H.

La instrucción GOTO Inicializa ha sido colocada en la dirección 00H, que es la que sigue a la directiva ORG 0. La primera instrucción ejecutada por el microcontrolador cuando se enciende o después de un RESET no es ORG 0, sino GOTO Inicializa porque ORG 0 es una directiva del ensamblador, no es una instrucción del microcontrolador. En la tabla 4.2 se muestra el mapa de memoria con las instrucciones tal y como quedan ubicadas.

MEMORIA DE PROGRAMA	
DIRECCIÓN	INSTRUCCIÓN
0000	GOTO 5 (Vector Reset)
0001	----------
0002	----------
0003	----------
0004	Vector Interrupción
0005	BSF STATUS,5
0006	CLRF TRISB
0007	BCF STATUS,5
0008	MOVLW 01010101
0009	MOVWF PORTB
000A	SLEEP
000B	----------
000C	----------
000D	----------
----	----------
----	----------
03FE	----------
03FF	----------

Tabla 4.2 Distribución del mapa de memoria del programa OUTPUT.ASM.

Aunque el ensamblador podría colocar la instrucción que sigue a GOTO Inicializa en la dirección 1, en nuestro caso el microcontrolador ha saltado a una dirección apuntada por la etiqueta Inicializa. Como ORG 5 está antes a dicha etiqueta, la posición de memoria 5 contiene la instrucción BSF STATUS,5 y es la dirección de la siguiente instrucción a ejecutar, es decir, que ahora el origen queda etiquetado en la dirección 5 y a partir de esta dirección se empieza a estructurar el programa.

Área de programa

En esta área es donde se ubican todas las etiquetas, instrucciones y operandos del código fuente que, como ya mencionamos, están en lenguaje ensamblador y son propios del microcontrolador.

4.2. Ensamblado del programa fuente

Las instrucciones del código fuente del ensamblador son comprensibles para el diseñador, pero el microcontrolador sólo entiende los números binarios. La función de un programa ensamblador es convertir el texto del código fuente en el equivalente de lenguaje máquina numérico del microcontrolador.

El siguiente paso en el proceso de desarrollo de un proyecto es ensamblar el archivo que contiene el código fuente y producir un archivo de código ejecutable. Una vez creado el ejecutable, ya se puede proceder a grabar el microcontrolador.

El programa ensamblador MPASM convierte el código fuente en código ejecutable en dos pasos. En el primero comprueba la correcta sintaxis de las instrucciones, los nombres de las etiquetas duplicados y asigna valores a los símbolos. En el segundo paso el ensamblador convierte todas las instrucciones en sus respectivos códigos máquina.

MPASM es parte de MPLAB, y se instala automáticamente cuando se instala MPLAB, la ventana principal es la que se muestra en la figura 4.6.

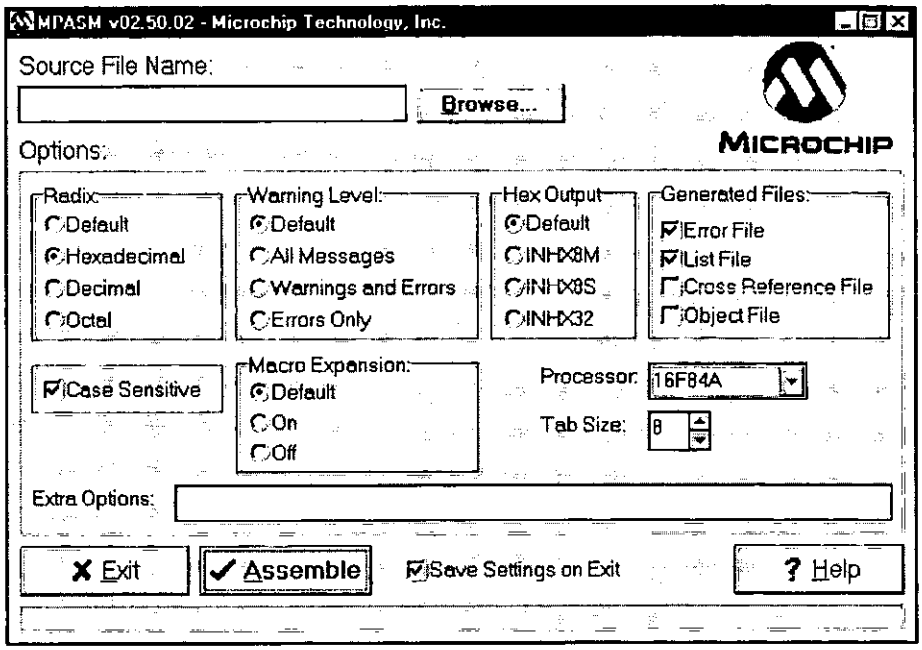


Figura 4.6 Ventana principal del ensamblador MPASM.

Como se puede apreciar en la figura 4.6 existen varias opciones de las cuales sólo se describen las más importantes.

En el campo *Source File Name* se debe escribir la ruta del archivo fuente, que debe tener la extensión **.ASM*, o bien pulsando en el botón *Browse* se pueden explorar tanto directorios, como unidades de disco disponibles dentro de la PC, para localizar el archivo fuente.

Radix: Configura la numeración base del archivo fuente, sin embargo ignora cualquier configuración incluida en el código fuente.

Warning level: Configura el nivel de alertas, mensajes y errores del archivo fuente, sin embargo ignora cualquier configuración incluida en el código fuente.

Hex output: Determina el tipo de archivo ejecutable que se obtiene por omisión. Este archivo, que tiene el mismo nombre que el archivo fuente pero con la extensión **.HEX*, es el ejecutable que servirá para la posterior programación del microcontrolador. El formato por omisión es el INTEL-HEX de 8 bits (INHX8M).

Generated Files: Habilita la creación de diferentes tipos de archivos. Habilitar la casilla de selección **Error File**, generará al momento de compilar un archivo con el mismo nombre del archivo fuente pero con la extensión **.ERR*, al habilitar la casilla **List File** se generará un archivo de extensión **.LST*, al habilitar la casilla **Cross Reference File** se generará un archivo con la extensión **.XRF* y el habilitar la casilla **Objet File** generará un archivo con la extensión **.OBJ*.

Processor: Es opcional, si no se especifica el tipo de microcontrolador se toma el que se definió mediante la directiva LIST P= del código fuente.

Cabe mencionar, que los archivos con extensión *.XRF y *.OBJ se emplean sólo en el caso de que se cree un proyecto. El archivo de extensión *.XRF contiene la información que involucra referencias cruzadas en un proyecto, cuando éste contiene varios archivos fuente y entre éstos existen etiquetas o vínculos con un mismo nombre, y los archivos con extensión *.OBJ se relacionan directamente con la simulación.

Una vez seleccionado el archivo a compilar y definido los parámetros que se deseen en las casillas de verificación de la ventana principal de MPASM se pulsa en el botón ASSEMBLE, de presentarse errores al momento de ensamblarse el archivo, se mostrará una caja de diálogo como la que se muestra en la figura 4.7 y no se generará el archivo ejecutable.

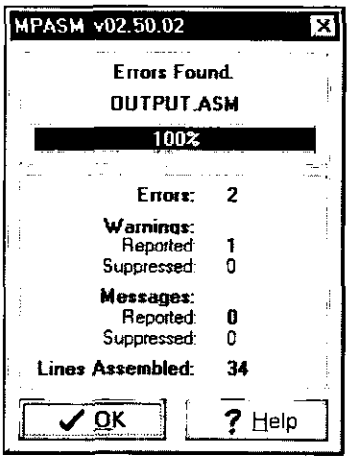


Figura 4.7 Mensaje de error.

El archivo de extensión **.ERR* contendrá la descripción de errores del archivo fuente y de extensión **.ASM*. la figura 4.8 muestra el archivo OUTPUT.ERR el cual puede ser explorado con cualquier editor de texto, como el NOTEPAD de Windows.

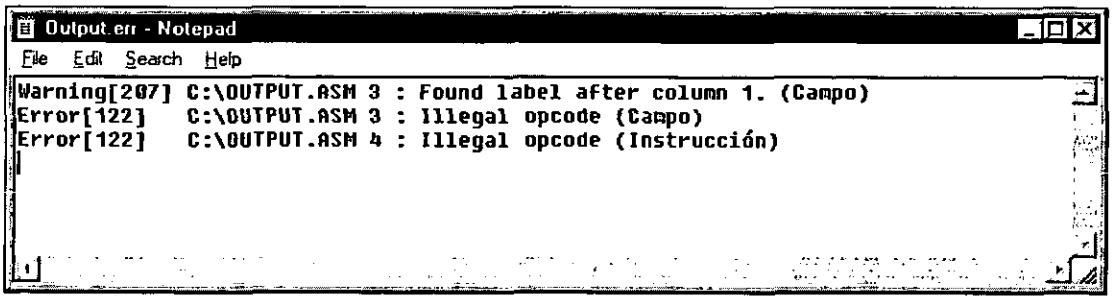


Figura 4.8 Contenido del archivo OUTPUT.ERR.

Una vez corregido el error en el archivo fuente se compila nuevamente, de no existir errores de compilación aparecerá una caja de diálogo como la de la figura 4.9 indicando que se ha ensamblado correctamente el archivo.

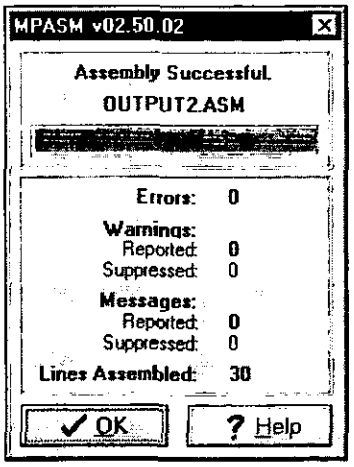


Figura 4.9 Mensaje de ensamble exitoso.

Ahora ya se ha generado un archivo **.HEX* que es el ejecutable que servirá para la grabación del microcontrolador. El contenido de este archivo también puede explorarse con cualquier editor de texto y tendrá un contenido similar al mostrado en la figura 4.10.

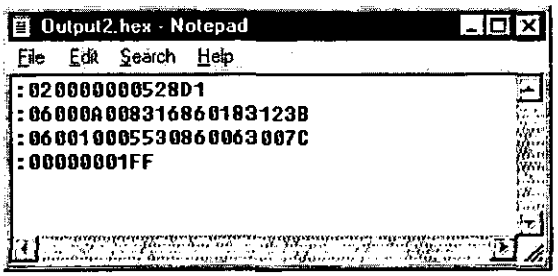


Figura 4.10 Contenido del archivo OUTPUT.HEX.

El fin del proceso de desarrollo de proyectos con microcontroladores está marcado por la programación del dispositivo. Dicho proceso se describe a continuación.

4.3. Grabación del microcontrolador

Otra ventaja que presentan los microcontroladores PIC es una arquitectura abierta al momento de su grabación, ya que no se requiere comprar costosos programadores. Para ello Microchip ha puesto a disposición, a través de su página en Internet, los requerimientos de cada dispositivo para su grabación, lo que ha llevado consigo a que un gran número de diseñadores creen tanto sus grabadores hardware como software.

Este desarrollo de herramientas ha traído consigo la comercialización de varios productos en el ramo, que van desde simuladores hasta programadores universales en una gran gama de precios.

Afortunadamente, no todos los diseñadores tienen objetivos comerciales y han puesto a disposición, vía Internet, sus diseños en hardware y sus programas grabadores gratuitamente.

Haciendo uso de esta opción y buscando en Internet hemos encontrado un diseño no comercial, que se adecua a nuestras necesidades, este grabador hardware tiene la limitante de grabar sólo un tipo de microcontrolador, pero presenta la gran ventaja de no necesitar polarización externa, ya que aprovecha la alimentación del puerto serial aunado a que se puede emplear como ICSP (In Circuit Serial Programming – Programación Serial en Circuito) es decir, que puede programar dispositivos serialmente sin desmontarlos de la tarjeta de prototipos..

Este grabador en su inicio, contaba sólo con un programa grabador que funciona en ambiente MS-DOS®, pero su uso se popularizó y posteriormente un diseñador llamado Tord Andersson desarrolló el programa grabador para el PIC16C84 en ambiente Windows® llamado PicProg que es el que emplearemos y su ventana se muestra en la figura 4.11.

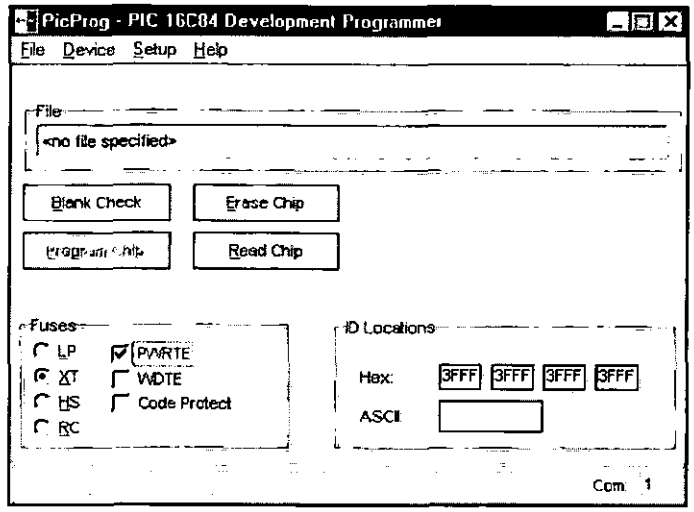


Figura 4.11 Ventana de PicProg.

Este programador si bien fue diseñado para programar el PIC16C84, puede también programar al PIC16F84 teniendo en consideración que, como ya se mencionó, el bit PWRTE de la Palabra de Configuración, tiene invertida su función en ambos. En el primero cuando este bit = 1 el temporizador de *Power-up* está activado, mientras que sucede lo contrario en el segundo, por lo que al momento de la grabación se recomienda tener especial atención en habilitar o deshabilitar dicho temporizador.

El programa cuenta con una barra de menús que permiten la ejecución de operaciones con archivos y configuración del programador. El menú *File* permite efectuar las siguientes operaciones:

Open File: Permite explorar el equipo y sus unidades de almacenamiento para localizar el archivo ejecutable con extensión **.HEX.*

Save File As: Permite guardar en un archivo (dándole el formato hexadecimal) las instrucciones que están en formato binario, contenidas en un microcontrolador, se requiere la previa lectura de la memoria del dispositivo.

Exit: Abandona el programa.

El menú *Device* contiene las siguientes funciones:

Blank Check: Verifica que el microcontrolador no tenga grabado algún programa en su memoria.

Erease Chip: Borra el contenido de la memoria del microcontrolador.

Read Chip: Lee el contenido de la memoria del microcontrolador.

Program Chip: Graba la memoria del microcontrolador. Esta función estará deshabilitada si no se ha seleccionado algún archivo ejecutable.

El menú *Setup* contiene los siguientes sub-menús:

Com Port: Esta opción abrirá la caja de diálogo que se muestra en la figura 4.12, y permite seleccionar el puerto serial (COM1, COM2, etc.) al que está conectado el grabador hardware.

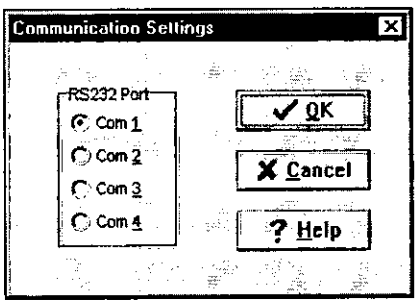


Figura 4.12 Caja de diálogo para la selección de puerto serial.

Check Hardware: Esta opción abrirá una caja de diálogo mostrada en la figura 4.13 y permite cambiar la configuración de terminales del puerto disponible en el equipo, entre un conector de 9 terminales (DB9), que es el tipo por omisión, y uno de 25 terminales (DB25) seleccionando la casilla de verificación correspondiente.

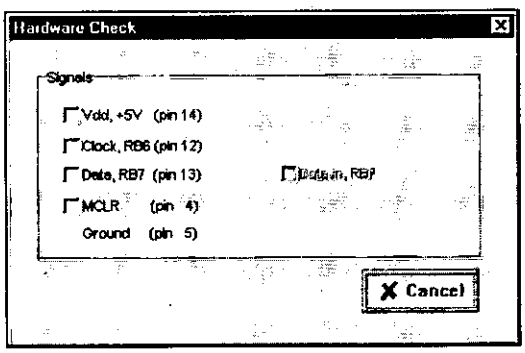


Figura 4.1. Caja de diálogo para la verificación de tipo de conector en la puerto serial.

En el menú *Help* se encuentran los siguientes sub-menús:

Contents: Muestra el contenido de la Ayuda.

Procedures: Muestra el procedimiento a seguir para la grabación y operación del programa.

About: Despliega la caja de diálogo que se muestra en la figura 4.14 en la que se muestra la información de la versión, el autor y el año en que fue creado el programa.

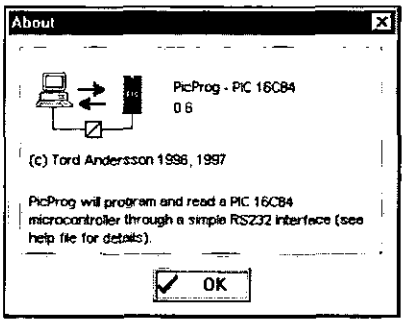


Figura 4.14 Caja de diálogo del sub-menú *About.*

En la ventana de la aplicación se encuentran las siguientes opciones:

File: Indica la ruta donde se ubica el archivo ejecutable con extensión **.HEX.*

Fuses: Permite la selección del tipo de oscilador que se empleará, si se desea habilitar el WDT (Watch Dog Timer), el PWRTE (Power-up Timer Enable) y la Protección del Código, estas opciones definirán la información a grabar en la Palabra de Configuración del microcontrolador.

ID Locations: Permite asignar una clave o etiqueta de cuatro caracteres que el diseñador desee incluir para distinguir su programa.

El programa también cuenta con botones de acceso inmediato a las funciones que se accesan por el menú *Device* y sus correspondientes funciones.

CAPITULO 5

MANUAL DE PRÁCTICAS

INTRODUCCIÓN

La finalidad de las prácticas es la de complementar los conocimientos teóricos para el desarrollo de proyectos con microcontrolador PIC; por lo que nos apoyaremos en el Sistema de Entrenamiento, el cual consta de una tarjeta principal, siete módulos de prueba y un módulo grabador.

La tarjeta principal tiene la función de dar soporte y acceso a las líneas de los puertos del microcontrolador a los diferentes módulos, para esto se tienen cuatro conectores de 11 columnas por 2 líneas distribuidos de manera que se facilite el montaje de los módulos sin que interfieran entre sí, así mismo se reservó una área para el montaje de una tableta de prototipos (protoboard) con el propósito de que se puedan implementar circuitos alternos de desarrollo y conectarlos a los puertos a través de un quinto conector.

En los 5 puertos de la tarjeta principal se tienen disponibles las líneas de los dos puertos del microcontrolador (Puerto A <RA0:RA4> y Puerto B <RB0:RB7>) y las líneas de polarización (+5V, Gnd y +12V) conforme a la distribución mostrada en la figura 5.1.

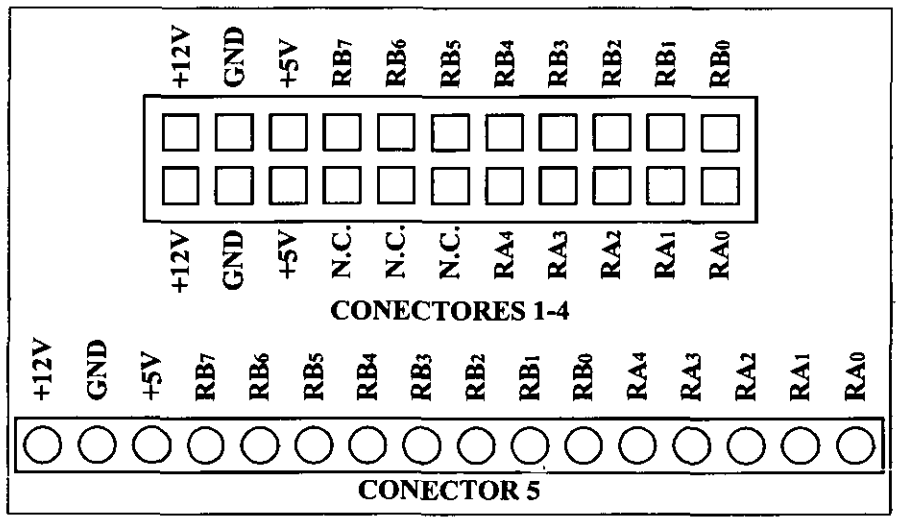


Figura 5.1 Distribución de las líneas disponibles en los conectores de la tarjeta principal.

Otra característica destacable de la tarjeta principal es que tiene un receptáculo tipo ZIF (Zero Insertion Force – Cero Fuerza de Inserción) lo que implica la facilidad de montaje e intercambio del microcontrolador, aunado a que cuenta con la circuitería necesaria para su funcionamiento, como son: polarización, oscilador y reset. La tarjeta principal esta montada en un gabinete que en su interior aloja a la fuente de alimentación, encargada de proporcionar los voltajes adecuados para el funcionamiento del sistema. La figura 5.2 muestra la distribución de los puertos de la tarjeta principal, la circuitería y gabinete.

Módulo con pantalla de siete segmentos, compuesto por cuatro displays de siete segmentos, tiene ocho terminales conectadas en paralelo a cada uno de los segmentos de los displays y cuatro terminales que habilitan a cada uno de los displays, las doce terminales están dispuestas en un conector macho de 11 columnas por 2 líneas, la figura 5.4 muestra el aspecto físico del módulo.

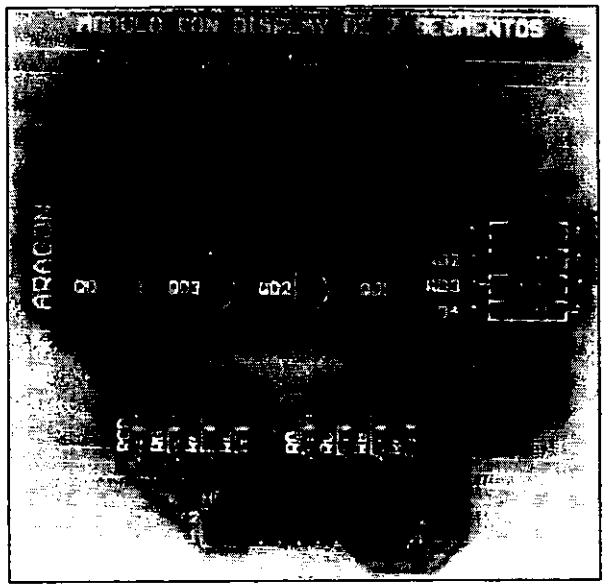


Figura 5.4 Aspecto físico del módulo con Display de siete segmentos.

Módulo con LCD, consiste en un Display LCD de 16 caracteres por 2 líneas, el cual tiene ocho terminales de datos, de éstas, cuatro se pueden deshabilitar a través de un DIP-Switch, para configurar el modo de operación a cuatro u ocho bits. Tiene otras dos terminales destinadas para el control del Display, todas las terminales están dispuestas en un conector macho de 11 columnas por 2 líneas, además cuenta con un control de contraste, el aspecto físico del módulo se muestra en la figura 5.5.

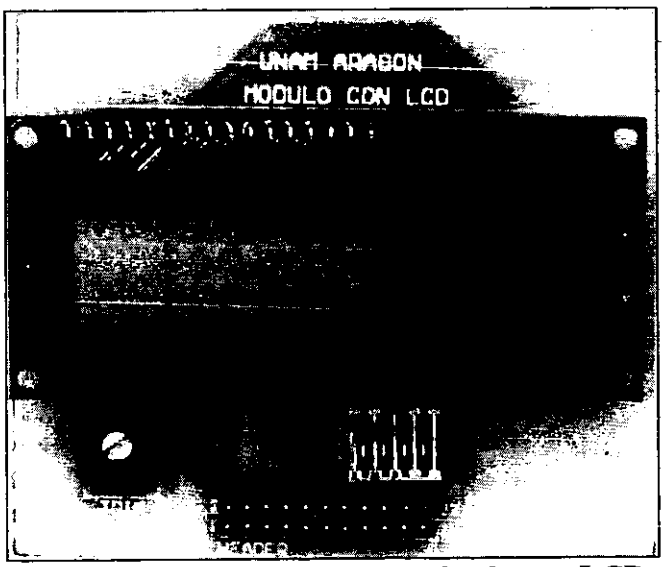


Figura 5.5 Aspecto físico del Módulo con LCD.

Módulo con motor Paso a Paso (PAP). Este módulo cuenta con dos circuitos de interfaz idénticos para el manejo de motores PAP, uno de ellos ya montado sobre la tarjeta. Tiene un conector de 6 terminales para cada motor. Cada circuito de interfaz emplea 4 terminales del sistema que están dispuestos en un conector macho de 11 columnas por 2 líneas, su implementación se muestra en la figura 5.6.

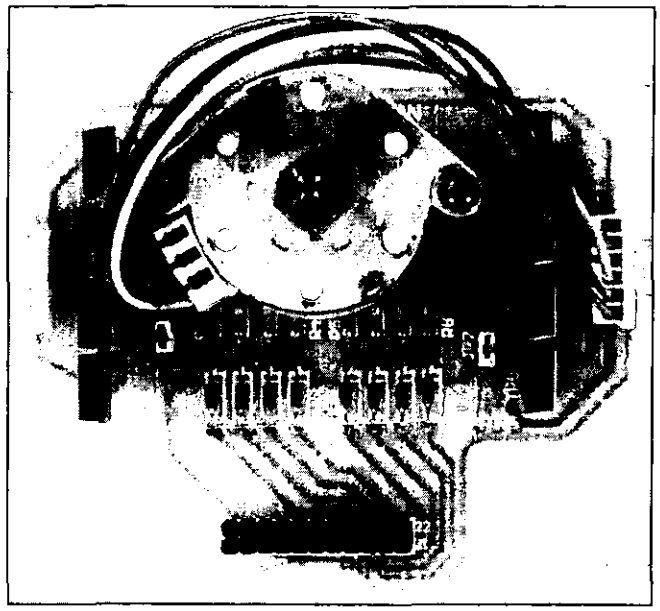


Figura 5.6 Implementación del módulo con motor Paso a Paso.

Módulo con relevadores, este módulo permite el control de cargas de C.D. y C.A. con dos relevadores independientes, los cuales están controlados a través de dos terminales, estas terminales están dispuestas en un conector macho de 11 columnas por 1 línea, el aspecto físico del módulo se muestra en la figura 5.7.

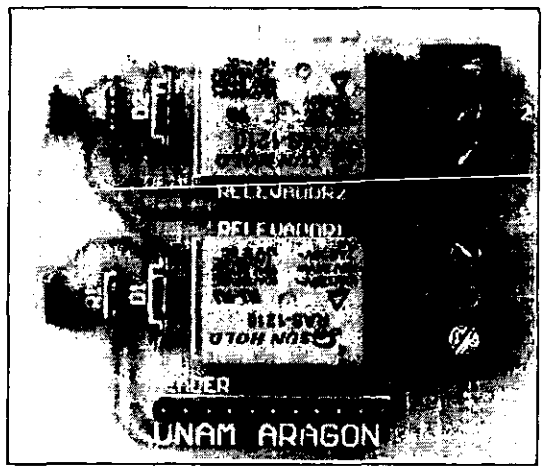


Figura 5.7 Aspecto físico del módulo con relevadores.

Módulo con optoacopladores, este módulo permite el control de cargas de C.D. y C.A. con dispositivos semiconductores, los cuales tienen acoplamiento óptico para aislarlos eléctricamente del sistema, se cuenta con conectores para el control de dos cargas de C.A. y dos cargas de C.D. las terminales de control están dispuestas en un conector macho de 11 columnas por 1 línea, la implementación del módulo se muestra en la figura 5.8.

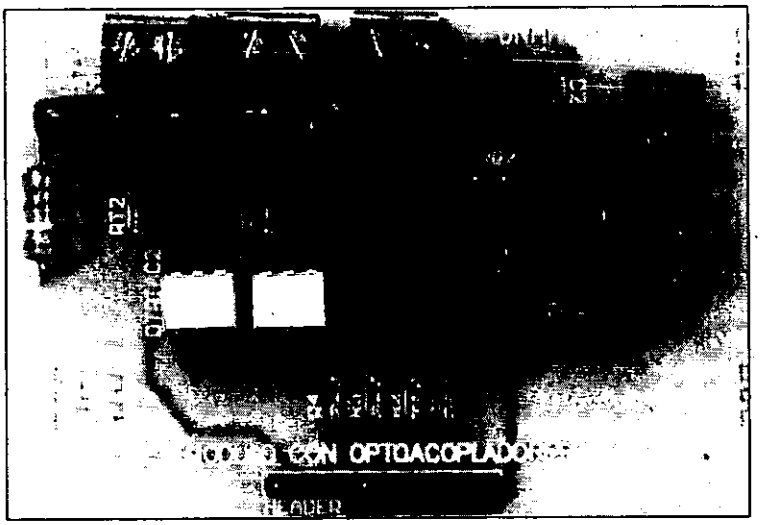


Figura 5.8 Implementación del módulo con optoacopladores.

Módulos digitales de entrada.

Módulo con Push-Button, este módulo proporciona señales digitales de entrada a través de cuatro push-button, sus terminales están dispuestas en un conector macho de 11 columnas por 1 línea, el aspecto físico del módulo se muestra en la figura 5.9.

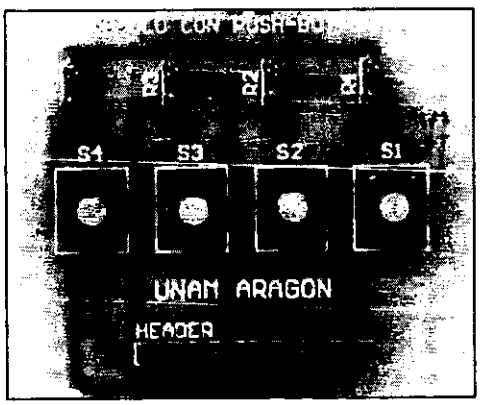


Figura 5.9 Aspecto físico del módulo con Push-Button.

Módulo con DIP-Switch, este módulo proporciona ocho señales digitales de entrada mediante DIP-Switch, las terminales están dispuestas en un conector macho de 11 columnas por 1 línea, la implementación del módulo se muestra en la figura 1.10.

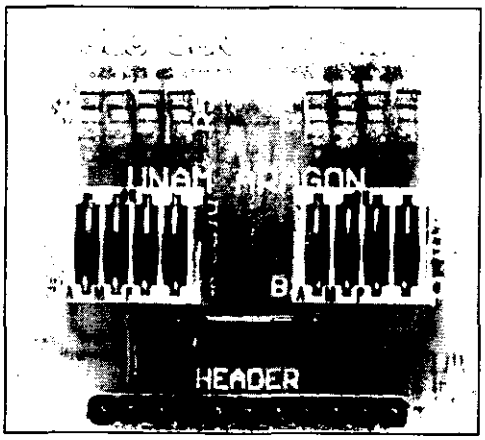


Figura 5.10 Implementación del módulo con DIP-Switch.

Módulo con teclado matricial, consiste en un arreglo matricial de push-button de cuatro columnas por cuatro filas, tiene ocho terminales que proporcionan señales digitales de entrada y están dispuestas en un conector macho de 11 columnas por 1 fila, el aspecto físico del módulo se muestra en la figura 1.11.

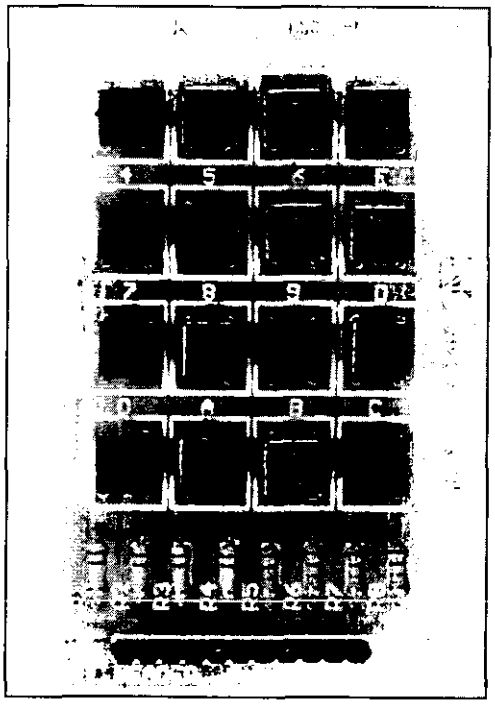


Figura 5.11 Aspecto físico del módulo con teclado matricial.

Módulos bidireccionales

Módulo RS-232, está destinado para permitir la comunicación serial del sistema a través de un protocolo RS-232, esto se consigue con un circuito integrado auxiliar (MAX-232) y un conector DB-9. Tiene dos terminales de comunicación (Tx y Rx), estas terminales están dispuestas en un conector macho de 11 columnas por 1 línea, la figura 5.12 muestra la implementación del módulo.

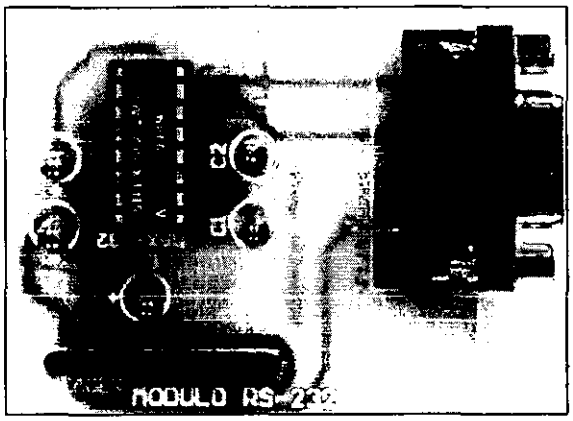


Figura 5.12 Implementación del módulo RS-232.

Módulo grabador es el módulo que permite la grabación del microcontrolador y puede proporcionar información del estado del dispositivo. La grabación y lectura del circuito integrado se consigue mediante una PC con un puerto serial disponible y el programa Pic-Prog, la figura 5.13 muestra el aspecto físico de este módulo.

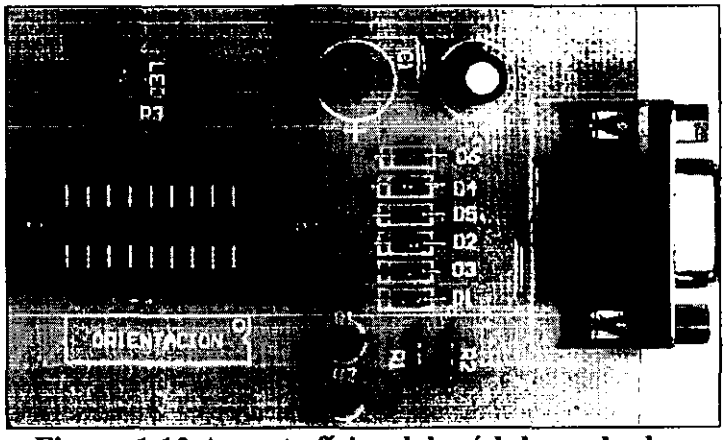


Figura 1.13 Aspecto físico del módulo grabador.

Nota: las características de todos los módulos están detalladas en el Capítulo 3

El sistema de entrenamiento está formado por el conjunto de herramientas tanto del tipo hardware como software, que se necesitan para desarrollar un proyecto con microcontroladores, a nivel software las principales herramientas del sistema son las siguientes:

- Programa editor.
- Programa ensamblador.
- Programa simulador.
- Programa grabador.

Cada uno de ellos comprende una fase del desarrollo de un proyecto.

En la figura 5.14 se muestra el diagrama de flujo con las fases que compone el desarrollo de un proyecto.

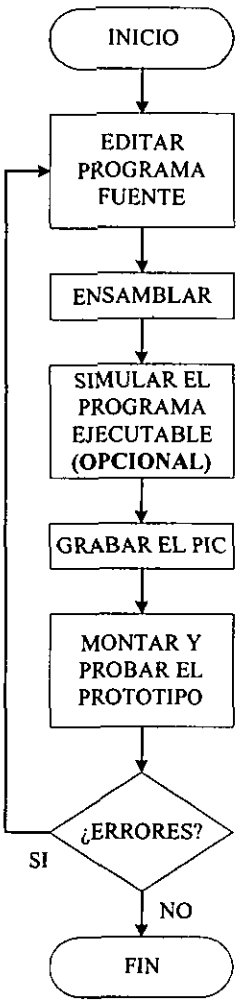


Figura 5.14. Diagrama de flujo del desarrollo de un proyecto.

A continuación se hace una breve reseña de las herramientas a nivel software de las fases que comprende el desarrollo de proyectos con microcontroladores.

Edición

Para apoyar al diseñador en la creación de proyectos con microcontroladores el fabricante ha desarrollado un programa llamado MPLAB y le ha anexado las siglas IDE (Integrated Development Enviroment - Ambiente Integrado de Desarrollo), que es un software integral y lo compone:

- Un administrador de proyectos.
- Un editor.
- Un ensamblador.
- Un simulador.
- Un grabador.

Los requerimientos mínimos para emplear MPLAB son los siguientes:

- PC compatible 486 o superior, (Pentium recomendado).
- Microsoft Windows 3.1x® o Windows 95/98®.
- Pantalla VGA (SVGA recomendada).
- 8 MB de memoria RAM (32 recomendada).
- 20 MB de espacio libre en Disco Duro.
- Ratón u otro dispositivo puntero.

Si se requiere información detallada del empleo de MPLAB se recomienda dirigirse a la página en Internet de Microchip en la dirección www.microchip.com en donde se encuentra el manual completo.

La figura 5.15 muestra la ventana principal de MPLAB y la ventana del editor.

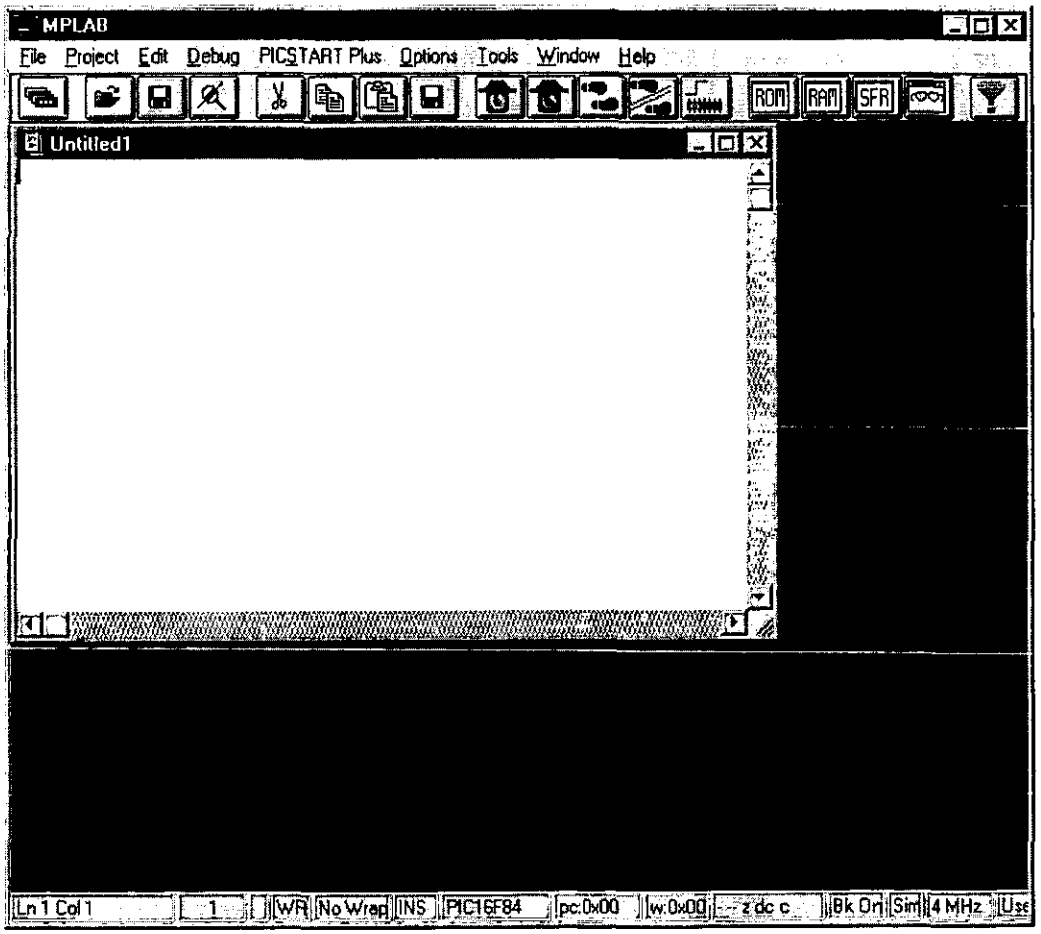


Figura 5.15. Ventana principal de MPLAB y del Editor para la escritura del código fuente.

Ensamblado

Las instrucciones del código fuente del ensamblador son comprensibles para el diseñador, pero el microcontrolador sólo entiende los números binarios. La función de un programa ensamblador es convertir el texto del código fuente en el equivalente de lenguaje máquina numérico del microcontrolador.

El siguiente paso en el proceso de desarrollo de un proyecto es ensamblar el archivo que contiene el código fuente y producir un archivo de código ejecutable. Una vez creado el ejecutable, ya se puede proceder a grabar el microcontrolador.

El programa ensamblador MPASM convierte el código fuente en código ejecutable en dos pasos. En el primero comprueba la correcta sintaxis de las instrucciones, los nombres de las etiquetas duplicados y asigna valores a los símbolos. En el segundo paso el ensamblador convierte todas las instrucciones en sus respectivos códigos máquina.

MPASM es parte de MPLAB, y se instala automáticamente cuando se instala MPLAB, la ventana principal es la que se muestra en la figura 5.16.

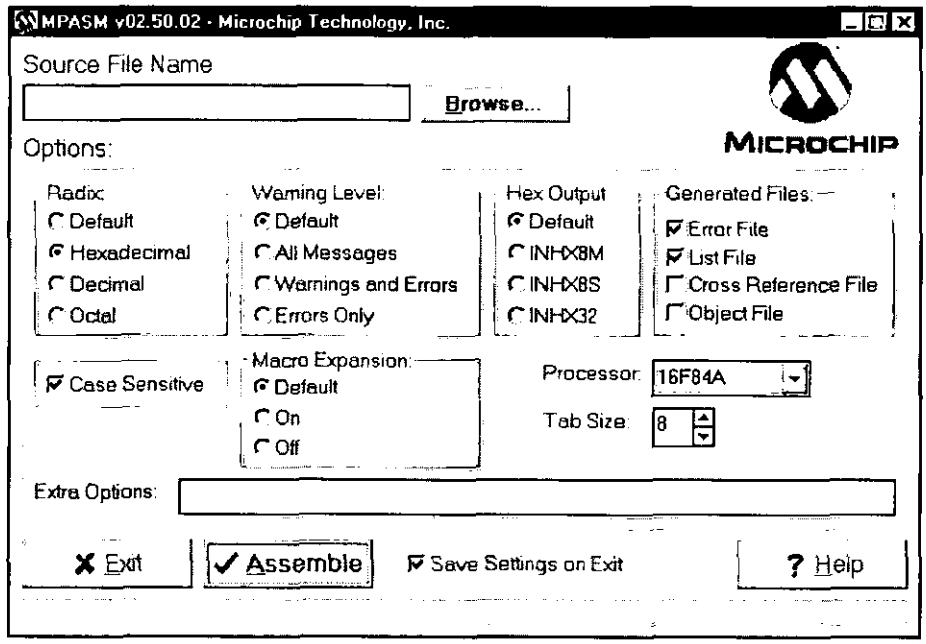


Figura 5.16. Ventana principal del ensamblador MPASM.

Simulación (opcional)

La simulación es en realidad un proceso complejo y requiere la definición de una gran cantidad de parámetros, por lo que en base a nuestra experiencia se sugiere como opcional.

Grabación del microcontrolador

MPLAB tiene integrado un programa grabador, que es compatible con los grabadores hardware que manufactura Microchip, como son el Pic Start 16C® , el Pic Start 16B®, el Pic Start Plus® y Pro Mate II® y con algunos grabadores ajenos a la empresa Microchip.

Otra ventaja que presentan los microcontroladores PIC es una arquitectura abierta al momento de su grabación, ya que no se requiere comprar costosos programadores. Para ello Microchip ha puesto a disposición, a través de su página en Internet, los requerimientos de cada dispositivo para su grabación, lo que ha llevado consigo a que un gran número de diseñadores creen tanto sus grabadores hardware como software.

Nosotros emplearemos el programa grabador encontrado en Internet llamado PicProg, el cual trabaja perfectamente con nuestro programador hardware, ya que fue diseñado para grabar el PIC16C84. Trabaja en ambiente Windows®. Su ventana se muestra en la figura 5.17.

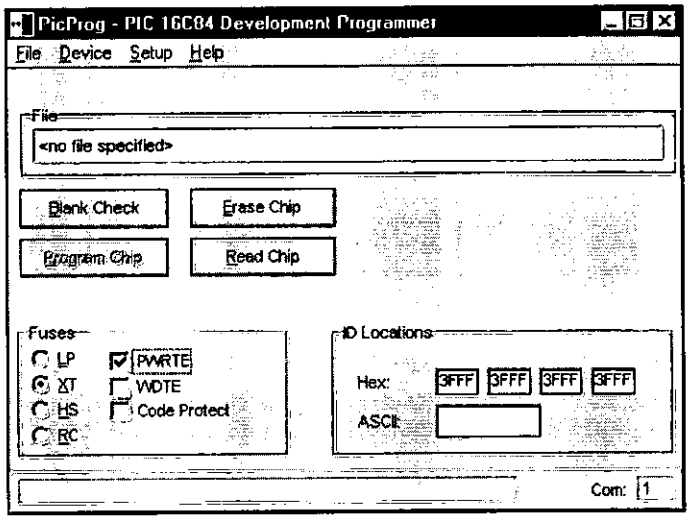


Figura 5.17. Ventana de PicProg.

Se dan detalles sobre el empleo de las herramientas a nivel software del sistema de entrenamiento en el Capítulo 4.

Práctica N° 1

MANEJO DE PUERTOS.

OBJETIVOS: Que el alumno se familiarice con el manejo de las instrucciones del lenguaje ensamblador propio de los microcontroladores PIC, para la configuración y uso de los puertos del PIC16F84, así como en las operaciones aritméticas sencillas.

DESARROLLO.

Actividad 1

Para cumplir con el objetivo de familiarizar al alumno en el manejo de las instrucciones del lenguaje ensamblador, y la configuración y uso de los puertos del microcontrolador; se propone el listado del programa OUTPUT.ASM. Se empleará el sistema de entrenamiento (figura 5.2) y el módulo de monitores LED (figura 5.3).

a) Escriba el listado de programa OUTPUT.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre OUTPUT y la extensión .ASM y guárdelo.

```
;*******************************************************************************
;Programa OUTPUT.ASM: Muestra en el PUERTO B a través del módulo de monitores LED el número
;binario 01010101
            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=16F84         ;Selecciona el tipo de microcontrolador a usar
        LIST    C=132           ;Archivo con listado de 132 caracteres
        RADIX   HEX             ;Especifica la numeración base

PORTB   EQU     06              ;Asigna al registro PORTB la dirección 06 de la
                                ;pagina 0 de la memoria de datos
TRISB   EQU     06              ;Asigna al registro TRISB la dirección 06 de la
                                ;pagina 1 de la memoria de datos
ESTADO  EQU     03              ;Asigna al registro ESTADO la dirección 03 de las
                                ;dos paginas de la memoria de datos

        ORG     00              ;Indica al ensamblador la dirección de la primera
        goto    inicio          ;instrucción del programa (00h)

inicio  bsf     ESTADO,5        ;Pone en 1 el quinto bit del registro ESTADO (PA0)
                                ;para seleccionar la pagina 1 de la memoria de datos
        clrf    TRISB           ;Programa al puerto b como salidas poniendo a cero el
                                ;registro TRISB
        bcf     ESTADO,5                ;Pone en 0 el quinto bit del registro ESTADO (PA0)
                                ;para seleccionar la pagina 0 de la memoria de datos

patron  movlw   B'01010101'     ;Coloca el patrón binario 01010101 en el registro W
        movwf   PORTB           ;Mueve el contenido del registro W al registro PORTB
                                ;para mostrar el patrón binario en los LED conectados
                                ;al PUERTO B

        goto    patron          ;Brinca a la etiqueta patrón y entra a un ciclo
                                ;infinito
        END                     ;Final del código fuente

*******************************************************************************
```

b) Inserte el módulo de monitores LED en la tarjeta principal del sistema de entrenamiento teniendo cuidado de insertarlo en el conector 1, y en la línea correspondiente al Puerto B. Recuerde la figura 5.1.

c) Ejecute la aplicación MPASM, configure las casillas de verificación como se muestra en la figura 1.16 y en la caja **Processor** elija 16F84A o bien el número de referencia de su microcontrolador. Pulse en el botón **Browse** para especificar la ruta donde se encuentra el archivo OUTPUT.ASM y finalmente pulse en el botón **Assemble**. Esta acción generará en la misma ruta donde se encuentra el archivo OUTPUT.ASM, un archivo OUTPUT.ERR en el cual, en caso de existir errores se indicará la línea y el error en el archivo OUTPUT.ASM. De no existir errores se generará también el archivo OUTPUT.HEX, el cual será el archivo destinado para la grabación del PIC.

d) Coloque el microcontrolador en el zócalo del Módulo Grabador, teniendo cuidado en la orientación del PIC que la terminal 1 del zócalo esté ubicada del mismo lado de la palanca de sujeción y baje la palanca para asegurarlo en su posición.

e) Ejecute la aplicación PicProg. En la ventana de la aplicación pulse en el menú **File**, seleccione **Open File**, especifique la ruta en donde se encuentra ubicado el archivo OUTPUT.HEX y pulse en **Aceptar**. Marque las casillas de verificación como se indica en el listado del programa y finalmente, pulse en el botón **Program Chip**.

f) Retire el PIC del Módulo Grabador e insértelo en el zócalo correspondiente de la tarjeta principal. Encienda el sistema de entrenamiento y anote sus observaciones.

g) Con base en el programa anterior diseñe un programa de manera que ahora las salidas se obtengan por el Puerto A y que muestre la activación de las terminales RA0, RA3 y RA4. Empleando para su evaluación el modulo de monitores LED

Actividad 2

Para complementar el objetivo y familiarizar al alumno en operaciones aritméticas simples, se propone el listado del programa IN-OUT.ASM. Se empleará el sistema de entrenamiento, el módulo con Push-Button (figura 5.9) y el módulo de monitores LED.

a) Capture el listado del programa IN-OUT.ASM que se muestra a continuación, en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre IN-OUT y la extensión .ASM y guárdelo.

```
;*******************************************************************************
;Programa IN-OUT.ASM: Controla los cuatro push-button del módulo conectados en el puerto A en las
;líneas RA0-RA3 y ;enciende el bit ;correspondiente del Módulo de monitores LED conectado en el
;puerto B en las líneas RB0-RB3 dependiendo ;de las entradas introducidas por el ;puerto A. Con
;un 0 en el puerto A se enciende el LED correspondiente del puerto B.

        ;configuración para las casillas de verificación en el programa grabador:
        ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

    LIST    P=16F84         ;Se usa el PIC16F84
    RADIX   HEX             ;Se emplea el sistema de numeración hexadecimal

    W       EQU     00      ;Cuando el destino es W, d=0
    F       EQU     01      ;Cuando el destino es el registro f, d=1
    PUERTAA EQU     05      ;El Puerto A ocupa la dirección 5 de la memoria de datos
    PUERTAB EQU     06      ;El Puerto B ocupa la dirección 6 de la memoria de datos
```

```
        ESTADO  EQU     03      ;El registro de Estado ocupa la dirección 3 de los bancos 0 y 1 de
                                ;la memoria de datos

        ORG     00              ;El programa comienza en la dirección 0 (Vector Reset)
        goto    inicio          ;Se salta a la etiqueta "inicio"
        ORG     05              ;Se asigna la dirección 05 a la siguiente instrucción

inicio  bsf     ESTADO,5        ;Pone a 1 el bit 5 del registro ESTADO. Acceso al banco 1.
        clrf    PUERTAB         ;Se configuran como salidas las líneas de la Puerta B
        movlw   0x0F            ;El registro W se carga con la literal binaria "00001111"
        movwf   PUERTAA         ;Se configuran como entradas los bits 0, 1, 2 y 3 del Puerto A
        bcf     ESTADO,5        ;Pone a 0 el bit 5 del registro ESTADO. Acceso al banco 0
bucle   clrf    PUERTAA         ;Pone a ceros el Puerto A
        movf    PUERTAA,W       ;Carga el registro de datos del Puerto A en W
        comf    PUERTAA,W       ;Complementa a 1 la entrada y la guarda en W
        movwf   PUERTAB         ;El contenido de W se deposita en el registro de datos
                                ;del Puerto B
        goto    bucle           ;Se crea un bucle cerrado e infinito

        END                     ;Fin del programa
```

****

b) Inserte el módulo de monitores LED en la tarjeta principal del sistema de entrenamiento teniendo cuidado de insertarlo en conector 1, en la línea correspondiente al Puerto B y el módulo con Push-button en el conector 2, en la línea correspondiente al Puerto A.

c) Repita del paso c) al paso e) de la Actividad 1, pero ahora para el archivo IN-OUT.ASM

d) Retire el PIC del Módulo Grabador e insértelo en el zócalo correspondiente de la tarjeta principal. Encienda el sistema de entrenamiento, pulse cualquier Push-button y anote sus observaciones.

e) Utilizando la instrucción *addfw* modifique el programa IN-OUT.ASM de tal manera que la información proporcionada por el Puerto de entrada se sume al número 02h y el resultado se muestre en el Módulo de Monitores LED conectado al Puerto de salida.

f) Para evaluar el punto anterior, se propone cambiar el módulo de Push-button por el módulo con DIP-Switch (figura 5.10).

Actividad 3: CUESTIONARIO.

a) Desarrolle el diagrama de flujo de los programas OUTPUT.ASM e IN-OUT.ASM.

b) ¿Cuáles son las instrucciones necesarias para configurar un puerto como entrada?

c) Desarrolle el diagrama de flujo de un programa que multiplique por 3 un número de cuatro bits introducido a través del Puerto A y el resultado de la operación se visualice en el Puerto B.

d) ¿Cuál es el procedimiento y qué instrucciones se necesitan para crear un ciclo infinito?

e) Investigue por lo menos tres métodos para obtener retardos.

f) Investigue la función que realizan las instrucciones *call* y *retlw*.

Práctica Nº 2

RETARDOS.

OBJETIVOS: Que el alumno se familiarice en el manejo del temporizador TMR0 y WDT del microcontrolador para obtener retardos e introducirlo en el manejo de la estructura de ciclos anidados para obtener retardos.

DESARROLLO

Actividad 1

Para cumplir con el objetivo de familiarizar al alumno en el manejo del temporizador TMR0, se propone el listado del programa PARPADEO.ASM, Se empleará el sistema de entrenamiento y el módulo de monitores LED.

a) Capture el listado del programa PARPADEO.ASM que se muestra a continuación, en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre PARPADEO y la extensión .ASM y guárdelo.

```
;*******************************************************************************
; PARPADEO.ASM: Programa que ilustra cómo realizar una temporización sin emplear interrupciones.

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=16F84         ;Define el procesador a usar
        RADIX   HEX             ;Define la numeración base

PUERTOB         EQU     0x06    ;El Puerto B se encuentra en la dirección 06h de la memoria de datos
ESTADO          EQU     0x03    ;El registro ESTADO ocupa la dirección 03h de la memoria de datos
TMR0_OPT        EQU     0x01    ;El registro TMR0 esta en la dirección 01 del banco 0 y OPCIÓN en la
                                ;dirección 01h del banco 1 de la memoria de datos
INTCON          EQU     0x0B    ;El registro INTCON está en la dirección 0Bh de ambos bancos de la
                                ;memoria de datos

        ORG     0               ;Inicio del programa en dirección 0

        bsf     ESTADO,5        ;Selecciona al Banco 1 de la memoria de datos.
        movlw   b'11010101'     ;Se configura el registro TMR0_OPT como se indica a continuación:

;*************************************************************************************
                ;OPCION,7=1 - Resistencias de polarización habilitadas
                ;OPCION,6=1 - Interrupción externa con flanco descendente
                ;OPCION,5=0 - Fuente de reloj externa para TMR0
                ;OPCION,4=1 - Flanco de la señal externa para incremento del TMR0 descendente
                ;OPCION,3=0 - Divisor de frecuencia asignado al TMR0
                ;OPCION,2:0=101 - La frecuencia de entrada se divide entre 64
;*************************************************************************************
        movwf   TMR0_OPT        ;Mueve el contenido del registro W al registro TMR0_OPT
        movlw   0x00            ;Carga ceros en el registro W
        movwf   PUERTOB         ;Configura el Puerto B como salida
        bcf     ESTADO,5        ;Selecciona al Banco 0 de la memoria de datos
        clrf    PUERTAB         ;Las líneas de salida del Puerto B = 0

parpa   bsf     PUERTOB,7       ;Enciende el led conectado a la línea RB7 = 1
        call    retardo         ;Llamada a subrutina de RETARDO
        bcf     PUERTOB,7       ;Apaga el led, RB7 = 0
        call    retardo         ;Llamada a subrutina de RETARDO
        goto    parpa           ;entra en un ciclo infinito

retardo clrf    TMR0_OPT        ;TMR0 = 0  y empieza su incremento
explora btfss   INTCON,2        ;Explora el bit 2 del registro INTCON
```

```
        goto    explora         ;No se ha desbordado el TMR0
        bcf     INTCON,2        ;Pone a cero el bit 2 del registro INTCON
        return                  ;El TMR0 se ha desbordado, y retorna al programa principal

        END                     ;Fin de programa
```

****

b) Ensamble el archivo PARPADEO.ASM y grabe el microcontrolador con el archivo PARPADEO.HEX. Conforme el procedimiento seguido en la actividad 1 de la práctica 1.

c) Inserte el módulo de monitores LED en la tarjeta principal del sistema de entrenamiento teniendo cuidado de insertarlo en el conector 1, en la línea correspondiente al Puerto B.

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal. Encienda el sistema de entrenamiento y anote sus observaciones.

e) Modifique en el programa PARPADEO.ASM la configuración del registro TMR0_OPT con diferentes rangos de división de frecuencia, repita de los pasos b) al d) y anote sus observaciones.

Actividad 2

Para complementar el objetivo en el manejo de la estructura de ciclos de retardo, se propone el listado del programa RETARDO.ASM. Se emplearán el sistema de entrenamiento y el módulo de monitores LED.

a) Capture el listado del programa RETARDO.ASM que se muestra a continuación, en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre RETARDO y la extensión .ASM y guárdelo.

```
**********
;RETARDO.ASM: Programa que ilustra como realizar una temporización sin emplear interrupciones. Se
;realiza una temporización que se emplea para hacer parpadear un diodo led en la línea RB7

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST            P=16C84         ;Define el procesador a usar
        RADIX           HEX             ;Define la numeración base

        PUERTAB         EQU   0x06
        TMR0_OPT        EQU   0x01      ;TMR0 en banco 0 y OPCIÓN en banco 1
        ESTADO          EQU   0x03
        CONTA           EQU   0x0D      ;Se declara el registro CONTA

        ORG    0                        ;Inicio del programa en dirección 0

        bsf    ESTADO,5                 ;Selecciona el Banco 1 de la memoria de datos
        movlw  b'11010110'              ;Se configura el registro TMR0_OPT como se indica a
                                        ;continuación:
;*****************************************************************************************
            ;OPCION,7=1 - Resistencias de polarización habilitadas
            ;OPCION,6=1 - Interrupción externa con flanco descendente
            ;OPCION,5=0 - Fuente de reloj externa para TMR0
            ;OPCION,4=1 - Flanco de la señal externa para incremento del TMR0 descendente
            ;OPCION,3=0 - Divisor de frecuencia asignado al TMR0
            ;OPCION,2:0=110 - La frecuencia de entrada se divide entre 128
;*****************************************************************************************
        movwf  TMR0_OPT         ;Mueve el contenido del registro W al registro TMR0_OPT
        movlw  0x00             ;Pone en ceros el registro W
        movwf  PUERTAB          ;Configura al Puerto B como salida
        bcf    ESTADO,5         ;Selecciona el Banco 0 de la memoria de datos
        clrf   PUERTAB          ;Pone las líneas del Puerto B a 0

parpa   bsf    PUERTAB,0        ;Enciende el led conectado en la línea RB0 = 1
```

```
        call   retardo        ;Llamada a subrutina RETARDO
        bcf    PUERTAB,0      ;Apaga el led conectado en la línea RB0 = 0
        call   retardo        ;Llamada a subrutina RETARDO
        goto   parpa          ;Salta a la etiqueta PARPA

retardo movlw  d'122'         ;Carga al registro W con la literal decimal 122
        movwf  CONTA          ;Mueve el contenido del registro W al registro CONTA
ciclo1  call   explora        ;Llama a la subrutina EXPLORA
        clrf   TMR0_OPT       ;Pone a ceros el TMR0 y empieza su incremento
        decf   CONTA,0        ;Decrementa el registro CONTA y lo guarda en el registro W
        movwf  CONTA          ;Mueve el contenido del registro W al registro CONTA
        btfss  ESTADO,2       ;Verifica el bit 2 (Z) del registro ESTADO, si es 1 brinca a RETURN
        goto   ciclo1         ;Vuelve a CICLO1
        return                ;Si el ciclo se ha ejecutado 122 veces regresa al programa principal

explora btfss  TMR0_OPT,6     ;Verifica el valor del bit 6 del TMR0, si es 1 brinca a RETURN
        goto   explora        ;El bit 6 del TMR0 no es igual a 1
        return                ;Ha llegado TMR0 al valor 128d  y regresa al programa principal
        END                   ;Fin de programa

*******************************************************************************
```

b) Ensamble el archivo RETARDO.ASM y grabe el microcontrolador con el archivo RETARDO.HEX. Conforme el procedimiento seguido en la actividad 1 de la práctica 1.

c) Inserte el módulo de monitores LED en la tarjeta principal del sistema de entrenamiento teniendo cuidado de insertarlo en conector 1, en la línea correspondiente al Puerto B.

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal. Encienda el sistema de entrenamiento y anote sus observaciones.

e) Modifique en el programa RETARDO.ASM con diferentes valores decimales en la subrutina *retardo* repita de los pasos b) al d) y anote sus observaciones.

Actividad 3

Para cumplir con el objetivo de introducir al alumno en el manejo del temporizador WDT se propone el programa SECUENCI.ASM. Se empleará el sistema de entrenamiento y el módulo de monitores LED

a) Capture el listado del programa SECUENCI.ASM que se muestra a continuación, en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre propuesto, la extensión .ASM y guárdelo.

```
*******************************************************************************
;SECUENCI.ASM: Este programa ilumina una serie de led's en forma secuencial principio-fin-
;principio
          ;configuración para las casillas de verificación en el programa grabador:
          ;********  WDTE = ON , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

      LIST   P=16F84         ;Indica el tipo de PIC

PORTB EQU    0x06    ;Dirección que ocupa el registro Puerto B en la memoria de datos
TRISB EQU    0x86    ;Dirección que ocupa el registro TRISB en la memoria de datos
OPCION EQU   0x81    ;El registro OPCION está en la dirección 81h de la memoria de datos
ESTADO EQU   0x03    ;El registro ESTADO está en la dirección 81h de la memoria de datos
CARRY EQU    0x00    ;CARRY es el bit 0 del registro ESTADO
RP0   EQU    0x05    ;RP0 es el bit 5 del registro ESTADO
MSB   EQU    0x07    ;La posición del bit más significativo

      org    0h

      bsf    ESTADO,RP0      ;Selecciona al Banco de registros 1
      clrf   TRISB           ;Puerto B configurado para salida
      movlw  b'00001010'     ;Se configura el registro OPCION como se indica a continuación
```

```
;*****************************************************************************************
;       OPCION,7=0 - Resistencias de polarización deshabilitadas
;       OPCION,6=0 - Interrupción externa con flanco ascendente
;       OPCION,5=0 - Fuente de reloj externa por la terminal RA4/T0CKI
;       OPCION,4=0 - Flanco de la señal externa para incremento del TMR0 ascendente
;       OPCION,3=1 - Divisor de frecuencia asignado al TMR0
;       OPCION,2:0 }
;       OPCION,1:1 } La frecuencia de entrada se divide entre 4
;       OPCION,0:0 }
;*****************************************************************************************
        movwf   OPCION          ;PRESCALER (1:4) asignado al WDT
        bcf     ESTADO,RP0      ;Selecciona al banco de registros 0
        clrf    PORTB           ;Led's apagados
        incf    PORTB,F         ;Se enciende LED de la derecha
        bcf     ESTADO,CARRY    ;Pone en cero el bit CARRY = 0

left    sleep                   ;Modo de bajo consumo
        rlf     PORTB,F         ;Enciende siguiente led a la izquierda
        btfss   PORTB,MSB       ;Revisa si el bit MSB está en 1, si sí, brinca una instrucción
        goto    left            ;Si no, repite el programa desde LEFT

right   sleep                   ;Modo de bajo consumo
        rrf     PORTB,F         ;Encender siguiente led a la derecha
        btfss   PORTB,0         ;Alcanzado final por la derecha?
        goto    right           ;No: repetir programa desde la etiqueta right
        goto    left            ;Si: Comienza un nuevo ciclo
        end                     ;Directiva de fin de programa

*****************************************************************************************
```

b) Ensamble el archivo SECUENCI.ASM y grabe el microcontrolador con el archivo SECUENCI.HEX. Conforme el procedimiento seguido en la actividad 1 de la práctica 1.

c) Inserte el módulo de monitores LED en la tarjeta principal del sistema de entrenamiento teniendo cuidado de insertarlo en conector 1, en la línea correspondiente al Puerto B.

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal. Encienda el sistema de entrenamiento y anote sus observaciones.

Actividad 4: CUESTIONARIO.

a) Dibuje el diagrama a bloques del programa PARPADEO.ASM.

b) Escriba una rutina para obtener un retardo de un segundo.

c) Investigue el procedimiento para evitar que el temporizador WDT se desborde.

d) Explique brevemente las funciones del temporizador TMR0.

e) ¿Qué función tiene la instrucción *sleep*?

Práctica Nº 3

INTERRUPCIONES

OBJETIVOS: El alumno comprobará en forma experimental mediante los programas propuestos, el manejo de interrupciones, específicamente por desbordamiento del TMR0 y por interrupción externa (RB0/INT)

DESARROLLO.

Actividad 1. Interrupción por desbordamiento del TMR0.

Para complementar el objetivo de familiarizar al alumno en el manejo del temporizador TMR0, se propone el listado del programa TMR0EXT. ASM. Se empleará el sistema de entrenamiento, el módulo de monitores LED y un generador de pulsos de frecuencia variable como el mostrado en la figura 5.18. Este circuito puede ser implementado en la tableta de prototipos del sistema de entrenamiento.

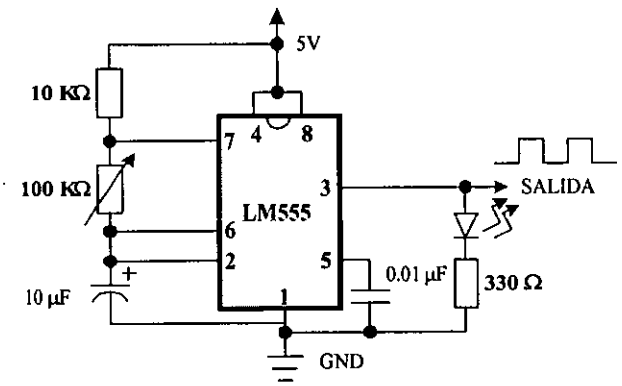


Figura 5.18 Generador de pulsos de frecuencia variable empleando un C.I. LM555.

Nota: Con los valores propuestos de los componentes este circuito puede generar pulsos en un rango de frecuencias de 0.5Hz a 14Hz.

a) Capture el listado del programa TMR0EXT.ASM que se muestra a continuación, en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre TMR0EXT y la extensión .ASM y guárdelo.

```
;*******************************************************************************
;Programa TMR0EXT.ASM permite el conteo de eventos externos a través de TMR0, para lo cual nos
;apoyaremos de un generador de pulsos como fuente señal digital. Los pulsos serán introducidos a
;través de la terminal RA4/T0CKI

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=16F84         ;Define el microcontrolador a usar
        RADIX   HEX             ;Especifica la numeración base

PORTA   EQU     05      ;El registro PORTA en la dirección 05 del banco 1 de la memoria de datos
PORTB   EQU     06      ;El registro PORTB en la dirección 06 del banco 1 de la memoria de datos
```

```
ESTADO  EQU     03      ;El registro ESTADO en la dirección 03 de la memoria de datos
INTCON  EQU     0B      ;El registro INTCON en la dirección 0B de la memoria de datos
TMR0_OP EQU     01      ;El registro TMR0 y OPTION se combinan para facilitar su manejo en la
                        ;dirección 01 de ambos bancos de la memoria de datos
CONTA   EQU     0C      ;Registro auxiliar en la dirección 0C de la memoria de datos

        ORG     0               ;El programa comienza en la dirección 00h (Vector RESET)
        GOTO    inicio          ;Brinca a la etiqueta inicio
        ORG     04              ;Direcciona al vector de interrupción
        GOTO    rsi             ;Instrucción a ejecutarse tras una interrupción


inicio  BSF     ESTADO,5        ;Selecciona el Banco 1 de la memoria de datos
        MOVLW   b'00011111'     ;Se configuran las terminales del Puerto A
        MOVWF   PORTA           ;como entradas
        MOVLW   b'00100000'     ;Se configura el registro OPCION como se indica a continuación:

;*****************************************************************************************
;       OPCION,7=0 - Resistencias de polarización deshabilitadas
;       OPCION,6=0 - Interrupción externa con flanco ascendente
;       OPCION,5=1 - Fuente de reloj externa por la terminal RA4/T0CKI
;       OPCION,4=0 - Flanco de la señal externa para incremento del TMR0 ascendente
;       OPCION,3=0 - Divisor de frecuencia asignado al TMR0
;       OPCION,2:0=0 - La frecuencia de entrada se divide entre 2
;*****************************************************************************************
        MOVWF   TMR0_OP
        CLRF    PORTB           ;Se configuran las terminales del Puerto B como salidas
        BCF     ESTADO,5        ;Se selecciona el Banco 0 de la memoria de datos
        CLRF    PORTB           ;Pone a ceros las terminales del Puerto B
        CLRF    TMR0_OP         ;Pone en cero al registro TMR0
        MOVLW   b'10100000'     ;Configura al registro INTCON como se indica a continuación:

;*****************************************************************************************
;       INTCON,7=1 -   Habilitado el permiso global de interrupciones
;       INTCON,6=0 -   Interrupción por escritura de la memoria EEPROM deshabilitada
;       INTCON,5=1 -   Interrupción por desbordamiento del TMR0 habilitado
;       INTCON,4=0 -   Interrupción externa por la terminal RB0 deshabilitada
;       INTCON,3=0 -   Interrupciones por cambio de estado en las terminales RB7:RB4 deshabilitada
;       INTCON,2:0=0-  Configura las condiciones iniciales de los señalizadores de las
;                      interrupciones
;*****************************************************************************************
        MOVWF   INTCON
        CLRF    CONTA           ;Pone en cero al registro auxiliar CONTA
prima   MOVLW   0XFF            ;Carga el valor inicial en el TMR0
        MOVWF   TMR0_OP
        BTFSC   PORTA,4         ;Verifica el estado del bit 4 del Puerto A, si RA4=0 omite la
                                ;siguiente instrucción
        GOTO    prima           ;Si RA4=1 brinca a la etiqueta PRIMA
bucle   MOVF    CONTA,0         ;Carga el contenido de CONTA en W y al aplicar la operación
        XORLW   02              ;XOR con el valor 02 se verifica si el ciclo de conteo se ha
                                ;efectuado 2 veces
        BTFSS   ESTADO,2        ;Si el ciclo se ha efectuado 2 veces, se comprueba el estado del bit
                                ;2 del registro ESTADO, si es 1 se omite la siguiente instrucción
        GOTO    bucle           ;Si el valor del bit 2 del registro ESTADO es cero brinca a la
                                ;etiqueta BUCLE
        CLRF    CONTA           ;Pone en ceros el registro CONTA
        BTFSS   PORTB,1         ;Comprueba el valor del bit 1 del Puerto B, si es 1, omite la
                                ;siguiente instrucción
        GOTO    prende          ;Si el bit 1 del Puerto B esta en 0, brinca a la etiqueta PRENDE
apaga   BCF     PORTB,1         ;Pone en 0 el bit 1 del Puerto B
        GOTO    bucle           ;Brinca a la etiqueta BUCLE
prende  BSF     PORTB,1         ;Pone en 1 el bit 1 del Puerto B
        GOTO    bucle           ;Brinca a la etiqueta BUCLE

;******************** Principia la rutina de servicio de interrupción (RSI)  *********************
rsi     BTFSS   INTCON,2        ;Comprueba el estado del bit señalizador de desbordamiento del TMR0
                                ;en el bit 2 del registro INTCON, si es 1 omite la siguiente
                                ;instrucción
        RETFIE                  ;Si el bit 2 de INTCON es 0, retorna de la interrupción y pone a 1
                                ;el bit 7 de INTCON
        INCF    CONTA,1         ;Incrementa en una unidad a CONTA y el resultado lo almacena en el
                                ;mismo registro
        MOVLW   0XFF            ;Carga el valor inicial en el TMR0
        MOVWF   TMR0_OP
        BCF     INTCON,2        ;Se devuelve a INTCON los valores iniciales
        BSF     INTCON,5        ;de los bits 2 y 5
```

```
        RETFIE                  ;Retorna de la interrupción y pone a 1 el bit 7 de INTCON
        END                     ;Fin de programa

*********************************************************************************
```

b) Ensamble el archivo TMR0EXT.ASM y grabe el microcontrolador con el archivo TMR0EXT.HEX. Conforme el procedimiento seguido en la actividad 1 de la práctica 1.

c) Inserte el módulo de monitores LED en la tarjeta principal del sistema de entrenamiento teniendo cuidado de insertarlo en conector 2, en la línea correspondiente al Puerto B.

d) Polarice al circuito del generador de pulsos digitales con las líneas dispuestas para este propósito a través del conector 5 del sistema de entrenamiento.

e) Conecte la salida del circuito generador de pulsos a la terminal RA4 del conector 5 del sistema de entrenamiento

f) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal. Encienda el sistema de entrenamiento y anote sus observaciones.

g) Varíe la frecuencia de oscilación del generador de pulsos mediante el potenciómetro. Observará que el diodo D2 se encenderá y apagará cada cuatro pulsos proporcionados por el generador.

h) Modifique en el programa TMR0EXT.ASM con diferentes valores para el conteo en la subrutina *bucle,* en la instrucción *xorlw* y repita de los pasos b), al f), anote sus observaciones.

i) Modifique en el programa TMR0EXT.ASM con diferentes valores hexadecimales para la carga del valor inicial del TMR0, repita de los pasos b), al f) y anote sus observaciones.

Actividad 2. Interrupción externa

Para comprender el funcionamiento de una interrupción por activación de la terminal RB0/INT, se propone el listado del programa INTEXT.ASM, Se empleará el sistema de entrenamiento, el módulo de monitores LED, el módulo con push-button y un generador de pulsos de frecuencia variable como el mostrado en la actividad 1.

a) Capture el listado del programa INTEXT.ASM que se muestra a continuación, en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre y la extensión indicada y guárdelo.

```
*********************************************************************************
;Programa INTEXT.ASM hace parpadear una led controlado a través del bit 1 del Puerto B si se
;acciona el pulsador conectado en RA0, si se produce una interrupción la rutina RSI realiza otra
;operación y una vez terminada esta rutina devuelve el control al programa principal.

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

              LIST    P=16F84         ;Programa escrito para el PIC 16F84
              RADIX   HEX             ;Mientras no se indique otra base, todos los valores
numéricos
                                      ;se expresan en Hexadecimal

ESTADO        EQU     03h     ;El registro ESTADO ocupa la posición 03h de la memoria de datos
PUERTOA       EQU     05h     ;El Puerto A está en la posición 05h de la memoria de datos
PUERTOB       EQU     06h     ;El Puerto B está en la posición 06h de la memoria de datos
INTCON        EQU     0Bh     ;El registro INTCON ocupa la posición 0Bh de la memoria de datos
OPCION        EQU     81h     ;El registro OPCION ocupa la posición 81h de la memoria de datos
RP0           EQU     05h     ;RP0 es el bit 5 del registro ESTADO
INTDEG        EQU     06h     ;INTDEG es el bit 6 del registro OPCION
INTF          EQU     01h     ;INTF es el bit 1 del registro INTCON
```

```
INTE            EQU     04h     ;INTE es el bit 4 del registro INTCON
GIE             EQU     07h     ;GIE es el bit 7 del registro INTCON
F               EQU     1       ;F=1 cuando el resultado de una operación entre W con un registro se
                                ;almacena en el registro
W               EQU     0       ;W=0 cuando el resultado de una operación entre W con un registro,
el
                                ;resultado se almacena en W
C               EQU     0       ;C es el bit 0 del registro ESTADO
PDel0           EQU     0C      ;El registro Pdel0 ocupa la posición 0C de la memoria de datos
PDel1           EQU     0D      ;El registro Pdel1 ocupa la posición 0D de la memoria de datos

                ORG     00H             ;La directiva ORG indica al programa en donde escribir la
                                        ;siguiente instrucción, que es el vector de RESET
                goto    Inicializa      ;Brinca a la dirección de la etiqueta "Inicializa"
                ORG     04H             ;Indica que la siguiente instrucción se escriba en la
                                        ;dirección 04 de la memoria de programa (Vector de
                                        ;interrupción)
                goto    rsi             ;Salta a la dirección de inicio de la Rutina de Servicio de
                                        ;Interrupción indicada por la etiqueta "rsi"

Inicializa      bsf     ESTADO,RP0      ;Se selecciona la página 1 de la memoria de datos
                movlw   b'00011111'     ;Se carga en W el valor binario "00011111" (1fh)
                movwf   PUERTOA         ;Se configura el Puerto A como entradas
                movlw   b'00000001'     ;Se carga en W la literal en binario "00000001" (01h)
                movwf   PUERTOB         ;Se configura el bit "0" del Puerto B como entrada y los
                                        ;demás como salidas
                bcf     OPCION,INTDEG   ;Se configura el Selector de Flanco de Interrupción (INTDEG)
                                        ;para que se acepte la interrupción en el flanco de bajada
                bcf     ESTADO,RP0      ;Regresa a la página 0 de la memoria de datos
                bcf     INTCON,INTF     ;Limpia la bandera de interrupción INT
                bsf     INTCON,GIE      ;Habilita al PIC para aceptar interrupciones
                bsf     INTCON,INTE     ;Permite la interrupción por activación de la terminal INT
                clrf    PUERTOB         ;Pone en ceros el Puerto B

Principal       btfsc   PUERTOA,0       ;Prueba el bit 0 del Puerto A y brinca si esta en "0"
                goto    Apaga           ;Si RA0 está en "1" brinca a la etiqueta "Apaga"
Enciende        bsf     PUERTOB,1       ;Pone en "1" el bit 1 del Puerto B
                goto    Principal       ;Salta a la etiqueta "Principal"
Apaga           bcf     PUERTOB,1       ;Pone en "0" el bit 1 del Puerto B
                goto    Principal       ;Salta a la etiqueta "Principal"

;****************************** Rutina de Servicio de Interrupción ********************************

rsi             btfss   INTCON,INTF     ;Comprueba si la interrupción se produció por la activación
de
                                        ;la terminal RB0/INT, si el bit INTF=1 se salta una
instrucción
                retfie                  ;Si el bit INTF esta en "0", sale de la rutina de servicio de
                                        ;interrupción, regresando el control al programa principal
                movlw   b'00000001'     ;Carga el valor binario al registro W
                bcf     ESTADO,0        ;Pone a cero el bit CARRY del registro ESTADO
                clrf    PUERTOB         ;Pone a ceros el Puerto B
                movwf   CUENTA          ;Carga el valor binario al registro auxiliar CUENTA
right           rrf     CUENTA,F        ;Enciende siguiente led a la derecha
                movfw   CUENTA          ;Carga el ultimo valor del registro CUENTA a W
                movwf   PUERTOB         ;Carga el valor de W al PUERTO B
                call    retardo         ;Llama a la subrutina RETARDO
                btfss   CUENTA,LSB      ;Alcanzado final por la derecha?
                goto    right           ;Si no, regresa a RIGHT y vuelve a rotar a la derecha
                clrf    PUERTOB         ;Si si, pone en ceros el PUERTO B
                bcf     INTCON,INTF     ;Limpia el estado de la Bandera de interrupción INT
                bsf     INTCON,INTE     ;Habilita la interrupción por activación de la terminal
RB0/INT
                clrf    PUERTOB         ;Pone a ceros el PUERTO B
                retfie                  ;Sale de la rutina de servicio de interrupción, regresando el
                                        ;control al programa principal

retardo         movlw   .239            ;Se carga el número de repeticiones
                movwf   PDel0           ;para el registro PDel0
PLoop1          movlw   .232            ;Se carga el número de repeticiones
                Movwf   PDel1           ;para el registro PDel1
PLoop2          clrwdt                  ;Pone a ceros al WDT (se consume un ciclo de retardo)
PDelL1          goto    PDelL2          ;Se consumen 2 ciclos de retardo
PDelL2          goto    PDelL3          ;Se consumen 2 ciclos de retardo
PDelL3          clrwdt                  ;Se consumen 1 ciclo de retardo
                decfsz  PDel1,1         ;Se decrementa el contenido de Pdel1 y revisa si llegó a cero
```

```
                goto    PLoop2          ;si no, salta a etiqueta PLoop2
                decfsz  PDel0,1         ;Decrementa Pdel0 una unidad y revisa si Pdel0 = 0
                goto    PLoop1          ;si no, salta a Ploop1
PDelL4          goto    PDelL5          ;Se consumen 2 ciclos de retardo
PDelL5          goto    PDelL6          ;Se consumen 2 ciclos de retardo
PDelL6          goto    PDelL7          ;Se consumen 2 ciclos de retardo
PDelL7          clrwdt                  ;Se consume 1 ciclo de retardo
                return                  ;Termina ciclo de retardo

                END                     ;Directiva de fin de programa
```

****

b) Ensamble el archivo INTEXT.ASM y grabe el microcontrolador con el archivo INTEXT.HEX conforme el procedimiento seguido en la actividad 1 de la práctica 1.

c) Inserte el módulo de monitores LED en la tarjeta principal del sistema de entrenamiento en el conector 1, en la línea correspondiente al Puerto B. A su vez, inserte el módulo con Push-button en el conector 2 en la línea correspondiente también al Puerto B.

d) Polarice el generador de pulsos y conecte su salida a la terminal correspondiente a la línea RA0 en el conector 5 del sistema de entrenamiento, notará que el diodo D2 del módulo de monitores LED parpadea al mismo ritmo que el generador, quedando demostrado que el microcontrolador está atendiendo las instrucciones del programa principal.

e) Oprima el botón S1 del módulo con Push-Button. Anote sus observaciones.

Observará que el microcontrolador deja de atender al programa principal pues ignora los pulsos introducidos a través de la línea RA0, y comenzará a atender la Rutina de Servicio de Interrupción (RSI), que es un programa que se ejecuta al presentarse la interrupción.

Actividad 3. Cuestionario

a) Dibuje el diagrama a bloques del programa TMR0EXT.ASM.

b) Para el funcionamiento de un programa se requiere que el PIC registre un evento cada 10 pulsos digitales aplicados externamente a través de la terminal RA4/T0CKI, la interrupción externa responda al flanco descendente, las resistencias de polarización estén deshabilitadas, y responda al flanco ascendente de la señal, escriba el valor binario que debe cargarse al registro OPCION.

c) Si en el programa TMR0EXT.ASM el registro INTCON se carga con el valor binario b'01101111' y se presenta una interrupción por desbordamiento del TMR0, describa brevemente lo que sucederá internamente en el PIC.

d) Modifique el programa INTEXT.ASM para que la interrupción se efectúe con un flanco positivo.

e) Investigue las diferentes fuentes de interrupción con las que cuenta el PIC16F84.

Práctica Nº 4

LA MEMORIA EEPROM DE DATOS.

OBJETIVO: Que el alumno se familiarice con los procedimientos de escritura y lectura de la memoria EEPROM de datos.

DESARROLLO.

Actividad 1

Para cumplir con el objetivo se propone el listado del programa EEPROM.ASM, Se empleará el sistema de entrenamiento y los módulos de monitores LED, Push-button y Dip-switch.

a) Capture el listado del programa EEPROM.ASM que se muestra a continuación, en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre y la extensión indicados y guárdelo.

```
;*******************************************************************************
;EEPROM.ASM Este programa permite al usuario grabar y leer datos en las primeras 16 posiciones de
;la memoria EEPROM de datos

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=16F84
        RADIX   HEX

ESTADO  EQU     03      ;Dirección del registro ESTADO en la memoria de datos
PUERTOA EQU     05      ;Dirección del registro PUERTO A
PUERTOB EQU     06      ;Dirección del registro PUERTO B
EECON1  EQU     88      ;Dirección del registro EECON1 en la memoria de datos
EECON2  EQU     89      ;Dirección del registro EECON2 en la memoria de datos
EEDATA  EQU     08      ;Dirección del registro EEDATA en la memoria de datos
EEADR   EQU     09      ;Dirección del registro EEADR en la memoria de datos
RD      EQU     00      ;Bit 0 del registro EECON2
WR      EQU     01      ;Bit 1 del registro EECON2
WREN    EQU     02      ;Bit 2 del registro EECON2
EEIF    EQU     04      ;Bit 4 del registro EECON2
PDEL0   EQU     0C      ;Dirección del registro PDel0 auxiliar en la rutina de retardo
PDEL1   EQU     0D      ;Dirección del registro PDel0 auxiliar en la rutina de retardo
CONTA   EQU     0E      ;Dirección del registro CONTA auxiliar en la rutina AVISA
DIREC   EQU     0F      ;Dirección del registro DIREC auxiliar para leer/escribir la memoria
                        ;EEPROM de datos

        ORG     00      ;Dirección del vector RESET, en la dirección 00h de la memoria de programa
                        ;se grabará la siguiente instrucción
        goto    inicio  ;Indica salto a la dirección indicada por la etiqueta INICIO
        ORG     05      ;Indica que se continúe grabando al programa a partir de la dirección 05h
                        ;de la memoria de programa

;*********************************  RUTINA DE MENU DE INICIO  ************************************
inicio bsf      ESTADO,5        ;Selecciona la página 1 de la memoria de datos
       movlw    b'00000111'     ;Carga el valor binario 00000111 en el registro W
       movwf    PUERTOA         ;Configura al PUERTO A: RA4 y RA3 salidas, RA2, RA1 y RA0 entradas
       bcf      ESTADO,5        ;Selecciona la página 0 de la memoria de datos

;Se inicia una exploración de las terminales a las que se tienen conectados los botones de
;LEE(RA0),ESCRIBE (RA1) y FIN (RA2)

menu    clrf    PUERTOA         ;Pone a ceros las salidas del PUERTO A (apaga el monitor FUNCION)
        btfss   PUERTOA,0       ;Prueba el estado del botón LEE, si RA0=1 salta una instrucción
        call    retardo         ;Si RA0=0 llama rutina de retardo para eliminar transitorios
        btfss   PUERTOA,0       ;Se vuelve a probar el botón LEE, si RA0=1 salta una instrucción
```

```
        call    preleer         ;Si RA0=0 llama rutina de lectura de la memoria
        btfss   PUERTOA,1       ;Prueba el estado del botón ESCRIBE, si RA1=1 salta una instrucción
        call    retardo         ;Si RA1=0 llama rutina de retardo para eliminar transitorios
        btfss   PUERTOA,1       ;Se vuelve a probar el botón ESCRIBE, si RA1=1 salta una instrucción
        call    pregra          ;Si RA1=0 llama rutina de escritura de la memoria
        goto    menu            ;Salta a la etiqueta MENU para reiniciar la exploración

;*****************************  RUTINA DE LECTURA DE LA MEMORIA  *******************************
preleerbsf      ESTADO,5        ;Selecciona la página 1 de la memoria de datos
        clrf    PUERTOB         ;Configura al PUERTO B como salidas
        bcf     ESTADO,5        ;Selecciona a la página 0 de la memoria de datos
        clrf    PUERTOB         ;Pone a ceros el PUERTO B (apaga los monitores)
        clrf    DIREC           ;Pone a ceros al registro DIREC

;Se hace la exploración en las terminales de los botones esperando la orden de FIN o de LEER

menu2   bsf     PUERTOA,3       ;Enciende monitor FUNCION
        btfss   PUERTOA,2       ;Prueba el botón FIN, si RA2=1 salta una instrucción
        call    retardo         ;Si RA2=0, llama rutina de RETARDO para eliminar transitorios
        btfss   PUERTOA,2       ;Vuelve a probar botón FIN, si RA2=1 salta una instrucción
        goto    salir           ;Si RA2=0 brinca a la etiqueta SALIR
        btfss   PUERTOA,0       ;Prueba el botón LEE, si RA0=1 salta una instrucción
        call    retardo         ;Si RA0=0, llama rutina de RETARDO para eliminar transitorios
        btfss   PUERTOA,0       ;Vuelve a probar botón LEE, si RA0=1 salta una instrucción
        goto    leer            ;Si RA0=0 salta a la rutina LEER
        goto    menu2           ;Salta a etiqueta MENU2 y reinicia la exploración de botones

;Rutina en la que se realiza la lectura del dato contenido en la dirección indicada por el
;registro DIREC y se presenta en los monitores del PUERTO B, además se incrementa el registro
;DIREC y en caso de alcanzar el máximo de direcciones llama a una subrutina e indica mediante el
;monitor conectado en RA3

leer    bcf     PUERTOA,3       ;Apaga monitor FUNCION
        movf    DIREC,0         ;Mueve el contenido del registro DIREC al registro W
        movwf   EEADR           ;Se carga la dirección a ser leída en el registro EEADR
        bsf     ESTADO,5        ;Selecciona la página 1 de la memoria de datos
        bsf     EECON1,RD       ;Habilita lectura de EEPROM de datos
        bcf     ESTADO,5        ;Selecciona la página 0 de la memoria de datos
        movf    EEDATA,0        ;Mueve el contenido del registro EEDATA a W
        movwf   PUERTOB         ;Se muestra el dato contenido en la dirección indicada en los
                                ;monitores del PUERTO B
        incf    DIREC,1         ;Incrementa al registro DIREC y actualiza el registro
        btfsc   DIREC,4         ;Prueba el bit 4 del registro DIREC para saber si se ha alcanzado la
                                ;dirección 16 de la memoria EEPROM de datos
        call    avisa           ;Si la dirección es 16 llama la rutina AVISA
        call    retardo         ;Llama la rutina retardo para hacer visible al usuario cuando la
                                ;operación ha terminado
        goto    menu2           ;Regresa a la exploración de los botones del menú de lectura

salir   clrf    PUERTOB         ;Apaga los monitores del PUERTO B
        return                  ;Regresa al MENU DE INICIO

;***********************  RUTINA DE ESCRITURA DE LA MEMORIA  ************************************
pregra  bsf     ESTADO,5        ;Selecciona la página 1 de la memoria de datos
        movlw   0xFF            ;Carga al registro W con la literal hexadecimal FF
        movwf   PUERTOB         ;Configura al PUERTO B como entradas
        bcf     ESTADO,5        ;Selecciona la Página 0 de la memoria de datos
        clrf    DIREC           ;Pone a ceros el registro DIREC

;Se hace una exploracion en las terminales de los botones esperando la orden de FIN o de ESCRIBE

menu3   bsf     PUERTOA,3       ;Enciende el monitor FUNCION
        btfss   PUERTOA,2       ;Prueba botón FIN, si RA2=1 salta una instrucción
        call    retardo         ;Si RA2=0 llama rutina de RETARDO para eliminar transitorios
        btfss   PUERTOA,2       ;Prueba botón FIN, si RA2=1 salta una instrucción
        return                  ;Si RA2=0 regresa al MENU DE INICIO
        btfss   PUERTOA,1       ;Prueba botón ESCRIBE, si RA1=1 salta una instrucción
        call    retardo         ;Si RA1=0 llama rutina de RETARDO para eliminar transitorios
        btfss   PUERTOA,1       ;Prueba botón ESCRIBE, si RA1=1 salta una instrucción
        goto    graba           ;Si RA1=0 llama subrutina GRABA
        goto    menu3           ;Después de avisar, brinca a MENU3

;Esta rutina sigue todos los pasos para escribir un dato almacenado en el registro EEDATA en una
;dirección almacenada en el registro EEADR, además se incrementa el registro DIREC encargado de
;llevar el conteo de las direcciones que se escriben y para determinar cuando se llena la memoria
```

```
graba   bcf     PUERTOA,3       ;Apaga al monitor FUNCION
        movf    DIREC,0         ;Mueve contenido de registro DIREC a W
        movwf   EEADR           ;Se carga el registro EEADR con el contenido de DIREC
        movf    PUERTOB,0       ;Se carga la información introducida por el PUERTO B en W
        movwf   EEDATA          ;Se carga la información de W en el registro EEDATA
        bsf     ESTADO,5        ;Selecciona la página 1 de la memoria de datos
        bsf     EECON1,WREN     ;Se habilita la memoria EEPROM para escritura
        movlw   0x55            ;Secuencia requerida por Microchip para poder
        movwf   EECON2          ;iniciar el proceso de la escritura en la memoria
        movlw   0xAA            ;EEPROM de datos antes de habilitar el bit 1 (WR) del
        movwf   EECON2          ;registro INTCON1
        bsf     EECON1,WR       ;Se habilita permiso de escritura de la EEPROM
espera  btfss   EECON1,EEIF     ;Prueba si ya se terminó de grabar la EEPROM de datos
        goto    espera          ;si no, espera
        bcf     EECON1,EEIF     ;Pone en cero la bandera EEIF una vez que ha finalizado el proceso
                                ;de escritura
        bcf     EECON1,WREN     ;Deshabilita la escritura de la EEPROM
        bcf     ESTADO,5        ;Selecciona la página 0 de la memoria de datos
        incf    DIREC,1         ;Incrementa en uno al registro DIREC
        btfsc   DIREC,4         ;Comprueba el registro DIREC para verificar si se han llenado las 16
                                ;direcciones de la memoria
        call    avisa           ;Una vez que se llena la memoria llama subrutina AVISA
        call    retardo         ;Llama rutina de RETARDO para hacer visible al usuario cuando se ha
                                ;finalizado el proceso de escritura
        goto    menu3           ;Regresa al MENU DE ESCRITURA

;*************************  RUTINA DE AVISO DE ULTIMA POSICIÓN DE MEMORIA  **********************
avisa   movlw   0x03            ;Carga el número de veces que el monitor de función va a parpadear
                                ;para hacer la indicación
        movwf   CONTA           ;Mueve el contenido de W al registro CONTA
loop    bsf     PUERTOA,3       ;Enciende el monitor FUNCION
        call    ret2            ;Llama rutina de retardo de 500ms (RET2)
        clrf    PUERTOA         ;Apaga el monitor FUNCION
        call    ret2            ;Llama rutina de retardo de 500ms (RET2)
        decfsz  CONTA,1         ;Decrementa el registro CONTA y prueba si CONTA=0
        goto    loop            ;Si CONTA no es igual a cero salta a etiqueta LOOP
        clrf    DIREC           ;Pone a ceros el registro DIREC
        return                  ;Regresa de subrutina

;*****************************  RUTINA DE RETARDO DE 250ms  ************************************
retardomovlw    .197            ;Literal que indica el número de repeticiones del ciclo externo
        movwf   PDEL0           ;Carga contenido de W en registro PDel0
PLoop1  movlw   .253            ;Literal que indica el número de repeticiones del ciclo interno
        movwf   PDEL1           ;Carga contenido de W en registro PDel1
PLoop2  clrwdt                  ;Instrucción que consume 1 ciclo de retardo
        clrwdt                  ;Instrucción que consume 1 ciclo de retardo
        decfsz  PDEL1,1         ;Decrementa registro PDel1 y checa si se llegó a cero
        goto    PLoop2          ;Si no, salta a etiqueta PLoop2
        decfsz  PDEL0,1         ;Decrementa registro PDel0 y checa si se llegó a cero
        goto    PLoop1          ;Si no, salta a etiqueta Ploop1
PDelL1  goto    PDelL2          ;Instrucción que consume 2 ciclos de retardo
PDelL2
        return                  ;Retardo completado, regresa de subrutina

;*******************************  RUTINA DE RETARDO DE 500ms  ************************************
ret2    call    retardo         ;Llama rutina de RETARDO de 250ms
        call    retardo         ;Llama rutina de RETARDO DE 250ms
        return                  ;Retorna de subrutina
        END

*********************************************************************************
```

b) Ensamble el archivo EEPROM.ASM y grabe el microcontrolador con el archivo EEPROM.HEX. Conforme el procedimiento seguido en la actividad 1 de la práctica 1.

c) Inserte los módulos en la tarjeta principal del sistema de entrenamiento según se indica a continuación: el módulo de monitores LED en el conector 1 en la línea correspondiente al Puerto B, el módulo con Push-Button en el conector 3 en la línea correspondiente al Puerto B y, por último, el módulo con Dip-Switch en el conector 2 en la línea correspondiente al Puerto A.

Al encender el sistema se entra inmediatamente a la rutina de MENÚ DE INICIO en donde solamente están habilitados los botones LEE y ESCRIBE. Para escribir un dato en la memoria EEPROM de datos se debe seguir la secuencia siguiente:

a) Presione el botón ESCRIBE, esto debe hacer encender el monitor FUNCION indicando que el PIC se encuentra ejecutando la subrutina de ESCRITURA DE LA MEMORIA, En ésta subrutina solamente están habilitados los botones FIN y ESCRIBE.

b) Mediante el módulo con Push-Button establezca el dato que quiera escribir en la EEPROM de datos y enseguida

c) Presione una vez el botón ESCRIBE, con esto el monitor FUNCION se apagará por una fracción de tiempo volviéndose a encender, indicando con esto que el dispositivo ha aceptado el dato y se encuentra en espera de un nuevo dato.

d) Repita los pasos 2 y 3 para escribir un nuevo dato. Nota: las direcciones se incrementan secuencialmente desde 0h hasta Fh cada vez que se presiona el botón ESCRIBE. El monitor FUNCION parpadea 3 veces al llegar a la dirección Fh como una indicación al usuario de que la memoria se ha llenado.

e) Para salir de la función de escritura bastará con presionar el botón FIN cuando se desee regresando al MENU DE INICIO.

Para leer los datos grabados en la memoria el procedimiento es el siguiente:

a) En la rutina MENU DE INICIO presione una vez el botón LEE. El monitor FUNCION se encenderá indicando que el dispositivo se encuentra en la subrutina ESCRITURA DE LA MEMORIA. En ésta subrutina solamente están habilitados los botones FIN y LEE.

b) Quite el módulo con Push-Button

c) Presione una vez el botón LEE. El monitor FUNCION se apagará por un breve instante indicando que la orden ha sido aceptada y en los monitores se mostrarán los datos secuencialmente en el mismo orden que en el que fueron escritos cada vez que se presione el botón LEE.

d) Para regresar al MENU DE INICIO deberá presionar el botón FIN.

e) Introduzca datos diferentes siguiendo el procedimiento descrito anteriormente hasta llenar la memoria EEPROM de datos.

f) Apague el sistema de entrenamiento y después de 1 minuto vuelva a encenderlo.

g) Siga el procedimiento de lectura y verifique el contenido de la memoria EEPROM de datos.

h) Anote sus observaciones.

i) Modifique el programa EEPROM.ASM para que acepte un máximo de 8 datos en la memoria EEPROM de datos.

Actividad 4: CUESTIONARIO.

a) ¿Cuál es la máxima capacidad de la memoria EEPROM de datos del PIC16F84?

b) ¿Cuántos ciclos de escritura/borrado soporta éste tipo de memoria?

c) ¿Típicamente cuánto tiempo dura un ciclo de escritura de una posición de la memoria EEPROM de datos?

d) ¿Cuánto tiempo le toma al microcontrolador efectuar una operación de lectura?

Práctica Nº 5

MANEJO DE UN TECLADO MATRICIAL.

OBJETIVO: Al finalizar esta práctica el alumno tendrá las bases en el manejo y control de un teclado matricial a través de la implementación de una tabla de decodificación.

DESARROLLO

Actividad 1. Para cumplir con el objetivo propuesto, se propone el listado del programa MATRIZ.ASM. Se empleará el sistema de entrenamiento, el módulo de teclado matricial (figura 5.11) y el módulo de Display de 7 segmentos (figura 5.4).

a) Escriba el listado de programa MATRIZ.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre y la extensión propuesto y guárdelo.

```
;*********************************************************************
;MATRIZ.ASM, PROGRAMA QUE MANEJA UN TECLADO MATRICIAL Y PRESENTA EN UN DISPLAY DE 7 SEGMENTOS EL
;CODIGO DE LA TECLA PRESIONADA

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

              LIST    P=16F84         ;Selecciona el tipo de microcontrolador a usar
              RADIX   HEX             ;Especifica la numeración base

PC            EQU     0X02    ;Registro PC en la dirección 02 de la memoria de datos
ESTADO        EQU     0X03    ;Registro ESTADO en la dirección 03 de la memoria de datos
TRISA         EQU     0X05    ;Registro TRISA en la dirección 05 del banco 1 de la memoria de
datos
PTOA          EQU     0X05    ;Registro PTOA en la dirección 05 del banco 0 de la memoria de datos
TRISB         EQU     0X06    ;Registro TRISB en la dirección 06 del banco 1 de la memoria de
datos
PORTB         EQU     0X06    ;Registro PORTB en la dirección 06 del banco 0 de la memoria de
datos
CONTADOR1     EQU     0X0C    ;Registro CONTADOR1 en la dirección 0C de la memoria de datos
CONTADOR2     EQU     0X0D    ;Registro CONTADOR2 en la dirección 0D de la memoria de datos
CONTADOR3     EQU     0X0E    ;Registro CONTADOR3 en la dirección 0E de la memoria de datos
TECLA         EQU     0X0F    ;Registro TECLA en la dirección 0F de la memoria de datos
DIGITO        EQU     0X10    ;Registro DIGITO en la dirección 10 de la memoria de datos
W             EQU     0X00    ;Cuando se escriba W significa 0
C             EQU     0X01    ;C es el bit 1 en el registro ESTADO
Z             EQU     0X02    ;Z es el bit 2 en el registro ESTADO
RP0           EQU     0X05    ;RP0 es el bit 5 en el registro ESTADO

              ORG     0x00            ;La siguiente instrucción se escribirá en el vector de RESET
              GOTO    inicializa      ;Brinca a la dirección apuntada por la etiqueta INICIALIZA
              ORG     0x05            ;El programa se empieza a escribir después del vector de
                                      ;interrupción

;*********************************  TABLA DE DECODIFICACION  *********************************
tabla         ADDWF   PC,1            ;Se suma el contenido de W al registro PC
;                       .gfedcba      Indican el segmento  del display correspondiente a cada bit
              RETLW   B'10001110'     ;Retorna de subrutina TABLA y carga a W con el
                                      ;código binario correspondiente al carácter F
              RETLW   B'10000110'     ;Código binario correspondiente al carácter E
              RETLW   B'10100001'     ;Código binario correspondiente al carácter D
              RETLW   B'11000110'     ;Código binario correspondiente al carácter C
              RETLW   B'10110000'     ;Código binario correspondiente al carácter 3
              RETLW   B'10000010'     ;Código binario correspondiente al carácter 6
              RETLW   B'10010000'     ;Código binario correspondiente al carácter 9
              RETLW   B'10000011'     ;Código binario correspondiente al carácter B
              RETLW   B'10100100'     ;Código binario correspondiente al carácter 2
```

```
                RETLW   B'10010010'    ;Código binario correspondiente al carácter 5
                RETLW   B'10000000'    ;Código binario correspondiente al carácter 8
                RETLW   B'10001000'    ;Código binario correspondiente al carácter A
                RETLW   B'11111001'    ;Código binario correspondiente al carácter 1
                RETLW   B'10011001'    ;Código binario correspondiente al carácter 4
                RETLW   B'11111000'    ;Código binario correspondiente al carácter 7
                RETLW   B'11000000'    ;Código binario correspondiente al carácter 0

;************************************  PROGRAMA PRINCIPAL  ************************************
inicializa      BSF     ESTADO,RP0     ;Selecciona la página 1 de la memoria de datos
                MOVLW   0XF0           ;La carga de W con la literal F0h configura al Puerto B: los
                                       ;cuatro bits más significativos como entradas y los cuatro
                MOVWF   TRISB          ;menos significativos como salidas
                MOVLW   0XFE           ;La carga de W con FEh configura al puerto A de manera que
                MOVWF   TRISA          ;el bit 0 como salida y los demás como entradas
                BCF     ESTADO,RP0     ;Selecciona la página 0 de la memoria de datos
                CLRF    CONTADOR1      ;Lleva a cero el contenido del registro CONTADOR1
                CLRF    CONTADOR2      ;Lleva a cero el contenido del registro CONTADOR2
                MOVLW   0x04           ;Se carga al registro CONTADOR3 con el valor 04h
                MOVWF   CONTADOR3      ;Carga al registro CONTADOR3 con el contenido de W
                BSF     PTOA,0         ;Apaga al display antes de explorar el teclado

principal       CLRF    TECLA          ;Lleva a cero al registro TECLA
                MOVLW   0x0E           ;Este valor coloca el cero que se va a rotar por las
                MOVWF   PORTB          ;terminales de salida (columnas) del puerto B en el bit RB0

chequeo_lin     BTFSS   PORTB,4        ;Se revisa el estado de los botones de la primera línea
                GOTO    ntecla         ;si hay botón accionado brinca a la etiqueta NTECLA
                INCF    TECLA          ;si no hay botón accionado incrementa el registro TECLA
                BTFSS   PORTB,5        ;Se revisa el estado de los botones de la segunda línea
                GOTO    ntecla         ;si hay botón accionado brinca a la etiqueta NTECLA
                INCF    TECLA          ;si no hay botón accionado incrementa el registro TECLA
                BTFSS   PORTB,6        ;Se revisa el estado de los botones de la tercera línea
                GOTO    ntecla         ;si hay botón accionado brinca a la etiqueta NTECLA
                INCF    TECLA          ;si no hay botón accionado incrementa el registro TECLA
                BTFSS   PORTB,7        ;Se revisa el estado de los botones de la cuarta línea
                GOTO    ntecla         ;si hay botón accionado brinca a la etiqueta NTECLA
                INCF    TECLA          ;si no hay botón accionado incrementa el registro TECLA

ultima_tecla?   MOVLW   0X11           ;Al cargar este valor y restarlo del valor del registro
                SUBWF   TECLA,W        ;TECLA podemos saber si ya se completó la exploración del
                BTFSC   ESTADO,Z       ;teclado al probar el bit 2 de ESTADO, si Z=0 brinca 1 inst.
                GOTO    principal      ;Cuando Z=1 brinca a la etiqueta PRINCIPAL
                BSF     ESTADO,C       ;Pone en 1 el bit de acarreo para la rotación del cero en la
                                       ;exploración de las columnas
                RLF     PORTB          ;Pasa el cero a la siguiente columna a la derecha
                GOTO    chequeo_lin    ;Brinca a la etiqueta CHEQUEO_LIN

;*******************  RUTINA DE DECODIFICACION Y PRESENTACION EN DISPALY  **********************
ntecla          MOVF    TECLA,W        ;Se mueve el contenido de TECLA a W para proceder a la
                CALL    tabla          ;decodificación de la tecla pulsada
                MOVWF   DIGITO         ;El código de la tecla pulsada se almacena en el registro
                                       ;DIGITO
                BSF     ESTADO,RP0     ;Selecciona la página 1 de la memoria de datos para
                                       ;reconfigurar
                CLRF    TRISB          ;al puerto B como salidas
                BCF     ESTADO,RP0     ;y selecciona la página 0
                MOVF    DIGITO,W       ;Transferimos el código de la tecla pulsada al puerto B
                MOVWF   PORTB          ;para presentarlo en el display de siete segmentos
                BCF     PTOA,0         ;Se habilita al display y este presenta el dígito
                                       ;correspondiente a la tecla pulsada

pausa           DECFSZ  CONTADOR1      ;Inicia un ciclo de retardo para que el usuario pueda leer y
                GOTO    pausa          ;comprobar el dígito de la tecla pulsada
                DECFSZ  CONTADOR2
                GOTO    pausa
                DECFSZ  CONTADOR3
                GOTO    pausa
                GOTO    inicializa     ;Brinca a la etiqueta inicializa

                END                    ;Directiva de fin de programa

*******************************************************************************
```

b) Tal y como se ha procedido en las prácticas anteriores, ensamble el archivo MATRIZ.ASM y grabe el microcontrolador con el archivo MATRIZ.HEX, dicho procedimiento se describe paso a paso en la actividad 1 de la práctica 1.

c) Inserte el módulo con Display de 7 segmentos en la tarjeta principal del sistema de entrenamiento en el conector 1 y el módulo con Teclado Matricial en el conector 2 en la línea correspondiente al Puerto B.

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal y encienda el sistema de entrenamiento

e) Presione cualquier tecla y observe el funcionamiento del sistema. Anote sus observaciones

f) Modifique en el programa MATRIZ.ASM de manera que el dígito correspondiente a la tecla presionada se despliegue en el display DP4 durante 2 segundos.

Actividad 2. CUESTIONARIO.

a) Dibuje el diagrama a bloques del programa MATRIZ.ASM.

b) Mencione tres aparatos en donde se apliquen teclados matriciales.

c) ¿Cuántas terminales de un microcontrolador se emplearían para manejar un teclado de 9 teclas?

d) Tomando en cuenta el programa MATRIZ.ASM ¿Por qué es necesario poner en 1 el bit CARRY (1) del registro ESTADO antes de explorar el estado de las siguientes filas?

Práctica Nº 6

MANEJO DE DISPLAYS DE 7 SEGMENTOS POR MULTIPLEXAJE

OBJETIVO: Al finalizar esta práctica el alumno adquirirá las bases necesarias en el manejo de visualizadores con displays de 7 segmentos aplicando técnicas de multiplexaje.

DESARROLLO

Actividad 1. Para cumplir con el objetivo propuesto, se propone el listado del programa MUX7SEG.ASM Se empleará el sistema de entrenamiento, el Módulo con de Display de 7 segmentos y el Módulo de Monitores LED (opcional)

a) Escriba el listado de programa MUX7SEG.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre y la extensión propuesto y guárdelo.

```
;*******************************************************************************
;Programa MUX7SEG.ASM Este programa maneja 4 displays de 7 segmentos en forma multiplexada.

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=PIC16F84      ;Directiva que indica que el programa esta escrito para el PIC16F84
        RADIX   HEX             ;Directiva que indica que todas las literales están en base
                                ;Hexadecimal

INDO    EQU     00h     ;Dirección del Registro de Direccionamiento Indirecto (INDO)
PC      EQU     02h     ;Dirección del Registro del Contador de Programa (PC)
ESTADO  EQU     03h     ;Dirección del Registro ESTADO
FSR     EQU     04h     ;Dirección del Registro selector (FSR)
PTOA    EQU     05h     ;El puerto A está en la dirección 05 de la RAM
PTOB    EQU     06h     ;El puerto B está en la dirección 06 de la RAM
LOOPS   EQU     0Dh     ;Registro auxiliar utilizado en rutina de retardo
LOOPS2  EQU     0Eh     ;Registro auxiliar utilizado en rutina de retardo
ROTA    EQU     0Fh     ;Registro auxiliar para rotar un cero que habilite los displays
DIG1    EQU     10h     ;Registro que indica el primer dígito a mostrar
DIG2    EQU     11h     ;Registro que indica el segundo dígito a mostrar
DIG3    EQU     12h     ;Registro que indica el tercer dígito a mostrar
DIG4    EQU     13h     ;Registro que indica el cuarto dígito a mostrar
TRISA   EQU     05h     ;Dirección del registro para configuración del Puerto A
TRISB   EQU     06h     ;Dirección del registro para configuración del Puerto B
Z       EQU     02h     ;Bandera de Cero del registro ESTADO
C       EQU     00h     ;Bandera de Carry del registro ESTADO
W       EQU     00h     ;Indica que el resultado se guarda en W

        org     0       ;En el vector de RESET (dirección 00 de la memoria de programa)
        goto    inicio  ;se indica un salto a la dirección indicada por la etiqueta INICIO
        org     5       ;El programa se escribe a partir de la dirección 5 en memoria de programa

;*************************  Subrutina de retardo de 3 milisegundos  ****************************
retardo movlw   d'3'            ;El registro LOOPS contiene el número
        movwf   LOOPS           ;de milisegundos totales del retardo
top2    movlw   d'110'          ;El registro LOOPS2 contiene el número
        movwf   LOOPS2          ;de repeticiones para retardo de 1 milisegundo
top     nop                     ;Se consume un ciclo de instrucción
        nop
        nop
        nop
        nop
        nop
        decfsz  LOOPS2          ;Pregunta si termino 1 milisegundo
        goto    top             ;si no, brinca a la etiqueta TOP
        decfsz  LOOPS           ;Pregunta si termina el retardo
        goto    top2            ;si no, regresa a la etiqueta TOP2
```

```
        retlw   0

;******  La TABLA contiene los valores para encender segmentos del display de ánodo común  ******
tabla   addwf   PC              ;Se suma W al PC para brincar al código que se quiere presentar en
                                ;los displays
                ; .gfedcba     ;Correspondencia de los bits con los segmentos de los displays
        retlw   b'11000000'     ;Código correspondiente al dígito 0
        retlw   b'11111001'     ;Código correspondiente al dígito 1
        retlw   b'10100100'     ;Código correspondiente al dígito 2
        retlw   b'10110000'     ;Código correspondiente al dígito 3
        retlw   b'10011001'     ;Código correspondiente al dígito 4
        retlw   b'10010010'     ;Código correspondiente al dígito 5
        retlw   b'10000010'     ;Código correspondiente al dígito 6
        retlw   b'11111000'     ;Código correspondiente al dígito 7
        retlw   b'10000000'     ;Código correspondiente al dígito 8
        retlw   b'10010000'     ;Código correspondiente al dígito 9
;*****************************************************************************************
***

inicio bsf      ESTADO,5        ;Se selecciona el banco 1 de la memoria RAM
       movlw    E0h             ;Se carga el registro W con E0h
       movwf    TRISA           ;Se programan las terminales del Puerto A como salidas
       movlw    00h             ;Se carga el registro W con 00h
       movwf    TRISB           ;Se programa el puerto B como salidas
       bcf      ESTADO,5        ;Se selecciona el banco 0 de la memoria RAM

       movlw    01h             ;En esta sección se seleccionan los datos que se muestran
       movwf    DIG1            ;en los displays, en este caso, 1, 2, 3 y 4
       movlw    02h
       movwf    DIG2
       movlw    03h
       movwf    DIG3
       movlw    04h
       movwf    DIG4

       movlw    0x0F            ;Se envían unos a los transistores del módulo conectados a los bits
       movwf    PTOA            ;0, 1, 2 y 3 del Puerto A para apagarlos
empe   movlw    0xF7            ;Este valor sirve para iniciar un 0 en el bit 4
       movwf    ROTA            ;del registro de rotación ROTA
       movf     DIG1,W          ;Con el registro selector (FSR) se apunta
       movwf    FSR             ;al primer dato que se va a mostrar
disp   movlw    0xFF            ;Coloca en unos el dato de los segmentos del display
       movwf    PTOB            ;para apagarlos
       movf     ROTA,W          ;Se pasa el contenido de ROTA al registro W y luego al Puerto A
       movwf    PTOA            ;para activar al display correspondiente
       movf     INDO,W          ;Lee el dato del registro apuntado por el FSR y lo guarda en W
       call     tabla           ;Se llama a la TABLA para que a través de W se descargue en el
       movwf    PTOB            ;Puerto B el dígito correspondiente a mostrar en el display activado
       call     retardo         ;Se llama la rutina de retardo de 3 milisegundos para visualización
       btfss    ROTA,0          ;Revisa si terminaron 4 rotaciones, si es así, brinca una
instrucción
       goto     empe            ;si ya rotaron todos, vuelve a empezar brincando a la etiqueta EMPE
       bsf      ESTADO,C        ;Pone el bit carry en 1 para que no afecte las rotaciones
       rrf      ROTA,1          ;Rota a la derecha el 0 que activa a los displays
       incf     FSR             ;Se incrementa FSR para apuntar al próximo dígito a mostrar
       goto     disp            ;Brinca a la etiqueta DISP

       END                      ;Directiva de fin de programa

*******************************************************************************
```

b) Ensamble el archivo MUX7SEG.ASM y grabe el microcontrolador con el archivo MUX7SEG.HEX.

c) Inserte el módulo con Display de 7 segmentos en la tarjeta principal del sistema de entrenamiento en el conector 4.

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal y encienda el sistema de entrenamiento. Anote sus observaciones.

e) Modifique en el programa MUX7SEG.ASM la rutina de retardo o implemente una nueva rutina para que el retardo sea de 500ms; repita los procedimientos de los incisos b), c) y d).

Nota: puede insertarse el Módulo de Monitores LED en cualquiera de los conectores libres en la línea correspondiente al Puerto A para hacer más evidente el funcionamiento del programa

f) Una vez más modifique el programa pero ahora para que se muestre en los displays el año de su fecha de nacimiento

Actividad 2. CUESTIONARIO.

a) ¿Cuál es la función de los registros INDO y FSR?

b) ¿De qué manera podrían emplearse menos líneas de los puertos del microprocesador para el control de este tipo de displays?.

c) ¿Podría usted hacer que en los displays se despliegue una palabra de 4 letras? Hágalo y describa el procedimiento a seguir para conseguirlo.

d) ¿Qué instrucciones y qué datos tendría que cambiar en el caso de que los displays fueran de cátodo común?

e) ¿De qué manera podrían conectarse y controlarse simultáneamente un teclado matricial y un display de 4 dígitos? Dibuje el diagrama de conexión de estos dos dispositivos con el microcontrolador, el diagrama de flujo del programa.

Práctica Nº 7

MANEJO DE DISPLAY LCD

OBJETIVOS: Al concluir esta práctica el alumno conocerá el funcionamiento y los métodos de control de displays LCD para desplegar mensajes alfanuméricos utilizando el bus de datos a 8 y 4 bits.

DESARROLLO

Actividad 1. Para estudiar el funcionamiento del Módulo LCD con bus a 8 bits, se propone el listado del programa LCD8BIT.ASM Se empleará el sistema de entrenamiento y el Módulo con LCD (figura 5.5).

a) Escriba el listado de programa LCD8BIT.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre y la extensión propuesto y guárdelo.

```
;*******************************************************************************
;Programa LCD8BIT.ASM: este programa hace que un mensaje circule en la pantalla de un modulo LCD
;utilizando el bus de datos a 8 bits

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

       LIST    P= PIC16F84     ;Indica el tipo de microcontrolador a usar
       RADIX   HEX             ;Indica la numeración base

;****************************** ZONA DE ETIQUETAS ***********************************
PC      EQU     02h     ;El registro PC está en la dirección 02 de la memoria de datos
ESTADO  EQU     03h     ;El registro ESTADO esta en la dirección 03 de la memoria de datos
PTOA    EQU     05h     ;05h es la dirección del Puerto A en la memoria de datos
PTOB    EQU     06h     ;06h es la dirección del Puerto B en la memoria de datos
R0C     EQU     0ch     ;Dirección del registro auxiliar R0C en la memoria de datos
R0D     EQU     0eh     ;Dirección del registro auxiliar R0D en la memoria de datos
R13     EQU     13h     ;Dirección del registro auxiliar para retardos en la memoria de datos
Z       EQU     02h     ;Z es el bit 2 del registro ESTADO
C       EQU     00h     ;C es el bit 0 del registro ESTADO
W       EQU     00h     ;Se escribe W cuando el resultado de una operación se almacena en W
R       equ     01h     ;Se escribe R cuando el resultado de una operación se almacena en el
                        ;registro
E       EQU     00h     ;La terminal E del LCD está conectada a la terminal RA0
RS      EQU     01h     ;La terminal RS del LCD está conectada a la terminal RA1

;****************************** PROGRAMA PRINCIPAL **********************************
       ORG     00      ;En el vector de reset se escribe la siguiente instrucción
       goto    inicio  ;Salta a la dirección indicada por la etiqueta "inicio"
       ORG     05h     ;El programa se escribe después del vector de interrupción

retardomovlw   0xFF    ;En esta rutina se genera el retardo necesario
       movwf   R13     ;para la validación de datos o instrucciones
decre  decfsz  R13,R   ;enviadas al módulo de LCD
       goto    decre
       retlw   0       ;Finalizado el retardo retorna de subrutina

;Esta rutina genera las señales de control y entrega el dato correspondiente al modulo utiliza
;interfaz a 8 bits
control bcf    PTOA,RS         ;RS=1 indica que lo que hay en el bus de datos es una
       goto    dato2           ;instrucción de control
dato   bsf     PTOA,RS         ;RS=0 indica que lo que hay en el bus de datos es un caracter
dato2  bsf     PTOA,E          ;E=1: Habilita al LCD
       movwf   PTOB            ;Entrega al bus de datos la instrucción de control o el código
                               ;del caracter
       call    retardo         ;Llama rutina de retardo para validar el dato
       bcf     PTOA,E          ;E=0: deshabilita al LCD
```

```
        call    retardo         ;Llama a subrutina de retardo
        retlw   0               ;Fin de rutina, regresa a programa principal

;La siguiente tabla contiene los caracteres del mensaje que se presenta en la pantalla del LCD,
;los caracteres entre comillas indican al ensamblador que el caracter se expresa en código ASCII

tabla2  addwf   PC,R    ;Suma el contenido de W al registro PC y lo guarda en el registro PC
        retlw   " "     ;Caracter en la posición PC + 0
        retlw   "U"     ;Caracter en la posición PC + 1
        retlw   "N"     ;Caracter en la posición PC + 2
        retlw   "A"     ;PC + 3
        retlw   "M"     ;PC + 4
        retlw   " "     ;PC + 5
        retlw   "M"     ;PC + 6
        retlw   "I"     ;PC + 7
        retlw   "C"     ;PC + 8
        retlw   "R"     ;PC + 9 etc.
        retlw   "O"
        retlw   "C"
        retlw   "O"
        retlw   "N"
        retlw   "T"
        retlw   "R"
        retlw   "O"
        retlw   "L"
        retlw   "A"
        retlw   "D"
        retlw   "O"
        retlw   "R"
        retlw   "E"
        retlw   "S"
        retlw   " "
        retlw   "P"
        retlw   "I"
        retlw   "C"
        retlw   " "
        retlw   0

inicio  movlw   0xFC    ;Se configuran los puertos según se requiere
        tris    PTOA    ;La instrucción TRIS es reconocida para la configuración de puertos aunque
        movlw   00      ;proviene de modelos anteriores de PICs para garantizar la compatibilidad
        tris    PTOB
;Los siguientes datos le indican al LCD la manera en que va a funcionar

begin   movlw   30              ;Inicia display a 8 bits y 1 línea
        call    control
        movlw   07              ;Selecciona el modo del desplazamiento
        call    control
        movlw   0C              ;Activa el display
        call    control

muestra movlw   0               ;Inicia el envío de caracteres
        movwf   R0C             ;al modulo
ciclo   movf    R0C,W           ;hace barrido de la tabla
        call    tabla2          ;Llama a la tabla para cargar en W el código del caracter
        call    dato            ;y pasarlo al LCD durante la subrutina DATO

        movlw   0xAF            ;Se introduce una rutina para retardo entre caracteres
        movwf   R0D             ;para que el usuario pueda leer el mensaje
retal   call    retardo
        call    retardo
        decfsz  R0D,R
        goto    retal

        incf    R0C,R           ;Se incrementa R0C para señalar la siguiente posición del caracter
        movlw   28h             ;del mensaje
        xorwf   R0C,W           ;Pregunta si termino el mensaje para volver
        btfss   ESTADO,Z        ;a empezar:
        goto    ciclo           ;No, salta a la etiqueta CICLO
        goto    muestra         ;Si, Salta a la etiqueta MUESTRA
        END
```

****

b) Ensamble el archivo LCD8BIT.ASM y grabe el microcontrolador con el archivo LCD8BIT.HEX.

c) Inserte el módulo con Display LCD en la tarjeta principal del sistema de entrenamiento en el conector 4.

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal y encienda el sistema de entrenamiento. Anote sus observaciones.

e) Modifique el programa LCD8BIT.ASM para que muestre un mensaje dc bienvenida a un establecimiento comercial, el que usted quiera.

Actividad 2. Con el fin de estudiar el funcionamiento del Módulo LCD con bus a 4 bits, se propone el listado del programa LCD8BIT.ASM Se empleará el sistema de entrenamiento y el Módulo con LCD.

a) Escriba el listado de programa LCD4BIT.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre y la extensión propuesto y guárdelo.

```
;*********************************************************************************
;lcd4bit.asm: este programa hace que un mensaje se repita indefinidamente en un modulo LCD de 2
;líneas con 16 caracteres.

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

       LIST    P=PIC16F84      ;Directiva que indica que el programa esta escrito para el PIC16F84
       RADIX   HEX             ;Directiva que indica que todas las literales están en base
                               ;Hexadecimal

;************************************ ZONA DE ETIQUETAS ***************************************
PC      EQU     02h     ;Contador de programa
ESTADO  EQU     03h     ;Registro de estados y bits de control
PTOA    EQU     05h     ;Puerto A
PTOB    EQU     06h     ;Puerto B
R0C     EQU     0ch     ;Registro auxiliar R0C
R0D     EQU     0eh     ;Registro auxiliar R0D
R0E     EQU     0fh     ;Registro auxiliar R0E
R13     EQU     13h     ;Registro auxiliar R13

;************************************ BITS ESPECIALES *****************************************
Z       EQU     02h     ;Bandera de cero Z en el registro ESTADO
C       EQU     00h     ;Bandera de carry C en el registro ESTADO
W       EQU     00h     ;Para almacenar en W
R       EQU     01h     ;Para almacenar en el mismo registro

;********************************** TERMINALES DEL PUERTO A ***********************************
E       EQU     00h     ;La terminal E del LCD conectada a la terminal RA0
RS      EQU     01h     ;La terminal RS del LCD conectada a la terminal RA1

;********************************** PROGRAMA PRINCIPAL ****************************************
       ORG     00      ;En la dirección del Vector de reset de la memoria de programa se
       goto    inicio  ;escribe la instrucción de salto a la etiqueta INICIO
       ORG     05h     ;El programa se escribe desde la dirección 05 de la memoria de programa

;Retardo para la rutina de control
retardomovlw   0xFF    ;Se carga a W con la literal FFh
       movwf   R13     ;Mueve el contenido de W al registro R13
decre  decfsz  R13,R   ;Decrementa al registro R13 y revisa el estado del bit Z
       goto    decre   ;si Z=0, salta a la etiqueta DECRE
       retlw   0       ;si Z=1, retorna de subrutina RETARDO

;Esta rutina se emplea para borrar el mensaje mostrado en la pantalla del LCD
limpia clrf    R0C             ;Pone en ceros el registro R0C
limpi  movlw   " "             ;Carga W con el código ASCII correspondiente a un espacio en blanco
       call    dato            ;Llama a la subrutina DATO
       incf    R0C,R           ;Incrementa al registro R0C
       movlw   50h             ;Carga a W con la literal hexadecimal 50h
       xorwf   R0C,W           ;Aplica una operación XOR entre W y R0C y el resultado lo almacena
                               ;en W
```

```
        btfss   ESTADO,Z        ;Pregunta si el resultado de la operación es cero (Z=1)
        goto    limpi           ;Si no, salta y ejecuta la rutina desde la etiqueta LIMPI
        retlw   0               ;Si si, retorna de la subrutina

;Esta rutina genera las señales de control y entrega el dato correspondiente al LCD, utiliza
;interfase a 4 bits
control bcf     PTOA,RS         ;Selecciona el registro de control del LCD
        goto    dato2           ;Salta a etiqueta DATO2
dato    bsf     PTOA,RS         ;Selecciona el registro de datos del LCD
dato2   bsf     PTOA,E          ;Habilita al LCD
        movwf   R0E             ;Mueve el contenido de W al registro R0E
        movlw   0x0F            ;Carga en W la literal 0Fh
        andwf   PTOB,R          ;Aplica una operación AND entre W y el Puerto B
        movf    R0E,W           ;Mueve el contenido del registro R0e a W
        andlw   0xF0            ;Aplica una operación AND entre la literal F0h y W
        iorwf   PTOB,R          ;Aplica una operación OR entre W y el Puerto B
        call    retardo         ;Llama a rutina de retardo
        bcf     PTOA,E          ;Deshabilita al LCD
        call    retardo         ;Llama a rutina de retardo
        bsf     PTOA,E          ;Habilita al LCD
        movlw   0x0F            ;Carga a W con 0Fh
        andwf   PTOB,R          ;AND entre W y Puerto B
        swapf   R0E,W           ;Intercambia los cuatro bits menos significativos con los cuatro
                                ;bits más significativos del registro R0E
        andlw   0xF0            ;AND entre W y la literal F0h
        iorwf   PTOB,R          ;OR entre Puerto B y W
        call    retardo         ;Llama a subrutina de retardo
        bcf     PTOA,E          ;Deshabilita al LCD
        call    retardo         ;Llama a subrutina de retardo
        retlw   00              ;Retorna de subrutina de control

tabla   addwf   PC,R            ;Mensaje que se muestra fijo en pantalla del LCD
        retlw   " "             ;Mensaje de la primera línea
        retlw   "E"
        retlw   "n"
        retlw   "t"
        retlw   "r"
        retlw   "e"
        retlw   "n"
        retlw   "a"
        retlw   "d"
        retlw   "o"
        retlw   "r"
        retlw   " "
        retlw   "P"
        retlw   "I"
        retlw   "C"
        retlw   " "
        retlw   " "

        retlw   " "             ;Mensaje de la segunda línea
        retlw   " "
        retlw   " "
        retlw   "U"
        retlw   "N"
        retlw   "A"
        retlw   "M"
        retlw   " "
        retlw   "A"
        retlw   "R"
        retlw   "A"
        retlw   "G"
        retlw   "O"
        retlw   "N"
        retlw   " "
        retlw   " "
        retlw   00

inicio  movlw   0xFC            ;Se programan los puertos
        tris    PTOA            ;según el circuito
        movlw   0x00
        tris    PTOB

begin   movlw   0x02            ;Inicia display LCD a 4 bits
        call    control
        movlw   0x28            ;Display LCD a 4 bits y 2 lineas
```

```
        call    control
        movlw   0x0c            ;Activa el display LCD
        call    control
        movlw   0x06            ;Hace que el mensaje permanezca fijo
        call    control
blank   call    limpia          ;Borra pantalla del display LCD
muestra clrf    R0C             ;Inicia el contador de caracteres
ciclo   movf    R0C,W           ;Hace barrido de la tabla
        call    tabla
        call    dato
        movlw   0xFF            ;Retardo entre caracteres
        movwf   R0D
retal   call    retardo
        decfsz  R0D,R
        goto    retal
        incf    R0C,R           ;Sigue con la tabla
        movlw   0x11
        subwf   R0C,W           ;Pregunta si esta mostrando el mensaje de la
        btfss   ESTADO,C        ;segunda línea
        goto    ciclo
        movlw   0x11            ;Pregunta si es la primera vez que entra
        xorwf   R0C,W           ;a la segunda línea para ir a iniciar
        btfss   ESTADO,Z        ;El puntero de la RAM del modulo LCD
        goto    line2
linea2  movlw   0xC0            ;Ubica el puntero de la RAM del modulo LCD
        call    control         ;en la segunda línea
line2   movlw   0x21            ;Pregunta si termino la segunda línea
        xorwf   R0C,W           ;para ir a iniciar de nuevo el mensaje o
        btfss   ESTADO,Z        ;para continuar en la segunda parte del mensaje
        goto    ciclo
        movlw   0x80            ;Ubica el puntero de RAM en la primera fila
        call    control
        goto    blank           ;Va a iniciar rutina de limpieza de pantalla

        END
```

****

b) Ensamble el archivo LCD4BIT.ASM y grabe el microcontrolador con el archivo LCD4BIT.HEX.

c) Conmute los interruptores del DIP-Switch del módulo con LCD a la posición OFF.

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal y encienda el sistema de entrenamiento. Anote sus observaciones.

Actividad 3 CUESTIONARIO

a) ¿De cuántos caracteres puede ser un mensaje en el programa LCD8BIT.ASM?

b) ¿Qué rutina se debe modificar para mostrar un mensaje de 60 caracteres?

c) ¿Cómo funciona la instrucción *swapf*?

d) Investigue la tabla de caracteres admitidos por el módulo LCD.

e) ¿Cuál es el principio de funcionamiento de un display LCD?

f) Investigue la denominación y funciones de las terminales, los tiempos de operación y el diagrama a bloques de un display LCD.

Práctica Nº 8

MANEJO DE MOTORES PASO A PASO

OBJETIVO: Al final de la práctica el alumno conocerá y empleará rutinas para el manejo de motores paso a paso (PAP).

DESARROLLO.

Actividad 1.

Para cumplir con el objetivo se propone el listado del programa MOTORPAP.ASM. Se empleará el sistema de entrenamiento, el módulo de monitores LED y el módulo con motor PAP (figura 5.6).

a) Escriba el listado de programa MOTORPAP.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre propuesto, la extensión .ASM y guárdelo.

```
;*******************************************************************************
;Programa con empleo de rutinas para el manejo de motores PAP en un modo de operación del motor
;de pasos completos

            ;configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=16F84         ;Define el microcontrolador a usar
        RADIX   HEX             ;Especifica la numeración base

        PUERTOA EQU     05      ;Asigna al registro PUERTOA la dirección 05 de la memoria de datos
        PUERTOB EQU     06      ;Asigna al registro PUERTOB la dirección 06 de la memoria de datos
        ESTADO  EQU     03      ;Asigna al registro ESTADO la diercción 03 de la memoria de datos
        PDel0   EQU     0C      ;Asigna al registro auxiliar PDel0 la dirección 0C de la memoria RAM
        PDel1   EQU     0D      ;Asigna al registro auxiliar PDel1 la dirección 0D de la memoria RAM

        ORG     00              ;Indica al ensamblador la dirección de la primera
        goto    INICIO          ;instrucción del programa (00h)

INICIO  bsf     ESTADO,5        ;Selecciona la pagina 1 de la memoria de datos
        movlw   00h             ;Rutina de configuración
        movwf   PUERTOB         ;del Puerto B como salidas
        movlw   0Fh             ;Rutina de configuración
        movwf   PUERTOA         ;del Puerto A como entradas
        bcf     ESTADO,5        ;Selecciona la pagina 0 de la memoria de datos
        clrf    PUERTOA         ;Pone a ceros el Puerto A
        clrf    PUERTOB         ;Pone a ceros el Puerto B

EXPLORA clrf    PUERTOB         ;Pone a ceros el Puerto B
        btfss   PUERTOA,0       ;Explora el estado del bit 0 del Puerto A
        call    RIGHT           ;si es cero llama a la subrutina RIGHT
        btfss   PUERTOA,1       ;Explora el estado del bit 1 del Puerto A
        call    LEFT            ;si es cero llama a la subrutina LEFT
        goto    EXPLORA         ;Regresa a la subrutina EXPLORA

RIGHT   movlw   B'00001100'     ;Se carga este valor binario para operar en motor
        movwf   PUERTOB         ;a pasos completos
        call    RETARDO         ;Tiempo de duración de los pulsos
ROTA1   bcf     ESTADO,0        ;Pone a cero el bit Carry del registro ESTADO
        rrf     PUERTOB         ;Se rota a la derecha el par de bits
        call    RETARDO         ;Tiempo de duración de los pulsos
        btfsc   PUERTOA,0       ;Comprueba si el boton continua pulsado si no
        return                  ;retorna de la subrutina
        btfss   PUERTOB,0       ;Comprueba si el bit 0 de Puerto B es uno
        goto    ROTA1           ;si no, regresa a la subrutina ROTA1
        movlw   B'00001001'     ;Se carga este valor binario en el Puerto B para
```

```
        movwf   PUERTOB         ;mantener la secuencia de los dos bits consecutivos
        call    RETARDO         ;Tiempo de duración de los pulsos
        clrf    PUERTOB         ;Pone a ceros el Puerto B
        goto    RIGHT           ;Terminado un ciclo regresa a RIGHT

LEFT    movlw   B'00000011'     ;Se carga este valor binario para operar en motor
        movwf   PUERTOB         ;pasos completos
        call    RETARDO         ;Tiempo de duración de los pulsos
ROTA2   bcf     ESTADO,0        ;Pone a cero el bit Carry del registro ESTADO
        rlf     PUERTOB         ;Se rota a la izquierda el par de bits
        call    RETARDO         ;Tiempo de duración de los pulsos
        btfsc   PUERTOA,1       ;Comprueba si el boton continua pulsado si no
        return                  ;retorna de la subrutina
        btfss   PUERTOB,3       ;Comprueba si el bit 3 de Puerto B es uno
        goto    ROTA2           ;si no, regresa a la subrutina ROTA2
        movlw   B'00001001'     ;Se carga este valor binario en el Puerto B para
        movwf   PUERTOB         ;mantener la secuencia de los dos bits consecutivos
        call    RETARDO         ;Tiempo de duración de los pulsos
        clrf    PUERTOB         ;Pone a ceros el Puerto B
        goto    LEFT            ;Terminado un ciclo regresa a LEFT

RETARDO movlw   .21             ;Configura el numero de repeticiones
        movwf   PDel0           ;para el registro Pdel0
LOOP1   movlw   .237            ;Configura el numero de repeticiones
        movwf   PDel1           ;para el registro Pdel1
LOOP2   clrwdt                  ;Pone a ceros el temporizador WDT
        decfsz  PDel1,1         ;Decrementa el registro Pdel1 y y si ha llegado a cero
                                ;se brinca una instruccion
        goto    LOOP2           ;Si no es cero regresa a la etiqueta LOOP2
        decfsz  PDel0,1         ;Decrementa el registro Pdel0 y y si ha llegado a cero
                                ;se brinca una instrucción
        goto    LOOP1           ;si no es cero regresa a la etiqueta LOOP2
        return                  ;Si ambos decrementos se han completado regresa
        END                     ;Directiva de fin de programa

*******************************************************************************
```

b) Ensamble el archivo MOTORPAP.ASM y grabe el microcontrolador con el archivo MOTORPAP.HEX.

c) Inserte el módulo con motor PAP en la tarjeta principal del sistema de entrenamiento en el conector 1 y el módulo con Push-Button en cualquier otro conector disponible en la línea correspondiente al Puerto A

d) Inserte el microcontrolador en el zócalo correspondiente de la tarjeta principal y encienda el sistema de entrenamiento.

e) Oprima alternativamente los botones S1 y S2 y anote sus observaciones.

f) Modifique el programa MOTORPAP.ASM para incrementar el retardo y se visualice en el módulo de monitores LED el estado y secuencia de bits aplicados al motor.

Actividad 2. Cuestionario

a) ¿Cuántos modos de operación tiene un motor PAP?

b) Escriba un programa para operar el motor en el modo de ½ paso.

c) Escriba un programa para controlar independientemente a través de los cuatro botones del módulo con Push-Button dos motores PAP en cualquiera de sus modos de operación.

d) Escriba por lo menos cinco aplicaciones comerciales de los motores PAP.

Práctica Nº 9

COMUNICACIÓN SERIAL RS-232.

OBJETIVO: Que el alumno emplee técnicas de comunicación serial RS-232 entre microcontroladores PIC y PC.

DESARROLLO

Actividad 1

Para cumplir con el objetivo se propone el programa ENVIA.ASM, el cual permite el flujo de datos de la PC hacia el microcontrolador. Se empleará el sistema de entrenamiento, el módulo RS-232 (figura 5.12), el módulo de monitores LED y una extensión para puerto serial.

a) Escriba el listado de programa ENVIA.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre propuesto, la extensión .ASM y guárdelo.

```
;*******************************************************************************
;Programa ENVIA.ASM: este programa envía datos de la PC al microcontrolador
;vía RS-232. Velocidad: 1200, Datos: 8 Bits, Sin paridad, Bit de paro: 1

            ;Configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=16F84         ;Especifica el microcontrolador a emplear
        RADIX   HEX             ;Especifica la numeración base

ESTADO  EQU     03h     ;Registaro ESTADO en la dirección 03H
PUERTOA EQU     05h     ;Puerto A en la dirección 05H
PUERTOB EQU     06h     ;Puerto B en la dirección 06H
TRISA   EQU     85h     ;Registro TRIS A en 85H
TRISB   EQU     86h     ;Registro TRIS B en 86H
R0D     EQU     0Dh
R0E     EQU     0Eh
CONTA   EQU     10h
RECEP   EQU     11h
Z       EQU     02h
C       EQU     00h
RP0     EQU     05h
Z       EQU     02h
C       EQU     00h
W       EQU     00h
R       EQU     01h
RX      EQU     06h

        ORG     00              ;Dirección del vector RESET
        GOTO    Inicio          ;Primera instrucción a ejecutar
        ORG     05

retardo movlw   .249            ;Carga el valor en el registro W
        goto    startup         ;brinca a la etiqueta STARTUP
delay1  movlw   .166            ;Carga el valor en el registro W
startup movwf   R0E             ;Carga el valor .249 en el registro R0E
redo    nop                     ;1 ciclo de retardo
        nop                     ;1 ciclo de retardo
        decfsz  R0E             ;Decrementa el registro R0E y se brinca un instrucción si es cero
        goto    redo            ;si no es cero brinca a la etiqueta REDO
        retlw   0               ;Retorna de subrutina y carga W con 0

recibir nop                     ;1 ciclo de retardo
```

```
        clrf    RECEP           ;pone a ceros el registro RECEP
        btfsc   PUERTOB,RX      ;Verifica el estado del bit RX del Puerto B
        goto    recibir         ;si es 1 brinca a la etiqueta RECIBIR
        call    retardo         ;si es cero llama a la subrutina RETARDO
rcvr    movlw   8               ;Carga el valor en W
        movwf   CONTA           ;Pasa el valor de W al registro CONTA
rnext   bcf     ESTADO,C        ;Pone a cero el bit C del registro ESTADO
        btfsc   PUERTOB,RX      ;Verifica el estado del bit RX del Puerto B
        bsf     ESTADO,C        ;Pone a uno el bit C del registro ESTADO
        rrf     RECEP           ;Rota a la derecha el valor del registro RECEP
        call    delay1          ;Llama a la subrutina DELAY1
        decfsz  CONTA           ;Decrementa el registro CONTA y se brinca un instrucción si es cero
        goto    rnext           ;si es 1 brinca a la etiqueta RNEXT
        retlw   0               ;Retorna de subrutina y carga W con 0

Inicio  bsf     ESTADO,RP0      ;Pone a uno el bit RP0 del registro ESTADO
        movlw   00h             ;Carga el valor en el registro W
        movwf   TRISA           ;Configura el Puerto A como salidas
        movlw   0xFF            ;Carga el valor en el registro W
        movwf   TRISB           ;Configura el Puerto B como entradas
        bcf     ESTADO,RP0      ;Pone a cero el bit RP0 del registro ESTADO
        clrf    RECEP           ;Pone a ceros el registro RECEP
        clrf    PUERTOA         ;Pone a ceros el Puerto A

ciclo   call    recibir         ;Llama a la subrutina RECIBIR
        movlw   30h             ;Carga el valor en el registro W
        subwf   RECEP,W         ;Resta el valor de W al registro RECEP y lo deja en W
        movwf   PUERTOA         ;Mueve el valor de W al Puerto A
        goto    ciclo           ;Brinca a la etiqueta CICLO

        END                     ;Directiva de fin de programa
```

****

b) Ensamble el archivo ENVIA.ASM y grabe el microcontrolador con el archivo ENVIA.HEX

c) Inserte el módulo RS-232 en la tarjeta principal del sistema de entrenamiento en cualquier conector, en la línea correspondiente al Puerto B; y el módulo de monitores LED en cualquier otro conector disponible, en la línea correspondiente al Puerto A.

d) Conecte el módulo RS-232 a través de la extensión con el puerto serial de la PC, ya sea COM1 o COM2 según se tenga la disponibilidad.

Para evaluar el programa se requiere de una aplicación en la PC que se encargue de configurar el puerto con los valores adecuados, de realizar el conteo de las veces que se pulsa una tecla y de mostrar ese número en la pantalla mientras lo envía por el puerto serial hacia el microcontrolador. En este caso utilizamos un programa en lenguaje C, se escogió este lenguaje debido a que es el más utilizado en aplicaciones electrónicas y permite configurar fácilmente los puertos. A continuación se muestra el listado del código fuente del programa empleado.

****

```
/* LA COMPUTADORA ENVIA DATOS SERIALES AL PIC */
#include <conio.h>
#include <stdio.h>
#include <bios.h>

int COM1,COM2,Puerto;                   /*definición de variables*/
int j,envio,configuracion;
int contador;
char tecla;
void main(void)
{
```

```
 clrscr();                          /*limpiar pantalla*/
 COM1=0;                            /*constantes de los puertos del PC*/
 COM2=1;
 Puerto=COM1;                       /*Aquí se debe indicar si es COM1 o COM2*/

 configuracion=0x83;          /*conf.puerto: 1200,8,N,1*/
 bioscom(0,configuracion,Puerto);          /*inicializa el COM del PC*/

 gotoxy(12,2);
 printf("Sistema de Entrenamiento con Microcontroladores PIC");

 gotoxy(10,5);
 printf("Envio de datos seriales hacia el microcontrolador - COM1");

 gotoxy(10,7);
 printf("  Enter = Incremento del contador     Escape = Salir");

 gotoxy(24,10);
 printf("El dato del contador es:");
 contador=0;

do{                                 /*ciclo de lectura de medida*/
    tecla=getch();
    contador++;
    if(contador==10) contador=0;
    gotoxy(34,12);
    printf("%d",contador);                  /*Obtiene tecla oprimida*/
    envio=bioscom(1,contador+0x30,Puerto);   /*envía caracter al micro*/
  }while(tecla!=27);                /*Hasta que se oprima ESC*/

 while(!kbhit());
 clrscr();
 printf(" HSF VGA ");

}
```

****

Una vez capturado, se compila y se crea el archivo ejecutable, que en nuestro caso son dos ENVIACOM1.EXE y ENVIACOM2.EXE, ya que cada uno direcciona los datos al puerto serial correspondiente.

e) Ejecute la aplicación ENVIACOM1.EXE o bien ENVIACOM2.EXE dependiendo del puerto al que se haya conectado el módulo RS-232.

f) Encienda el sistema, pulse consecutiva y pausadamente cualquier tecla de la PC, anote sus observaciones.

g) Modifique el programa ENVIA.ASM para emplear el Módulo con display de 7 segmentos en lugar del Módulo de monitores LED.

Actividad 2

Para complementar el objetivo se propone ahora el listado del programa RECIBE.ASM el cual permite el flujo de datos del microcontrolador hacia la PC. Se empleará el sistema de entrenamiento, el módulo RS-232, el módulo con Push-Button y una extensión para puerto serial.

a) Escriba el listado de programa RECIBE.ASM en el editor de MPLAB o en algún otro procesador de texto. Al archivo asígnele el nombre propuesto, la extensión .ASM y guárdelo.

```
;*********************************************************************************
;Programa RECIBE.ASM: este programa envía datos del microcontrolador a la PC vía RS-232.
;Velocidad: 1200, Datos: 8 Bits, Sin paridad, Bit de paro: 1

            ;Configuración para las casillas de verificación en el programa grabador:
            ;*******  WDTE = OFF , CODE PROTECT = OFF, PWRTE = OFF, OSC = XT  *******

        LIST    P=16F84         ;Especifica el microcontrolador a emplear
        RADIX   HEX             ;Especifica la numeración base

ESTADO  EQU     03h     ;Registro ESTADO en la dirección 03H
PUERTOA EQU     05h     ;Puerto A en la dirección 05H
PUERTOB EQU     06h     ;Puerto B en la dirección 06H
TRISA   EQU     85h     ;Registro TRIS A en 85H
TRISB   EQU     86h     ;Registro TRIS B en 86H
TRANS   EQU     0Ch
R0D     EQU     0Dh
R0E     EQU     0Eh
UNIDAD  EQU     10h
DECENA  EQU     11h
CENTENA EQU     12h
R14     EQU     14h
R1B     EQU     1Bh
LOOPS   EQU     13h
LOOPS2  EQU     14h
CONTA   EQU     15h
Z       EQU     02h
RP0     EQU     05h
Z       EQU     02h
C       EQU     00h
W       EQU     00h
R       EQU     01h
TX      EQU     07h

        ORG     00              ;Dirección del vector RESET
        GOTO    Inicio          ;Primera instrucción a ejecutar

delay1  movlw   .166            ;Carga el valor en el registro W
startup movwf   R0E             ;Carga el valor decimal 166 en el registro R0E
redo    nop                     ;1 ciclo de retardo
        nop                     ;1 ciclo de retardo
        decfsz  R0E             ;Decrementa el registro R0E y se brinca un instrucción si es cero
        goto    redo            ;si no es cero brinca a la etiqueta REDO
        retlw   0               ;Retorna de subrutina y carga W con 0

retardo movlw   d'100'          ;Carga el valor en el registro W
        movwf   LOOPS           ;Carga el valor de W en el registro LOOPS
top2    movlw   d'110'          ;Carga el valor en el registro W
        movwf   LOOPS2          ;Carga el valor de W en el registro LOOPS2
top     nop                     ;1 ciclo de retardo
        nop                     ;1 ciclo de retardo
        nop                     ;1 ciclo de retardo
        nop                     ;1 ciclo de retardo
        nop                     ;1 ciclo de retardo
        nop                     ;1 ciclo de retardo
        decfsz  LOOPS2          ;Decrementa el registro LOOPS2 y se brinca un instrucción si es cero
        goto    top             ;Brinca a la etiqueta TOP
        decfsz  LOOPS           ;Decrementa el registro LOOPS2 y se brinca un instrucción si es cero
        goto    top2            ;Brinca a la etiqueta TOP
        retlw   0               ;Retorna de subrutina y carga W con 0

;*********************************RUTINA PARA ENVIAR DATO*********************************
enviar  movwf   TRANS           ;Carga el valor de W al registro TRANS
```

```
xmrt    movlw   8               ;Carga el valor en W
        movwf   R0D             ;Carga el valor de W en el registro R0D
        bcf     PUERTOB,TX      ;Pone a 0 el bit TX del Puerto B
        call    delay1          ;Llama a la subrutina DELAY1
xnext   bcf     PUERTOB,TX      ;Pone a 0 el bit TX del Puerto B
        bcf     ESTADO,C        ;Pone a 0 el bit C del registro ESTADO
        rrf     TRANS           ;Rota a la derecha el valor del registro TRANS
        btfsc   ESTADO,C        ;Verifica el estado del bit C del registro ESTADO
        bsf     PUERTOB,TX      ;si es 1 pone a 1 el bit TX del Puerto B
        call    delay1          ;si es 0 llama a la subrutina DELAY1
        decfsz  R0D             ;Decrementa el registro R0D y se brinca un instrucción si es cero
        goto    xnext           ;si es 1 brinca a la etiqueta XNEXT
        bsf     PUERTOB,TX      ;Pone a 0 el bit TX del Puerto B
        call    delay1          ;Llama a la subrutina DELAY1
        retlw   0               ;Retorna de subrutina y carga W con 0

Inicio  bsf     ESTADO,RP0      ;Rutina
        movlw   00h             ;para
        movwf   PUERTOA         ;configuración
        movlw   07Fh            ;de
        movwf   PUERTOB         ;Puertos
        bcf     ESTADO,RP0      ;del Microcontrolador
        bsf     PUERTOB,TX      ;Pone a 1 el bit TX del Puerto B
        clrf    CONTA           ;Pone a ceros el registro CONTA
ciclo   movf    CONTA,W         ;Mueve el registro CONTA a W
        movwf   PUERTOA         ;Carga el valor de W al Puerto A
        addlw   30h             ;Operación SUMA de W con 30H
        call    enviar          ;Llama a la subrutina ENVIAR
        call    retardo         ;Llama a la subrutina RETARDO
pulsa   btfsc   PUERTOB,0       ;Verifica el estado del bit 0 del Puerto B
        goto    pulsa           ;Brinca a la etiqueta PULSA
        call    retardo         ;Llama a la subrutina RETARDO
        btfsc   PUERTOB,0       ;Verifica el estado del bit 0 del Puerto B
        goto    pulsa           ;Brinca a la etiqueta PULSA
        incf    CONTA           ;Incrementa el registro CONTA
        movf    CONTA,W         ;Mueve el registro CONTA a W
        xorlw   0Ah             ;Operación XOR de W con el valor 0AH
        btfsc   ESTADO,Z        ;Verifica el estado del bit Z del registro ESTADO
        goto    Inicio          ;Brinca a la etiqueta INICIO
        goto    ciclo           ;Brinca a la etiqueta CICLO
        END                     ;Directiva de fin de programa
```

****

b) Ensamble el archivo RECIBE.ASM y grabe el microcontrolador con el archivo RECIBE.HEX

c) Inserte el módulo RS-232 en la tarjeta principal del sistema de entrenamiento en cualquier conector, en la línea correspondiente al Puerto B; y el módulo con Push-Button en cualquier otro conector disponible, en la línea correspondiente al Puerto A.

d) Conecte el módulo RS-232 a través de la extensión con el puerto serial de la PC, ya sea COM1 o COM2 según se tenga la disponibilidad.

Al igual que en la actividad 1 se requiere de un programa en la PC, del cual el listado se muestra a continuación:

****

/* LA COMPUTADORA RECIBE LOS DATOS SERIALES ENVIADOS POR EL PIC */

```
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>
#include <bios.h>

 int puerto,COM1,COM2;
```

```
 int k,j,dato;                                  /*definición de variables*/
 int config;
/* int COM1,COM2;*/
 char lectura[1];
 char dato1[2];

char leer()
{
   do{
         dato=bioscom(2,0x83,puerto);          /*leer dato recibido*/
    } while (((dato<31)|(dato>127))&(!kbhit()));
   return(dato);
}

void main(void)
{
 COM1=0;
 COM2=1;
 puerto=COM1;                                   /* Aquí se debe indicar si es COM1 o COM2*/

 clrscr();                                      /*limpiar pantalla*/
 config=0x83;  /*configurar puerto: 1200 baudios,dato de 8 bits,
 no paridad, 1 bit de parada*/
 bioscom(0,config,puerto);                      /*configuracion de los puertos*/

 gotoxy(13,4);
 printf("Sistema de Entrenamiento con Microcontroladores PIC");
 gotoxy(8,6);
 printf("La computadora recibe los datos enviados por el micro - COM1");
 gotoxy(29,8);
 printf("Escape = Salir");
 gotoxy(23,10);
 printf("El dato del contador es:");

do{
   if(!kbhit()) dato1[0]=leer();
   if(!kbhit())
   {
     gotoxy(40,12);
     printf("%1s ",dato1);
   }
 }while(!kbhit());
 clrscr();
 printf(" HSF AVG ");
}

*****************************************************************************
```

Los ejecutables obtenidos en nuestro caso son RECIBECOM1.EXE y RECIBECOM2.EXE, los cuales direccionan los datos al puerto COM1 y COM2 respectivamente.

e) Ejecute la aplicación RECIBECOM1.EXE o bien RECIBECOM2.EXE dependiendo del puerto al que se haya conectado el módulo RS-232.

f) Encienda el sistema, pulse consecutiva y pausadamente el botón S1 del módulo con Push-Button, anote sus observaciones.

Actividad 3. Cuestionario

a) Describa brevemente las características del protocolo RS-232.

b) Explique someramente la operación del circuito integrado MAX-232.

c) ¿Qué otros tipos de protocolos de comunicación pueden tener los microcontroladores PIC?

d) ¿Qué son las redes neuronales?

e) ¿Se puede implementar una pequeña red neuronal con microcontroladores PIC?, Si o no y ¿por qué?

f) ¿De cuanto tiempo debe ser el retardo para el envío de datos del microcontrolador hacia la PC, considerando una velocidad de envío de 1200bps?

CONCLUSIONES

El desarrollo de este trabajo nos permitió conocer y aplicar las caracteristicas que proporcionan los microcontroladores PIC, enfocando estas propiedades a la creación del sistema de entrenamiento como una herramienta de apoyo para el sector educativo. Sabemos que es difícil que se cambie el plande estudios basándose únicamente en nuestra propuesta, pero creemos fiememente que este tipo de dispositivos podría incluirse dentro del plan de estudios del área eléctrica-electrónica.

El sistema de entrenamiento que proponemos fue diseñado con la idea de que se empleara sistemáticamente en el Laboratorio de Electrónica de la Universidad Nacional Autónoma de México Campus Aragón, teniendo para esto algunas consideraciones en el manejo y operación de dicho laboratorio. Cumpliendo con el objetivo de presentar una propuesta para la U.N.A.M. en el aspecto de proporcionar al estudiantado, herramientas con tecnología de punta que permitan elevar el nivel académico de los ingenieros, dejamos a consideración de la Dirección de Ingeniería la posibilidad de emplear dicha herramienta

Al estudiar al PIC16F84 pudimos conocer algunas características que son comunes a otras familias de este tipo de microcontroladores, características que los hacen sobresalir por sobre otros microcontroladores de 8 bits; entre las más importantes podemos mencionar su arquitectura Harvard que permite un acceso simultaneo a la memoria de programa y a la memoria de datos, un reducido número de instrucciones (arquitectura RISC) y, todo lo anterior, en conjunción con una estructura segmentada de las instrucciones, le permite al PIC reducir el tiempo de ejecución y una compactación del código de las instrucciones del programa almacenado en su memoria. Además, al comprender el manejo de registros, recursos internos y juego de instrucciones de este microcontrolador, hace más fácil aprender el funcionamiento de cualquier otro modelo de PIC.

Si bien el sistema de entrenamiento está orientado a un modelo de microcontrolador en específico, tiene posibilidades de expandir sus alcances, ya que tiene compatibilidad con el PIC16C71, el cual, además de guardar una correspondencia de terminal a terminal y otras características similares a las del PIC16F84 cuenta con convertidor A/D de 4 canales con resolución de 8 bits destinado al tratamiento de señales analógicas. Esto además de permitir el diseño de nuevos módulos, solventa la realización de proyectos y prácticas que requieran el manejo de señales analógicas, con lo que se aumentan las prestaciones del sistema. No obstante, la compatuibilidad con un solo modelo de microcontrolador de características superiores, limita el desarrollo de prácticas en las que se requieran mayores recursos y que pudieran estar disponibles en otra gama de la familia de microcontroladores PIC.

No podemos negar que el haber estado ligados al trabajo de laboratorio primero como alumnos y como instructores de laboratorio en la E.N.E.P. Aragón y actualmente como asesores y administradores de laboratorios de ingeniería en una institución educativa privada y, por consiguiente, al estar contínuamente en contacto con equipos didácticos, fue precisamente lo que nos motivó a realizar esta investigación y darle al sistema de entrenamiento la versatilidad y comodidad que proporciona el manejo de módulos independientes.

BIBLIOGRAFÍA

- Diseño digital
  M. Morris Mano
  Editorial Prentice – Hall Hispanoamericana S.A. 1987.

- Sistemas electrónicos digitales
  Enrique Mandado
  Editorial Marcombo S.A. España 1991, Alfaomega S.A. de C.V. México 1992.

- Electrónica teoría de circuitos
  Robert Boylestad, Louis Nashelsky
  Editorial Prentice – Hall Hispanoamericana S.A. 1994.

- Microcontroladores PIC la solución en un chip
  Eugenio Martín Cuenca, José Mª Angulo Usategui, Ignacio Angulo Martínez
  Editorial Paraninfo. España 1998.

- Microcontroladores PIC
  Christian Tavernier
  Editorial Paraninfo. España 1997.

- Microcontroladores PIC diseño práctico de aplicaciones
  José Mª Angulo Usategui, Ignacio Angulo Martínez
  Editorial McGraw-Hill, Interamericana de España S.A.V. España 1999.

- Curso avanzado de microcontroladores PIC
  Edison Duque C.
  Compañía Editorial Tecnológica CEKIT. Colombia 1998.

- MPLAB® IDE, SIMULATOR, EDITOR User's Guide
  Microchip Technology Incorporated. EE. UU. 1998.

- MPASM® User's Guide with MPLINK and MPLIB
  Microchip Technology Incorporated. EE. UU. 1998.

- Revista Electrónica & Computadores, Año1 Nº 1
  Publicaciones CEKIT S.A. Colombia 1995.

- Revista Electrónica & Computadores, Año1 Nº 2
  Publicaciones CEKIT S.A. Colombia 1995.

- Curso práctico de circuitos digitales y microprocesadores
  Compañía Editorial Tecnológica CEKIT. Colombia 1995

APÉNDICE A

XIAMEN OCULAR LCD DEVICES CO.,LTD

SPECIFICATIONS OF LCD MODULE

Part number : **GDM1602A series**
Date: **April 2, 1998**

SPECIFICATIONS OF LCD MODULE GDM1602A

ABSOLUTE MAXIMUM RATINGS

ITEM	SYMBOL	MIN	MAX	UNIT
POWER VOLTAGE	V_{DD} -V_{SS}	0	7.0	V
INPUT VOLTAGE	V_{IN}	V_{SS}	V_{DD}	
OPERATING TEMPERATURE RANGE	T_{OP}	0	+50	°C
STORAGE TEMPERATURE RANGE	T_{ST}	-20	+60	

*WIDE TEMPERATURE RANGE IS AVAILABLE

(OPERATING/STORAGE TEMPERATURE AS WIDE AS -20□+70/-30□+80□)□

OPTICAL CHARACTERISTICS

FOR TN TYPE DISPLAY MODULE (T_A=25□, V_{DD}=5.0V±0.25V)

ITEM	SYMBOL	CONDITION	MIN.	TYP.	MAX.	UNIT
VIEWING ANGLE	θ	C_R≥4	-25	-	-	DEG
	Φ		-30	-	30	
CONTRAST RATIO	C_R		-	2	-	-
RESPONSE TIME(RISE)	T_R	-	-	120	150	MS
RESPONSE TIME(FALL)	T_R	-	-	120	150	MS

FOR STN TYPE DISPLAY MODULE (T_A=25□, V_{DD}=5.0V±0.25V)

ITEM	SYMBOL	CONDITION	MIN.	TYP.	MAX.	UNIT
VIEWING ANGLE	θ	C_R≥2	-60	-	35	DEG
	Φ		-40	-	40	
CONTRAST RATIO	C_R		-	6	-	-
RESPONSE TIME(RISE)	T_R	-	-	150	250	MS
RESPONSE TIME(FALL)	T_R	-	-	150	250	MS

XIAMEN OCULAR LCD DEVICES CO.,LTD. A2

SOUTH 5/F, GUANGXIA BUILDING, TORCH HIGH-TECH DEVELOPMENT AREA,
XIAMEN 361006.P.R.CHINA TEL: 86-592-5715579 FAX: 86-592-6026021

ELECTRICAL CHARACTERISTICS

DC CHARACTERISTICS

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYPE	MAX.	UNIT
SUPPLY VOLTAGE FOR LCD	V_{DD}-V_O	T_A=25□	—	4.6	—	V
INPUT VOLTAGE	V_{DD}		4.7	—	5.5	V
SUPPLY CURRENT	I_{DD}	V_{DD}=5.0V;T_A=25□	—	1.5	2.5	MA
INPUT LEAKAGE CURRENT	I_{LKG}		—	—	1.0	μA
"H" LEVEL INPUT VOLTAGE	V_{IH}		2.2	—	V_{DD}	V
"L" LEVEL INPUT VOLTAGE	V_{IL}	TWICE INITIAL VALUE OR LESS	0	—	0.6	V
"H" LEVEL OUTPUT VOLTAGE	V_{OH}	LOH= -0.25MA	2.4	—	—	V
"L" LEVEL OUTPUT VOLTAGE	V_{OL}	LOL=1.6MA	—	—	0.4	V
BACKLIGHT SUPPLY POWER	V_F		—	4.2	4.5	V

AC CHARACTERISTICS

READ CYCLE (V_{DD}=5.0V+10%,V_{SS}=OV, T_A=25□)

PARAMETER	SYMBOL	TEST PIN	MIN.	TYPE	MAX.	UNIT
ENABLE CYCLE TIME	T_C	E	500	-	-	NS
ENABLE PULSE WIDTH	T_W		300	-	-	
ENABLE RISE/FALL TIME	T_R, T_F		-	-	25	
RS,R/W SETUP TIME	T_{SU}	RS; R/W	100	-	-	
RS.R/W ADDRESS HOLD TIME	T_H		10	-	-	
READ DATA OUTPUT DELAY	T_D	DB0-DB7	60	-	190	
READ DATA HOLD TIME	T_{DH}		20	-	-	

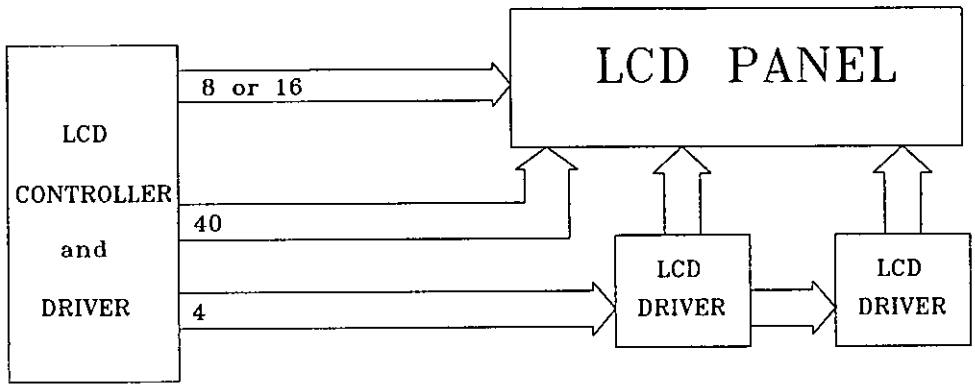
XIAMEN OCULAR LCD DEVICES CO.,LTD. A3

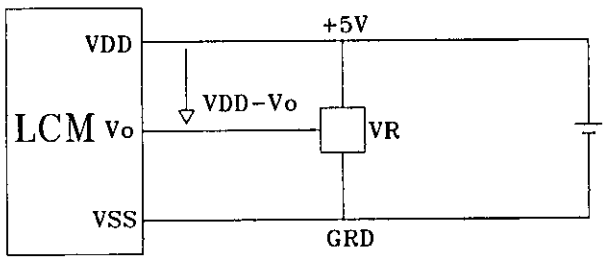
SOUTH 5/F. GUANGXIA BUILDING. TORCH HIGH-TECH DEVELOPMENT AREA.
XIAMEN 361006.P.R.CHINA TEL: 86-592-5715579 FAX: 86-592-6026021

WRITE CYCLE

PARAMETER	SYMBOL	TEST PIN	MIN.	TYPE	MAX.	UNIT
ENABLE CYCLE TIME	T_C	E	500	-	-	NS
ENABLE PULSE WIDTH	T_W		300	-	-	
ENABLE RISE/FALL TIME	T_R,T_F		-	-	25	
RS,R/W SETUP TIME	T_{SU1}	RS; R/W	100	-	-	
RS,R/W ADDRESS HOLD TIME	T_{H1}		10	-	-	
DATA SETUP TIME	T_{SU2}	DB0-DB7	60	-	-	
DATA HOLD TIME	T_{H2}		10	-	-	

BLOCK DIAGRAM





VDD-VO: LCD DRIVING VOLTAGE

VR: 10K-20KΩ

XIAMEN OCULAR LCD DEVICES CO.,LTD. A4

SOUTH 5/F, GUANGXIA BUILDING, TORCH HIGH-TECH DEVELOPMENT AREA,
XIAMEN 361006.P.R.CHINA TEL: 86-592-5715579 FAX: 86-592-6026021

RELIABILITY AND LIFE TIME

1.RELIABILITY TEST

STORAGE CONDITION	CONTENT	EVALUATIONS AND ASSESSMENT*			
		CURRENT CONSUMPTION	OOZING	CONTRAST	OTHER APPEARANCES
OPERATION AT HIGH TEMPERATURE AND HUMIDITY	40□,90% RH,240HRS	TWICE INITIAL VALUE OR LESS	NONE	MORE THAN 80% OF INITIAL VALUE	NO ABNORMALITY
HIGH TEMPERATURE STORAGE	60□, 240HRS	TWICE INITIAL VALUE OR LESS	NONE	MORE THAN 80% OF INITIAL VALUE	NO ABNORMALITY
LOW TEMPERATURE STORAGE	-20□, 240HRS	TWICE INITIAL VALUE OR LESS		MORE THAN 80% OF INITIAL VALUE	NO ABNORMALITY

*EVALUATIONS AND ASSESSMENT TO BE MADE TWO HOURS AFTER RETURNING TO ROOM TEMPERATURE (25□±5□).

*THE LCDS SUBJECTED TO THE TEST MUST NOT HAVE DEW CONDENSATION.

2. LIQUID CRYSTAL PANEL SERVICE LIFE

50,000 HOURS MINIMUM AT 25±10□,45±20%RH.

SOUTH 5/F. GUANGXIA BUILDING. TORCH HIGH-TECH DEVELOPMENT AREA.
XIAMEN 361006.P.R.CHINA TEL: 86-592-5715579 FAX: 86-592-6026021

STANDARD CHARACTER PATTERN

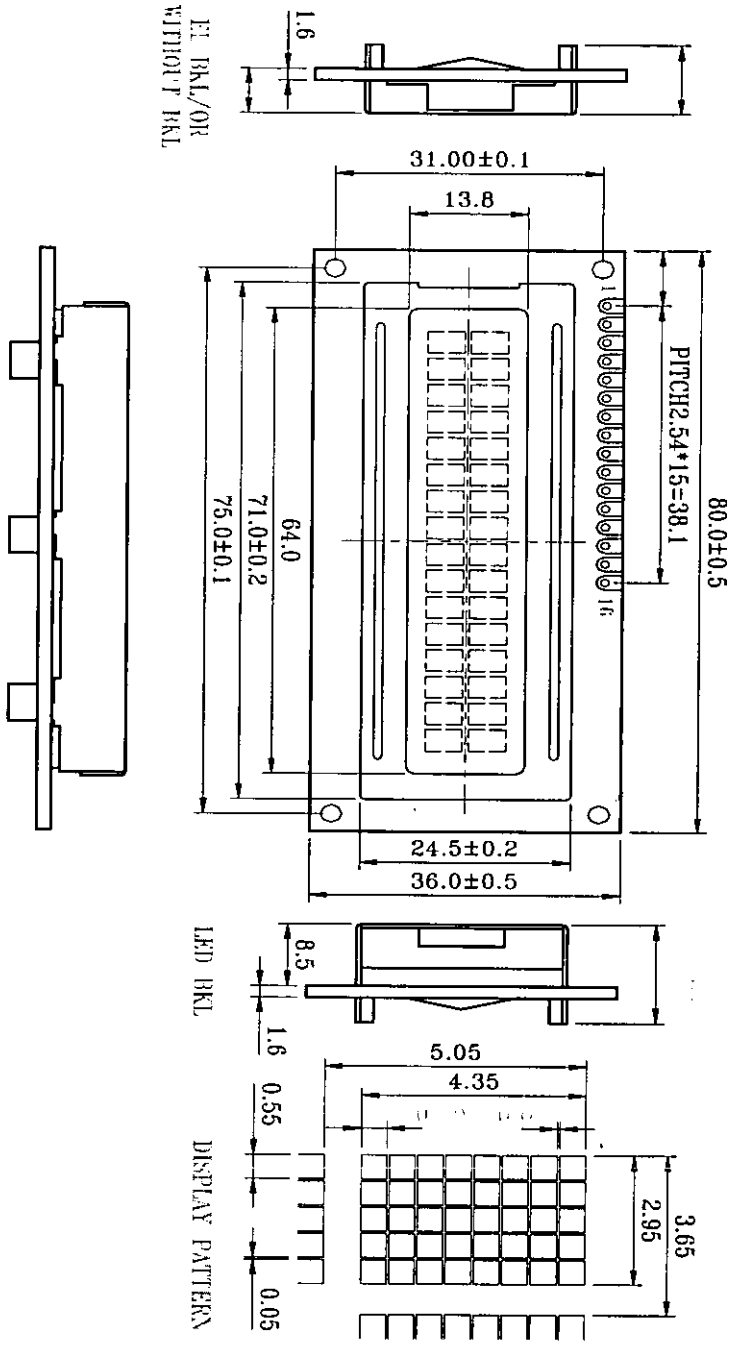
Upper 4 Bits / Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				ー	タ	ミ	α	p
xxxx0001	(2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
xxxx0010	(3)		"	2	B	R	b	r			「	イ	ツ	メ	ß	θ
xxxx0011	(4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
xxxx0100	(5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
xxxx0110	(7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
xxxx1000	(1)		(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
xxxx1001	(2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	⁻¹	y
xxxx1010	(3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
xxxx1011	(4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
xxxx1100	(5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
xxxx1101	(6)		−	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
xxxx1110	(7)		.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
xxxx1111	(8)		/	?	O	_	o	←			ッ	ソ	マ	°	ö	

NOTE: the user can specify any pattern for character-generator RAM

XIAMEN OCULAR LCD DEVICES CO.,LTD. A6

SOUTH 5/F, GUANGXIA BUILDING, TORCH HIGH-TECH DEVELOPMENT AREA,
XIAMEN 361006.P.R.CHINA TEL: 86-592-5715579 FAX: 86-592-6026021

PERFORMANCE FEATURES	
LC FLUID:	TN, STN
POLARIZER:	REFLECTIVE, TRANSFLECTIVE
	TRANSMISSIVE
COLOR:	GRAY,YELLOW,BLUE
BACKLIGHT:	LED, YELLOW-GREEN
TEMPERATURE RANGE:	STANDARD, WIDE
CONTROLLER:	KS0066



PIN	SIGNAL	PIN	SIGNAL
1	VSS	9	DB2
2	VDD	10	DB3
3	V0	11	DB4
4	RS	12	DB5
5	R/W	13	DB6
6	E	14	DB7
7	DB0	15	LED+
8	DB1	16	LED-

XIAMEN OCULAR LCD DEVICES CO.,LTD.

A7

SOUTH 5/F, GUANGXIA BUILDING, TORCH HIGH-TECH DEVELOPMENT AREA,
XIAMEN 361006.P.R.CHINA TEL: 86-592-5715579 FAX: 86-592-6026021

APÉNDICE B

MICROCHIP

PIC16F84A

18-pin *Enhanced* Flash/EEPROM 8-Bit Microcontroller

MICROCHIP

PIC16F84A

18-pin *Enhanced* Flash/EEPROM 8-Bit Microcontroller

Devices Included in this Data Sheet:

- PIC16F84A
- *Extended voltage range device available* (PIC16LF84A)

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- *1024 words of program memory*
- 68 bytes of data RAM
- 64 bytes of data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt on change
  - Data EEPROM write complete

Peripheral Features:

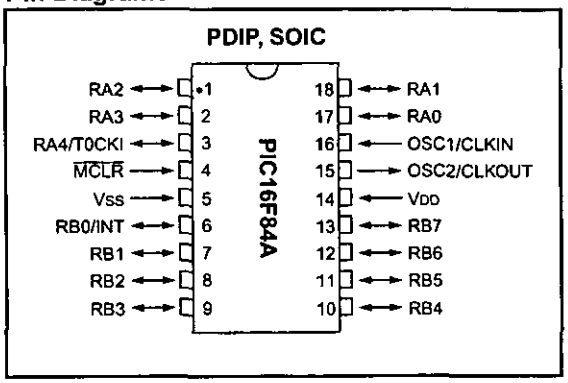
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

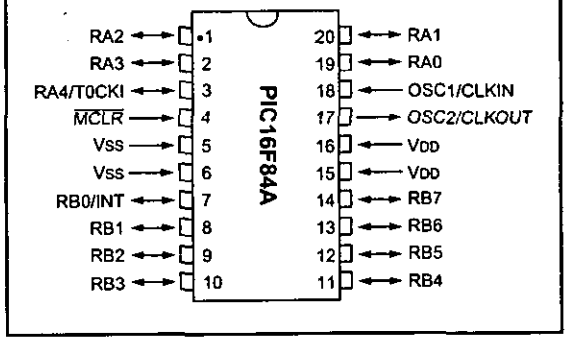
- 1000 erase/write cycles *Enhanced* Flash program memory
- 1,000,000 typical erase/write cycles EEPROM data memory
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code-protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS *Enhanced* Flash/EERPOM Technology:

- Low-power, high-speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

© 1998 Microchip Technology Inc.

ELECTRICAL CHARACTERISTICS FOR PIC16F84A

Absolute Maximum Ratings †

Ambient temperature under bias ..... -55°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to Vss (except VDD, $\overline{MCLR}$, and RA4) ..... -0.3V to (VDD + 0.3V)
Voltage on VDD with respect to Vss ..... -0.3 to +7.5V
Voltage on $\overline{MCLR}$ with respect to Vss(1) ..... -0.3 to +14V
Voltage on RA4 with respect to Vss ..... -0.3 to +8.5V
Total power dissipation(2) ..... 800 mW
Maximum current out of Vss pin ..... 150 mA
Maximum current into VDD pin ..... 100 mA
Input clamp current, IIK (VI < 0 or VI > VDD) ..... ± 20 mA
Output clamp current, IOK (VO < 0 or VO > VDD) ..... ± 20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 20 mA
Maximum current sunk by PORTA ..... 80 mA
Maximum current sourced by PORTA ..... 50 mA
Maximum current sunk by PORTB ..... 150 mA
Maximum current sourced by PORTB ..... 100 mA

Note 1: Voltage spikes below Vss at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin rather than pulling this pin directly to Vss.

Note 2: Power dissipation is calculated as follows: $Pdis = VDD \times \{IDD - \sum IOH\} + \sum \{(VDD-VOH) \times IOH\} + \sum(VOl \times IOL)$.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PRELIMINARY

DC CHARACTERISTICS: PIC16F84A-04 (Commercial, Industrial)
PIC16F84A-20 (Commercial, Industrial)

DC Characteristics Power Supply Pins			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial)				
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D001	VDD	**Supply Voltage**	4.0	—	5.5	V	XT, RC and LP osc configuration
D001A			4.5	—	5.5	V	HS osc configuration
D002*	VDR	**RAM Data Retention Voltage** (Note 1)	1.5*	—	—	V	Device in SLEEP mode
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	VSS	—	V	See section on Power-on Reset for details
D004*	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05*	—	—	V/ms	PWRT enabled (PWRTE bit clear)
D004A*			TBD	—	—		PWRT disabled (PWRTE bit set) See section on Power-on Reset for details
	IDD	**Supply Current** (Note 2)					RC and XT osc configuration (Note 4)
D010			—	1.8	4.5	mA	FOSC = 4.0 MHz, VDD = 5.5V
D010A			—	3	10	mA	FOSC = 4.0 MHz, VDD = 5.5V (During Flash programming)
							HS osc configuration (PIC16F84A-20)
D013			—	10	20	mA	FOSC = 20 MHz, VDD = 5.5V
D020	IPD	**Power-down Current** (Note 3)	—	7.0	28	µA	VDD = 4.0V, WDT enabled, industrial
D021			—	1.0	14	µA	VDD = 4.0V, WDT disabled, commercial
D021A			—	1.0	16	µA	VDD = 4.0V, WDT disabled, industrial
		Module Differential Current (Note 5)					
D022*	ΔIWDT	Watchdog Timer	—	6.0	20*	µA	WDTE bit set, VDD = 4.0V, commercial
			—	—	25*	µA	WDTE bit set, VDD = 4.0V, extended

* These parameters are characterized but not tested.

† *Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are* for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD, T0CKI = VDD, MCLR = VDD; WDT enabled/disabled as specified.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula IR = VDD/2Rext (mA) with Rext in kOhm.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD measurement.

Preliminary

DC CHARACTERISTICS: **PIC16F84A-04 (Commercial, Industrial)**
PIC16F84A-20 (Commercial, Industrial)
PIC16LF84A-04 (Commercial, Industrial)

DC Characteristics All Pins Except Power Supply Pins			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) Operating voltage VDD range as described in DC spec Section 9.1 and Section 9.2.				
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
		Input Low Voltage					
	VIL	I/O ports					
D030		with TTL buffer	VSS	—	0.8	V	4.5V ≤ VDD ≤ 5.5V (Note 4)
D030A			VSS	—	0.16VDD	V	entire range (Note 4)
D031		with Schmitt Trigger buffer	VSS	—	0.2VDD	V	entire range
D032		$\overline{MCLR}$, RA4/T0CKI	VSS	—	0.2VDD	V	
D033		OSC1 (XT, HS and LP modes)	VSS	—	0.3VDD	V	(Note 1)
D034		OSC1 (RC mode)	VSS	—	0.1VDD	V	
		Input High Voltage					
	VIH	I/O ports		—			
D040		with TTL buffer	2.0	—	VDD	V	4.5V ≤ VDD ≤ 5.5V (Note 4)
D040A			0.25VDD +0.8	—	VDD	V	entire range (Note 4)
D041		with Schmitt Trigger buffer	0.8 VDD	—	VDD		entire range
D042		$\overline{MCLR}$, RA4/T0CKI	0.8 VDD	—	VDD	V	
D043		OSC1 (XT, HS and LP modes)	0.7 VDD	—	VDD	V	(Note 1)
D043A		OSC1 (RC mode)	0.9 VDD	—	VDD	V	
D050	VHYS	Hysteresis of Schmitt Trigger inputs	—	0.1	—	V	
D070	IPURB	**PORTB weak pull-up current**	50*	250*	400*	µA	VDD = 5.0V, VPIN = VSS
		Input Leakage Current (Note 2,3)					
D060	IIL	I/O ports	—	—	±1	µA	VSS ≤ VPIN ≤ VDD, Pin at hi-impedance
D061		$\overline{MCLR}$, RA4/T0CKI	—	—	±5	µA	VSS ≤ VPIN ≤ VDD
D063		OSC1	—	—	±5	µA	VSS ≤ VPIN ≤ VDD, XT, HS and LP osc configuration

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. Do not drive the PIC16F84A with an external clock while the device is in RC mode, or chip damage may result.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: The user may choose the better of the two specs.

PRELIMINARY

DC CHARACTERISTICS: PIC16F84A-04 (Commercial, Industrial)
PIC16F84A-20 (Commercial, Industrial)
PIC16LF84A-04 (Commercial, Industrial)

DC Characteristics All Pins Except Power Supply Pins			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C (commercial) -40°C ≤ TA ≤ +85°C (industrial) Operating voltage VDD range as described in DC spec Section 9.1 and Section 9.2.				
Parameter No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Output Low Voltage					
D080	VOL	I/O ports	—	—	0.6	V	IOL = 8.5 mA, VDD = 4.5V
D083		OSC2/CLKOUT	—	—	0.6	V	IOL = 1.6 mA, VDD = 4.5V, (RC Mode Only)
		Output High Voltage					
D090	VOH	I/O ports (Note 3)	VDD-0.7	—	—	V	IOH = -3.0 mA, VDD = 4.5V
D092		OSC2/CLKOUT (Note 3)	VDD-0.7	—	—	V	IOH = -1.3 mA, VDD = 4.5V (RC Mode Only)
		Open Drain High Voltage					
D150	VOD	RA4 pin	—	—	8.5	V	
		Capacitive Loading Specs on Output Pins					
D100	COSC2	OSC2 pin	—	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1.
D101	CIO	All I/O pins and OSC2 (RC mode)	—	—	50	pF	
		Data EEPROM Memory					
D120	ED	Endurance	10K*	10M	—	E/W	25°C at 5V
D121	VDRW	VDD for read/write	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D122	TDEW	Erase/Write cycle time	—	4	8*	ms	
		Program Flash Memory					
D130	EP	Endurance	100*	1000	—	E/W	
D131	VPR	VDD for read	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D132	VPEW	VDD for erase/write	4.5	—	5.5	V	
D133	TPEW	Erase/Write cycle time	—	4	8	ms	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. Do not drive the PIC16F84A with an external clock while the device is in RC mode, or chip damage may result.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: The user may choose the better of the two specs.

PRELIMINARY

APÉNDICE C

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

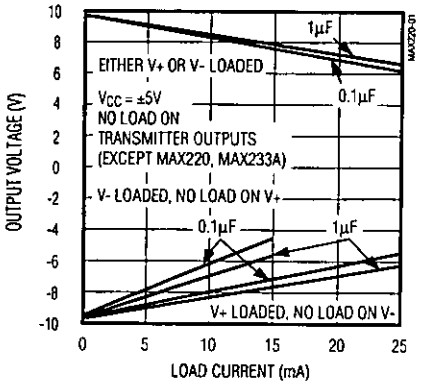
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

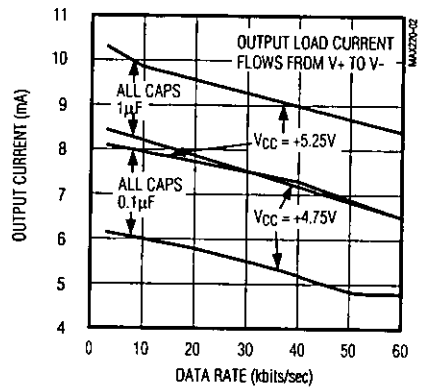
Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243

OUTPUT VOLTAGE vs. LOAD CURRENT



AVAILABLE OUTPUT CURRENT vs. DATA RATE



MAX222/MAX242 ON-TIME EXITING SHUTDOWN