UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

"DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS CON MICROCONTROLADORES PIC PARA EL LABORATORIO DE ELECTRÓNICA DE LA F. E. S. CUAUTITLÁN"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO MECÁNICO ELECTRICISTA

P R E S E N T A:

ALEJANDRO SOTO CARBAJAL

ASESOR: **ING. JORGE BUENDÍA GÓMEZ**

CUAUTITLÁN IZCALLI, EDO. DE MÉXICO 2003

A

TESIS
FALLA
DE
ORIGEN

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

"Diseño e implementación de prácticas con microcontroladores PIC para el laboratorio de electrónica de la F.E.S. Cuautitlan"

que presenta **el** pasante: **Alejandro Soto Carbajal**
con número de cuenta: **09324202-6** para obtener el título de :
Ingeniero Mecánico Electricista

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a **9** de **Diciembre** de **2002**.

PRESIDENTE	**Arq. José Luis Ríos Y Lorenzo**	
VOCAL	**Ing. Jorge Buendía Gómez**	
SECRETARIO	**M. C. Valentín Roldán Vázquez**	
PRIMER SUPLENTE	**Ing. Anselmo Angoa Torres**	
SEGUNDO SUPLENTE	**Lic. José Teofanes Zagal Díaz**	

A mis padres, Josefina Carbajal y Lorenzo Soto, por brindarme todo su cariño y apoyo incondicional para realizar mis estudios; de quienes tomo sus sabios consejos.

A mis hermanos, Laura, Javier y Elizabeth, por todo su apoyo.

Al Ing. Jorge Buendía Gómez, por haber aceptado dirigir este trabajo de tesis, por su valor como profesor y ser humano, y por todas sus enseñanzas.

A todos mis buenos y verdaderos amigos que tuve la oportunidad de conocer dentro de la Facultad y que siempre estuvieron a mi lado.

A mi querida Universidad, que me continúa dando la oportunidad de seguir formándome.

"Si el universo tiene un principio,
supondremos que tuvo un creador,
pero, si en efecto el cosmos se contiene a si mismo,
¿habrá lugar para un sumo hacedor?"

(STEPHEN HAWKING)

"Por un inmortal poder
todas las cosas lejanas
o cercanas, están
ocultamente ligadas entre
sí, de modo que no puedes
arrancar una flor sin perturbar
las estrellas"

(FRANCIS THOMPSON)

ÍNDICE

INTRODUCCIÓN

INTRODUCCIÓN.

Ante el inminente desarrollo de la tecnología y su avance a pasos agigantados, se hace necesario para el Ingeniero Mecánico Electricista, tener los conocimientos, al menos básicos, sobre los nuevos sistemas digitales que se implementan hoy en día en la industria, y en los que basan su eficiencia y rapidez en la producción.
En las más recientes décadas la industria ha desplazado modelos obsoletos en sus estructuras de producción y automatización, así es como circuitos integrados de la más alta tecnología y sistemas digitales basados en estos chips han permeado cada vez más llevando a cabo los procesos complicados en diversos campos.

Con el nacimiento de los microprocesadores en décadas pasadas y adjunto a ellos las microcomputadoras, la era digital se vio favorecida, no solo en la solución de problemas de índole matemático, sino también solucionando problemas en procesos industriales, análisis económicos, fenómenos biológicos y tratamiento directo de señales analógicas.
Por otra parte, en nuestros días, los microcontroladores han venido a revolucionar la forma de implementar sistemas digitales. Estos microcircuitos con recursos mayores que los microprocesadores, minimizan espacios, costos y diseños.
Además de los microcontroladores, se crearon otros circuitos más especializados para otras aplicaciones: dentro de ellos encontramos a los DSP's *(Digital Signal Processors)*, que enfocan sus aplicaciones en las telecomunicaciones por poseer características de velocidad alta.
Actualmente se están desarrollando e investigando nuevas posibilidades dentro de los microcircuitos, una de ellas son las llamadas MEMS, que son circuitos híbridos que no solo poseen características digitales, sino que dentro de su estructura tengan dispositivos analógicos, y no solo de características eléctricas, ya que se pretende incorporar elementos mecánicos e hidráulicos en el mismo silicio.

Al margen de la mención hecha anteriormente, el objetivo de esta tesis es elaborar un trabajo sobre microcontroladores PIC, y no se pretende hacer una comparación estricta

con otros circuitos; más bien aprovechar los recursos de los que disponen estos chips y elaborar diseños prácticos.

Baso este trabajo de investigación en el hecho de que a pesar de que mucha gente en nuestro país ya trabaja con microcontroladores, es factible darle un enfoque más hacia la investigación, sobretodo que un estudiante de electrónica debe de tener los conocimientos básicos sobre la materia. Conociendo las características de un modelo de un fabricante en específico, es muy probable tener bases para comprender el funcionamiento de otros microcontroladores de diversos fabricantes; ya sea un HC11 de Intel, un COP de National Semiconductors o un AVR de ATMEL.

El auge de los microcontroladores PIC inicio en el continente Europeo (Países como España y Yugoslavia). En el caso de Universidades Europeas, específicamente en España, los microcontroladores PIC de la empresa Microchip han tenido una gran demanda en diseños específicos de microrobots y PLC's, además de ser usados ampliamente por estudiantes de electrónica.

En nuestro país, en el último par de años han estado presentes diseños con PIC's y su demanda se ha incrementado considerablemente por parte de estudiantes y diseñadores.

A pesar de que la bases están dadas en algunas de las materias de la carrera de Ingeniería Electrónica, considero que se hace clara la necesidad de integrar en un plan de estudios posterior una materia sobre microcontroladores, incluyendo diversos modelos de ellos.

El trabajo de tesis que se presenta a continuación no se basa como tal en modelo de un sistema complejo, sino que pretende hacer un compendio, haciendo mención de los recursos de los que disponen los microcontroladores PIC y elaborando una serie de prácticas sencillas, en las que se tomen en cuenta todas las capacidades de los modelos de microcontroladores usados. En estas prácticas se utilizan dispositivos sencillos y de bajo costo para tales fines. Sin embargo, considero, que trabajando de esta manera, se tendrá un

claro panorama de lo que podría implementarse en sistemas de mayor grado de complejidad.

En el desarrollo de las prácticas, se utilizan básicamente dos modelos de microcontroladores pertenecientes a la familia de los PIC de Microchip, estos son los PIC16X8X y PIC16F87X (concretamente los PIC16F84 y PIC16F876).

Las justificaciones en las que fundamento el uso de estos dos tipos de microcontroladores son las siguientes:

- A pesar de que el PIC16F84 es un modelo de microcontrolador relativamente viejo, es básico para empezar a entender el funcionamiento de estos dispositivos. Su reducido repertorio de instrucciones hace que los programas editados en su propio lenguaje ensamblador no sean complicados de entender, aún para los estudiantes que poco conocen sobre el tema. Los recursos de los que dispone son los suficientes para la implementación de sistemas sencillos y confiables.
- Los PIC16F87X por decirlo así, son la descendencia directa del viejo PIC16F84, ya que poseen casi las mismas características y el mismo número de instrucciones. Su diferencia radica en la capacidad de memoria, en el número de registros, en el número de pines, número de puertos de entrada - salida además de la novedosa incorporación de un convertidor analógico – digital de 10 bits de resolución. En pocas palabras, este modelo contiene más argumentos para elaborar un diseño mas complejo.
- Es claro que la mayor justificación que encuentro, es el bajo costo de estos circuitos. No se hará uso alguno de tarjetas de desarrollo de Microchip, ya que tienen un precio elevado.
- El circuito programador no representa ningún problema en este proyecto, ya que Microchip proporciona el protocolo de programación en su página Web y mediante este es fácil diseñar un circuito que otorgue las señales necesarias para que se grabe el PIC; además de que también en la red existen diversos diseños de programadores "caseros" que suelen ser muy confiables y con un alta

compatibilidad con computadoras personales y ambiente Windows o MS – DOS.

No se puede hablar de que los microcontroladores PIC son los mejores, ni los menos indicados, su uso radica en satisfacer las necesidades de un diseño en especial. En este caso se eligieron ciertos tipos de PIC para dar una visión un poco clara de los elementos técnicos de diseño mediante estos.

Debo admitir que se presentaron muchas complicaciones en la elaboración de la tesis, sin embargo, considero que un Ingeniero debe de abarcar ciertos temas que se relacionen directamente con su área de trabajo, más aún si las bases para la comprensión de estos están a la mano.

CAPÍTULO 1

ANTECEDENTES

ANTECEDENTES

1.1. Antecedentes.

Durante la época de los 70's surgieron en el mercado circuitos integrados denominados microprocesadores. Nacieron de la necesidad de revolucionar más ampliamente la electrónica digital. Los sistemas que se desarrollaron en un principio fueron más enfocados a sistemas computacionales, es decir, implementar sistemas que son capaces de realizar tareas a altas velocidades.
Posteriormente se usaron estos tipos de circuitos integrados en sistemas de control e instrumentación industrial ofreciendo nuevas posibilidades.
Sin embargo una de las grandes desventajas que presentan estos circuitos, es que para el procesamiento y control de datos necesita dentro de su entorno dispositivos externos, como son memorias de tipo ROM, memorias tipo RAM, además de dispositivos de E/S lo que se conoce mejor como puertos.

A partir de estas carencias comprendidas en el entorno de los microprocesadores, algunos fabricantes se plantearon la posibilidad y el objetivo de englobar en un solo circuito las características externas de las que carecía un microprocesador.
El circuito que nació de este nuevo enfoque es conocido como microcontrolador.
Las aplicaciones más específicas que encuentra un microcontroladores están más destinadas hacia el ramo industrial.

En general las características de las familias de microcontroladores no difieren demasiado de un fabricante a otro, pero cierto tipo de microcontroladores no poseen ciertas características que poseen otros, por ejemplo, algunos pueden carecer de memoria ROM y tienen que depender de una memoria de este tipo externa para guardar el programa.

En la actualidad, existen infinidad de modelos y de fabricantes de microcontroladores, la elección de alguno de ellos dependerá de la aplicación en algún campo en específico, así como de las necesidades de que cada diseñador considere.

1.2. Microrobots.

Una de las aplicaciones en las que actualmente los microcontroladores fundamentan su aplicación en el campo industrial, es la microrobótica. La microrobótica es una forma de lo que se conoce como robótica industrial. Artefactos con características que faciliten determinadas tareas laborales que al hombre le resultarían difíciles de realizar.
Dependiendo de las características específicas de cada microcontrolador, tiene habilidades para que con un programa relativamente sencillo de realizar puedan manejar dentro de su entorno dispositivos como motores, sensores, relevadores o actuadores. Así es definido el principio básico de operación de un robot.

Un microrobot en términos más específicos, se define como un dispositivo móvil que reacciona ante el entorno de acuerdo con un plan de acciones programados por el usuario. El movimiento se basa en la acción de motores perfectamente sincronizados y para reconocer el medio, requiere el uso de sensores.
La gran variedad de tareas que puede realizar un robot es infinita (limpiar, explorar, analizar, controlar, etc.). La tarea que tendrían que realizar estos artefactos, no sería tan importante, si no fuera por la precisión y rapidez con que la deben de ejecutar.

Uno de estos más grandes ejemplos de realización o la creación de artefactos tan complejos en su estructura mediante la utilización de microcontroladores, es la tecnología que presenta el MARS GLOBAL SOJOURNER, robot que fue enviado a Marte por la NASA para exploración del planeta.

Como se ha visto las posibilidades que engloba un diseño mediante un microcontrolador son incalculables, y solo la imaginación de los ingenieros que proyectan y programan artefactos inteligentes constituye el límite de sus posibilidades. Prueba de la importancia y su proyección de futuro en las facultades de ingeniería, informática y electrónica de todo el mundo es la organización de certámenes anuales en los que participan los ingenieros más inverosímiles.

Las posibilidades de aplicación de los microcontroladores cada vez son mayores, pero también cada vez requieren más prestaciones.

1.3. Posibilidades de los microcontroladores.

Los microcontroladores son circuitos integrados que contienen en su estructura interna dispositivos internos de los cuales carecen los microprocesadores. Los microcontroladores nacen de la necesidad de propósitos específicos dentro de la industria, y de aplicaciones prácticas.
A diferencia de un microprocesador el cual necesita de una serie de circuitería externa extra para integrar interfaces con el "mundo exterior" como son líneas de entrada/salida, memoria, timers, etc., los microcontroladores (dependiendo del fabricante) contienen algunos o todos de estos recursos en su estructura interna. Muchos de ellos poseen unidades de memoria. En resumen un microcontrolador posee dentro de él, procesador y periféricos dentro de un solo circuito integrado.

La arquitectura en la que se basan actualmente la mayoría de los microcontroladores es del tipo RISC *(Reduced Instruction Set)*, ya que su repertorio de instrucciones es muy reducido, esta es otra de las diferencias que tiene con un microprocesador los cuales tienen una composición del tipo CISC con un número mucho más elevado de instrucciones.

La funcionabilidad es la parte más importante en la que difieren un microprocesador y un microcontrolador. La capacidad de almacenamiento de datos o del programa del microcontrolador, depende de que tenga o no una memoria de tipo ROM, en algunos casos se debe disponer de una memoria exterior para guardar el programa.

La unidad de control de procesamiento como en un microprocesador, tiene la capacidad de multiplicar, dividir, sumar, sustraer y mover sus contenidos de una

localización de memoria a otra. Las localizaciones de memoria son generalmente llamados registros.

Los buses son representados con un grupo de varias líneas. Existen diversos tipos de buses: de direcciones, de datos, de control y de alimentación. Los buses básicamente sirven, para transmitir direcciones de la memoria y para conectar todos los bloques del microcontrolador.

1.3.1. Arquitectura Harvard.

La arquitectura que siguen los procesadores internos en algunos tipos de microcontroladores es el modelo Harvard. En esta arquitectura, el CPU se conecta de forma independiente y con buses distintos de la memoria de instrucciones y con la base de datos. La arquitectura Harvard permite a la CPU acceder de manera simultanea a las dos memorias. Además propicia numerosas ventajas al funcionamiento del sistema.

Generalmente también se aplica la técnica de segmentación (pipe-line) en la ejecución de las instrucciones del microcontrolador. La segmentación permite al procesador realizar al mismo tiempo la ejecución de la segunda instrucción y la búsqueda del código de la siguiente. De esta forma se puede ejecutar cada instrucción en un ciclo (un ciclo de instrucción equivale a 4 ciclos de reloj). La segmentación permite al procesador ejecutar cada instrucción en un ciclo de instrucción.

En cada ciclo se realiza la búsqueda de una instrucción y la ejecución de la anterior. Las instrucciones de salto ocupan dos ciclos al no conocer la dirección de la siguiente instrucción hasta que no se haya completado la de bifurcación.

El formato de las instrucciones en un microcontrolador, no solo de cada fabricante sino también de cada modelo. Así, pueden tener las instrucciones un formato de 10, 12, 14 bits y más según sea el caso. Es conveniente usar procesadores que tengan una longitud de palabra grande, ya que, es una característica muy ventajosa en la optimización de la

memoria de instrucciones y facilita enormemente la construcción de ensambladores y compiladores.

1.4. ¿Cuál microcontrolador usar?

La respuesta no es difícil, en cierto caso cada uno debe saber que condiciones satisfacer para un cierto tipo de uso. En el inicio de la era de los microcontroladores, dos compañías productoras dominaban el mercado, INTEL y MOTOROLA. Por ciertas circunstancias en muchos países Intel llegó a ser más popular, por un lado los microcontroladores de esta compañía eran más accesibles. Por otro lado, la buena cooperación que existía entre Intel y las Universidades, además de las herramientas de desarrollo que Intel daba a estas, hicieron una base sólida para trabajar fácilmente dentro del entorno de Intel. Se debe admitir que el precio de los microcontroladores de Intel (que siempre han sido más económicos que los de Motorola) englobaba esta situación. Una ligera pasividad de Motorola en los mercados no Americanos y la orientación que le dio esta compañía a los grandes sistemas, solo incremento el número de usuarios de Intel en muchos países. Este número de usuarios se ve incrementado cuando Intel otorga la licencia de los microcontroladores basados en el 80C31 para otras compañías.

Compañías como Dallas Semiconductor y Siemens, incrementan el desempeño de los chips y por otro lado tienen la compatibilidad con el 80C31. Esto ha permeado en la actualidad, pero aparecen nuevas compañías e inician la producción de microcontroladores.

Microchip.

El primer sobresaliente sobre estas dos grandes compañías mencionadas anteriormente fue Microchip de E.U. en Arizona. Su inicio silencioso en el mercado fue a la vez parte de su éxito. Ellos anunciaron un hueco en el mercado. Calculando las ventajas y desventajas tomo el riesgo esta compañía y rápidamente se coloco dentro del mercado. Esta compañía ofreció algo completamente nuevo: un microcontrolador con una arquitectura simple, un reducido número de instrucciones (RISC) y mínimo poder de

disipación. Si bien Microchip ha estado por un periodo relativamente corto en el mercado estos microcontroladores PIC ganan un largo número de usuarios, gracias también al gran soporte que ofrece esta compañía y al reducido costo de los chips, (especialmente la versión OTP es muy usada). Es de mencionar que Microchip fue pionera en introducir un microcontrolador de 8 pines solamente. De esta manera han evolucionado los recursos de estos microcontroladores hasta tener hoy en día chips con osciladores internos, convertidores analógico-digitales, etc.

Atmel.

Casi al mismo tiempo con la aparición de Microchip, otra compañía prepara estrategias para incluir en el mercado sus propios microcontroladores, el nombre de ésta es Atmel. La historia de esta compañía es menos arriesgada. Su primer microcontrolador fue basado en la arquitectura del ya conocido 80C31 de Intel, pero Atmel implantó en el una memoria flash de gran capacidad, la cual era muy sencilla de reprogramar. Basado en este diseño su progreso fue más simple.

El próximo paso importante que dio Atmel fue introducir al mercado su microcontrolador AT89C1051, y posterior a este el AT89C2051, que solamente difieren en la capacidad de su memoria flash (1 y 2 Kb respectivamente).

Posteriormente Atmel decide desarrollar una línea completamente nueva de microcontroladores, también como Microchip y otras compañías, con tecnología RISC, pero con un repertorio un poco más amplio de instrucciones.

El nombre de las más recientes familias de microcontroladores de Atmel es AVR y sus designaciones son AT90S, AT90S1200 con 1Kb de memoria flash, encapsulado DIP con 20 pines; el AT90S8515 con 8 Kb de memoria flash, encapsulado DIP de 40 terminales y comunicación SCI, SPI.

El Atmega103 tiene 128 Kb de memoria flash, 4 Kb de EEPROM y 4 Kb de memoria RAM, convertidor analógico-digital de 10 bits, comunicación SCI, SPI y empaquetado con 64 pines, es este último una excelente versión.

Soporte en software.

Lo opción de software es muy importante, cuando se selecciona un microcontrolador. El precio de las herramientas de software pueden ser excesivamente altas. Atmel dio un gran paso al elaborar un excelente ensamblador y simulador para sus microcontrolador AVR y ser usado en plataformas de Windows.
Motorola por su parte posee el Pcbug, el viejo ensamblador ASM11 y programas para crear y compilar en lenguaje C para uso público. Microchip también ofrece mediante su página web ensambladores y simuladores.
Intel ha estado durante un periodo más prolongado en el mercado y probablemente es la compañía que más soporte ofrezca a sus usuarios.

Otros fabricantes.

Existen muchos más fabricantes de microcontroladores, entre algunos de ellos figuran SGS Thompson, Hitachi, National (con los famosos COP); todos ellos que gradualmente han permeado el mercado.
Intel y Motorola han dejado de ser por mucho las mayores productoras de microcontroladores para cederles el paso a otras compañías. Sin embargo poseen una gran tradición en microcontroladores de 16 y 32 bits además de un gran número de usuarios.

CAPÍTULO 2

LOS MICROCONTROLADORES PIC

LOS MICROCONTROLADORES PIC.

2.1. ¿Qué es un PIC?

Como se dijo en el apartado pasado un microcontrolador PIC, es un circuito integrado fabricado por la empresa MICROCHIP.

Un microcontrolador PIC ha sido planteado desde su concepción como un controlador de periféricos. Un PIC ***(Peripheral Interface Controller)*** es un circuito integrado desarrollado específicamente para el control de dispositivos periféricos; haciendo la función de una Unidad Central de Procesamiento.

Haciendo una analogía simple entre un cuerpo humano, el cerebro sería la CPU y el PIC sería como el sistema nervioso central, generando así señales digitales que hacen que un dispositivo eléctrico, mecánico o neumático, pueda ser controlado de manera sencilla.

Un microcontrolador PIC en términos simples lo podemos definir como una pequeña computadora, este circuito, como una CPU hace cálculos aritméticos de funciones utilizando solo software. La capacidad de almacenar programas dentro de la memoria, así como la frecuencia de funcionamiento de un PIC viene dada por el modelo de microcontrolador.

Estos microcircuitos fueron diseñados para minimizar espacios en el diseño de sistemas englobando en un solo circuito integrado muchas de las posibilidades de las que carecían anteriormente circuitos como los microprocesadores.

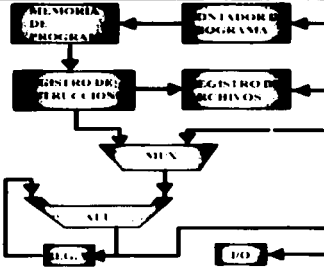


Figura 2.1 Composición esquemática generalizada para los microcontroladores PIC.

2.2. Características.

Entre más de 50 fabricantes de microcontroladores que existen en el mundo es muy difícil seleccionar alguno. En cada aplicación, las características de funcionamiento y desempeño, determinarán el más conveniente.

Los PIC por ahora tienen una gran aceptación entre la comunidad de técnicos y aficionados que trabajan con microcontroladores. El éxito, en algún momento, de estos PIC terminará algún día con el advenimiento de otro producto que salga al mercado y le robe la imagen.

La razón de una excelente acogida de estos circuitos entre los profesionales de la microelectrónica y microinformática, por mencionar algunos, son los siguientes:

- Sencillez de manejo.
- Buena información.
- Precio.
- Buen promedio de parámetros: velocidad, consumo, tamaño, alimentación, código compacto.

- Herramientas de desarrollo.
- Diseño rápido.
- La gran variedad de modelos PIC permite el que mejor responde a los requerimientos de aplicación.

En los albores del siglo XXI y en su reducida vida, los PIC ocupan las posiciones de cabeza en el ranking mundial, compitiendo con gigantes como Intel y Motorola.
En 1990 ocupaba el vigésimo puesto, pero a medida que pasa el tiempo adquiere más clientes, y actualmente son más de 100 millones de microcontroladores PIC los que vende Microchip cada año.

Dentro de los diferentes tipos de microcontroladores los hay que procesan datos de 4, 8, 16 y 32 bits, por su sencillez, el más popular y representativo es el de 8 bits, por que resulta el más sencillo y eficaz para la mayoría de los diseños típicos.

Una de las labores más importantes del ingeniero de diseño es la elección del modelo de microcontrolador que mejor satisfaga las necesidades del proyecto con el mínimo de presupuesto.

En 1997 Microchip disponía de 52 versiones de microcontroladores PIC, y 4 diferentes familias de 8 bits para adaptarse a las necesidades de clientes potenciales.

Como ya se ha mencionado, los microcontroladores PIC, son fabricados por Microchip Technology, que tiene sus principales fábricas de producción en Arizona. Este fabricante desarrolla una amplia gama de microcontroladores de muy diversas características, pero todos ellos basados en la misma arquitectura o estructura interna y los repertorios de instrucciones son prácticamente los mismos, por lo que, conocido el funcionamiento de uno de ellos, es muy sencillo adaptarse a cualquier otro modelo. Las diferencias esenciales entre unos modelos y otros estriban en la mayor o menor capacidad de memoria y en las extensiones o comunicaciones con el exterior (E/S) que serán diferentes en función de la aplicación que vayamos a dar en cada proyecto.

Para lograr una compactación de código óptima y una velocidad superior a la de sus competidores, los microcontroladores PIC incorporan en su procesador 3 de las características más avanzadas:

- Procesador tipo RISC.
- Procesador segmentado.
- 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 ortogonalidad en el formato de las instrucciones, que en el caso de los PIC de la gama media tienen longitud de 14 bits. De esta forma se consigue una compactación en el código del programa, en el caso de los PIC16X84 2.24 veces mayor al de los 68HC05, funcionando a la misma frecuencia.

En general, podemos afirmar que todos los PIC siguen el modelo Harvard de estructura interna, de forma tal que, la Unidad Central de Proceso dispone de los buses necesarios para comunicarse por un lado con la memoria de programa y por otro lado con la memoria de datos, de forma totalmente independiente.

Las siguientes figuras muestran el modelo Harvard simplificado para los modelos de microcontroladores PIC16F84 y PIC16F87X, con los cuales se trabajara:

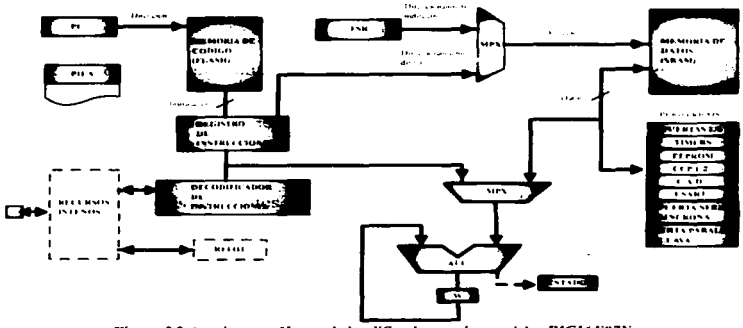


Figura 2.2 Arquitectura Harvard simplificada para los modelos PIC16F87X.

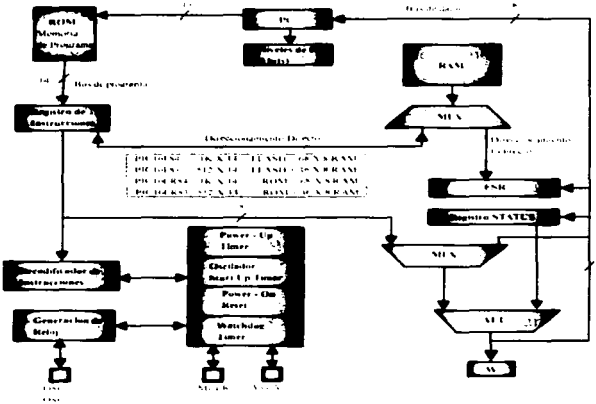


Figura 2.3 Arquitectura Harvard simplificada para los modelos PIC16X8X.

En la figura siguiente podemos ver, cómo internamente, el microcontrolador dispone de buses separados para acceder a las distintas unidades de memoria.

Figura 2.4 Organización de la memoria en los PIC

Estos microcontroladores, están diseñados según la tecnología RISC, por la cual disponen de un juego de instrucciones bastante reducido. Además, todas las instrucciones pueden controlar o trabajar con el elemento (registro, etc.) del microchip que se precise, bien como elemento destino o como elemento fuente, ya que todos ellos (puertos de entrada/salida, posiciones de memoria, etc.) están implementados físicamente como registros.

En la ejecución de las instrucciones, durante el desarrollo de un programa, los PIC emplean un procedimiento de segmentación *(pipeline)* que les permite ejecutar la instrucción en curso y al mismo tiempo buscar el código de la instrucción siguiente, ahorrando de este modo un precioso tiempo de ejecución. Cada instrucción se ejecuta en "un ciclo de instrucción" que equivale a "cuatro ciclos de reloj".

Con la estructura segmentada se pueden realizar simultáneamente 2 fases en las 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 diferenciación de los 2 tipos de memoria permite que cada uno tenga la longitud y el tamaño más adecuados.

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.

La ALU efectúa sus operaciones lógico aritméticas con 2 operandos, uno que recibe desde el registro W (work), que hace las veces de "acumulador" en los que los microprocesadores convencionales, y otro que puede provenir de cualquier registro o en W. Esta funcionalidad da un carácter completamente ortogonal a las instrucciones que pueden utilizar cualquier registro como operando fuente y destino. La memoria de datos RAM implementa en sus posiciones los registros específicos y los de propósito general.

Los microcontroladores PIC de Microchip, abarcan una gran variedad de modelos, que sin duda, permiten a los usuarios elegir el más interesante o apropiado para los intereses de su proyecto, permitiendo incluso la elección del más económico. Además la empresa Microchip pone a disposición del servicio técnico en general gran variedad de herramientas de diseño (simuladores, programadores, emuladores, compiladores, etc.) que facilitan el desarrollo de sistemas de control.

2.2.1. Ventajas de los PIC.

Como ya se dijo disponen de un procesador RISC (Computador de juego de instrucciones reducidas). Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, los de la gama media disponen de 35 instrucciones y casi 60 instrucciones los de la gama alta.

Todas las instrucciones son ortogonales. Cualquier instrucción puede manejar cualquier elemento de fuente o como destino.

Arquitectura basada en un banco de registros. Esto significa que todos los objetos del sistema (puertos de E/S, temporizadores, posiciones de memoria, etc.) están implementadas físicamente como registros.

Diversidad de modelos de microcontroladores con prestaciones y recursos diferentes. La gran variedad de modelos de microcontroladores PIC permite que el usuario seleccione el más conveniente para un proyecto determinado.

Herramientas de soporte potentes y económicas. La empresa Microchip y otras que utilizan los PIC ponen a disposición de los usuarios numerosas herramientas para desarrollar hardware y software. Son muy abundantes los programadores, los simuladores, los emuladores en tiempo real, ensambladores, compiladores C, interpretes y compiladores BASIC.

Los microcontroladores MICROCHIP combinan un alto desempeño a bajo costo en un empaquetado muy reducido, ofreciendo la mejor relación precio/rendimiento en la industria. Durante el surgimiento de ellos y su existencia, relativamente joven, más de 200 millones de estos productos se distribuyen para satisfacer los consumibles: periféricos de computadora, automatización, sistemas de control, aplicaciones en seguridad y telecomunicaciones. La combinación de los microcontroladores de 8-bits, con las tecnologías de OTP avanzada, EEPROM, memoria FLASH, memoria ROM y la industria: hacen de microchip un líder proveedor en el desarrollo de herrarmientaspara ajustarse mejor a las necesidades reales.

El alto nivel de integración reduce notablemente la cantidad de componentes externos y los costos de desarrollo, mejora el desempeño del sistema, reduce la interferencia electromagnética, minimiza el consumo de potencia y agiliza el tiempo de realización. Los microcontroladores cuentan con un repertorio de instrucciones compatible, y una gran variedad de periféricos y un amplio rango de empaquetados y de voltaje.

Combinando la características RISC con la arquitectura Harvard de bus-dual, los microcontroladores PIC de 8-bit de MICROCHIP son más rápidos y flexibles y es el núcleo de la arquitectura más popular para los nuevos diseños de microcontroladores. Disminuyendo a su vez, la migración entre familias de productos debido al simple set de instrucciones.

2.3. Gamas de microcontroladores PIC.

Descripción general y aplicaciones.

La diversidad de los modelos PIC tiene una finalidad: poder seleccionar el más adecuado para cada aplicación, por eso conviene tener un conocimiento fundamental y actualizado de cada gama.

Entre las familias con las que cuenta Microchip en su gama de microcontroladores, podemos mencionar las siguientes:

La gama enana es representada por la familia 12CXX, que es un conjunto de microcontroladores CMOS, con 8 patas, de precio muy bajo y que resultan altamente competitivos. Poseen un formato único para sus instrucciones, aunque hay modelos de 12, 14 y 16 bits. Los más usados son los de 16 bits, que tienen una extraordinaria compactación en el código de programa, pudiéndose cifrar en el doble con otros similares.

Al igual que todos los modelos de PIC, la arquitectura RISC, tipo Harvard, unida a la segmentación del procesador (pipeline) consigue que el ciclo de ejecución de una instrucción sea de 1µs cuando funciona a una frecuencia de 4 Mhz. Todas las instrucciones duran este tiempo, excepto las de salto que duran el doble.

Sus aplicaciones de esta gama se ven reflejadas en sistemas de seguridad y dispositivos de bajo consumo que gestionan trasmisores y receptores de bajas señales.

La gama baja representada por el PIC14000 resulta muy interesante por los recursos incorporados en el: convertidor analógico-digital de media y alta resolución (10 a 16 bits) sensor interno de temperatura, comunicación serie y muy bajo consumo. Son aptos para el control de cargadores de baterías, monitores del estado de pilas y baterías fuentes de alimentación ininterrumpibles, gestión del consumo de energía de alimentación y sistemas de adquisición de dato, especialmente de temperatura.

Otro microcontrolador de la gama baja, es el PIC16C5X, que tiene proyección hacia los dispositivos de alta velocidad usados en la industria de automoción el control de motores y los receptores trasmisores y procesadores de bajo consumo encargados de tareas relacionadas con las telecomunicaciones. Esta familia consta de recursos propios adicionales, pila de 5 niveles y múltiples fuentes de interrupción. En la gama media existe la subfamilia PIC16C62X, recomendable en aplicaciones relativas a cargadores de baterías y control de sensores remotos de bajo consumo.

Con 10 versiones diferentes, la subfamilia 16C6X, de la gama media se caracteriza por la incorporación de varios temporizadores, modulo de captura/comparación, puerta serie SPI e I^2C, puerta paralela PSP.
Mejorando estos modelos, la serie PIC16C7X es una muestra del procesador RISC funcionando a 20 Mhz con un ciclo de instrucción de 20 ns. Los componentes de esta familia son muy usados en los sistemas de seguridad, así como en el control remoto de sensores en automoción.

Los PIC16X8X y sus variantes tienen inclusión de memoria FLASH o EEPROM, para contener el código de programa. Aplicaciones típicas de esta familia es el control de puertas de garage, instrumentación, inmobilizadores de vehículos, tarjetas codificadas y pequeños sensores. La grabación de estos en el propio circuito les hace recomendables para el almacenamiento de datos de calibración y para modificación del programa en ciertas condiciones del entorno en que se ocupa el microcontrolador.

La serie 16C9XX incluye dentro de estructura un controlador programable de pantallas de cristal líquido (LCD).

La subfamilia PIC17C4X, con 58 instrucciones de 16 bits, tarda en ejecutar cada una 121 ns cuando funciona a frecuencia de 33Mhz. El núcleo del procesador esta mejorado y la pila contiene 16 niveles. La líneas de alimentación se aumentaron y contiene un multiplicador hardware de 8*8 bits en un ciclo., para aplicaciones de cálculo matemático, instrumentación, telecomunicaciones e industriales.

Los PIC17C752 y PIC 17C756, pertenecen a una gama alta y vienen en encapsulados de 64 y 68 pines respectivamente. Poseen una elevada capacidad de memoria que alcanza los 8k x 16 y 16k x16, en el área reservada a las instrucciones, 454 y 902 bytes para datos. Poseen 12 canales para un convertidor analógico digital de 10 bits, 4 canales para un módulo de captura de 16 bits, 2 USART, bus I^2C.

A principios del siglo XXI, Microchip lanza el mercado los microcontroladores RISC-FLASH, en la serie PIC16F87X, derivada de la subfamilia PIC16F84 pero con más recursos y más potente, con características similares de grabación y borrado eléctrico.

Es una serie que encaja dentro de la gama media y que alcanza una memoria de código de hasta 8 K de palabras de 14 bits en memoria FLASH. Soporta hasta 386*8 bytes de memoria RAM y 256*8 bytes de memoria EEPROM. Pueden programarse solo con 5VCD y existe acceso a lectura y escritura de la memoria de programa.
Estos microcontroladores pueden tener hasta tres timers o contadores, 2 módulos de captura, comparación y modulación por ancho de pulsos, convertidor analógico digital, de 10 bits, canal de comunicaciones síncrono SSP con módulo 12C y SPI, canal USART/SCI, puerta paralela esclava de 8 bits y detección de falta de alimentación.

MODELO.	PIC16F84	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Memoria de programa (FLASH).	Bytes: 1792 Palabras:1024x14	7168 4096x14	7168 4096x14	14336 8192x14	14336 8192x14
Memoria de datos.	EEPROM: 64 RAM: 68	128 192	128 192	256 368	256 368
C A/D.	No	5(10 bits)	8(10 bits)	5(10 bits)	8(10 bits)
Detección de baja tensión.	No	Si	Si	Si	Si
Líneas E/S	13	22	33	22	33
Comunicación serie.	No	USART/MSSP	USART/MSSP	USART/MSSP	USART/MSSP
CCP.	No	2	2	2	2
Temporizadores.	1-8bit, 1-WDT	1-16bit, 2-8bit 1-WDT	1-16bit, 2-8bit 1-WDT	1-16bit, 2-8bit 1-WDT	1-16bit, 2-8bit 1-WDT
Frec. Max. Mhz.	20	20	20	20	20
ICSP.	Si	Si	Si	Si	Si
Encapsulados.	18p, 18SO, 20SS	28SP, 28SO	40P, 44l., 44PQ, 44PT	28SP, 28SO	40P, 44L, 44PQ, 44PT
Fuentes de interrupción.	4	13	14	13	14
Com. Paralelo.	No	No	Si	No	Si

Tabla 2.1 Características de los microcontroladores PIC16F84 Y PIC16F87X.

CAPÍTULO 3

COMPOSICIÓN INTERNA Y REGISTROS DE LOS MICROCONTROLADORES PIC

COMPOSICIÓN INTERNA Y REGISTROS DE LOS MICROCONTROLADORES PIC.

Cabe mencionar antes de plantear las definiciones que estructuran este capítulo que, los conceptos referidos a los registros de los que disponen los microcontroladores PIC solo corresponden a los modelos que se usaran en las prácticas que se plantean al final de este trabajo de tesis.

Los PIC, al igual que otros tipos de procesadores emplea para su correcto funcionamiento, dos tipos de memoria, la memoria de solo lectura y la memoria de lectura y escritura. En este caso, el propio PIC incorpora en su propia estructura los dos tipos antes mencionados, que denominaremos memoria de programa y memoria de datos respectivamente.

3.1. La memoria RAM.

La memoria de datos tiene posiciones implementadas en la RAM. En la sección de la RAM, se alojan los registros fundamentales en el funcionamiento del procesador y en el manejo de todos sus periféricos, además de registros que el programador puede usar para información de trabajo propia de aplicación.

Según sea el caso, cada modelo de PIC contiene un cierto numero de bancos de memoria, conteniendo en ellos los registros; por ejemplo en los PIC de la gama media, la memoria de datos está organizada para alojar un máximo de 4 bancos de 128 bytes cada uno. En el caso de la familia PIC16X8X, el PIC16C84 sólo tiene implementados los 48 primeros bytes de los bancos 0 y 1. En el resto de los PIC de esta familia se destinan dos bits del registro *STATUS (RP0 y RP1)* para determinar el banco y otros 7 para elegir una de las 128 posiciones del banco seleccionado. El PIC16F84 tiene 68 bytes en RAM.

En el caso de los modelos de microcontroladores PIC16F87X, al igual que los modelos anteriores se disponen de 4 bancos de memoria de 128 bytes cada uno en la RAM

estática. En las posiciones iniciales de cada banco se ubican los registros específicos que gobiernan al procesador y sus recursos. Dos modelos de los PIC16F87X tienen 192 bytes de RAM (PIC16F873/4) y los otros dos 368 bytes (PIC16F876/7).

La memoria RAM, está compuesta por registros o bytes. El número de registros o bytes depende del elemento o PIC de la familia que consideremos.

Los registros de la memoria de datos se dividen básicamente en dos grupos: Los registros de funciones especiales y los registros de propósito general.

Los registros de funciones especiales son los siguientes:

- Temporizadores. Registros TMR (TMR0, TMR1, TMR2, etc.).
- Contador de programa PCL
- Registro de Estado STATUS
- Registro de selección de banco FSR
- Registros de E/S, PORTA, PORTB, PORTC, etc.
- Registro de direccionamiento indirecto INDF.

Los registros de funciones especiales se emplean para controlar las funciones de entrada/salida y realizar las necesarias configuraciones para el correcto funcionamiento del sistema.

Los registros de propósito general se emplean para manejar datos e información mediante las instrucciones del programa a ejecutar.

BANCO 0		BANCO 1	
INDF	00H	INDF	80h
TMR0	01H	OPTION	81H
PCL	02H	PCL	82H
STATUS	03H	STATUS	83H
FSR	04H	FSR	84H
PORTA	05H	TRISA	85H
PORTB	06H	TRISB	86H
X	07H	X	87H
EEDATA	08H	EECON1	88H
EEADR	09H	EECON2	89H
PCLATH	0AH	PCLATH	8AH
INTCON	0BH	INTCON	8BH
	0CH		8CH
68 Registros de Propósito General		Mapeados en el banco 0	
	4FH		CFH
	50H		DFH
X		X	
	7FH		FFH

Tabla 3.1 Memoria RAM en el PIC16F84 y el PIC16CR84.

BANCO 0		**BANCO 1**		**BANCO 2**		**BANCO 3**	
INDF	00H	INDF	80h	INDF	100H	INDF	180H
TMR0	01H	OPTION. REG	81H	TMR0	101H	OPTION. REG	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H	X	105H	X	185H
PORTB	06H	TRISB	86H	PORTB	106H	TRISB	186H
PORTC	07H	TRISC	87H	X	107H	X	187H
PORTD	08H	**TRISD**	88H	X	108H	X	188H
PORTE	09H	**TRISE**	89H	X	109H	X	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	PIE1	8CH	EEDATA	10CH	EECON1	18CH
PIR2	0DH	PIE2	8DH	EEADR	10DH	EECON2	18DH
TMR1L	0EH	PCON	8EH	EEDATH	10EH	RESERV.	18EH
TMR1H	0FH	X	8FH	EEADRH	10FH	RESERV.	18FH
T1CON	10H	X	90H	X	110H	X	190H
TMR2	11H	SSPCON2	91H		111H	X	191H
T2CON	12H	PR2	92H		112H	X	192H
SSPBUF	13H	SSPADD	93H		113H	X	193H
SSPCON	14H	SSPSTAT	94H	X	114H	X	194H
CCPR1L	15H	X	95H		115H	X	195H
CCPR1H	16H	X	96H		116H	X	196H
CCP1CON	17H	X	97H		117H	X	197H
RCSTA	18H	TXSTA	98H		118H	X	198H
TXREG	19H	SPBRG	99H		119H	X	199H
RCREG	1AH	X	9AH		11AH	X	19AH
CCPR2L	1BH	X	9BH		11BH	X	19BH
CCPR2H	1CH	X	9CH		11CH	X	19CH
CCP2CON	1DH	X	9DH		11DH	X	19DH
ADRESH	1EH	ADRESL	9EH		11EH	X	19EH
ADCON0	1FH	ADCON1	9FH		11FH	X	19FH
Registros de Propósito General 96 bytes	20H 7FH	Registros de Propósito General 80 bytes	A0H FFH	Mapeados con 20H-7FH	120H 17FH	Mapeados con A0H-FFH	1A0H 1FFH

Tabla 3.2 Memoria RAM para los PIC16F873-4 con 192 bytes útiles.

BANCO 0		BANCO 1		BANCO 2		BANCO 3	
INDF	00H	INDF	80h	INDF	100H	INDF	180H
TMR0	01H	OPTION. REG	81H	TMR0	101H	OPTION. REG	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H	X	105H	X	185H
PORTB	06H	TRISB	86H	PORTB	106H	TRISB	186H
PORTC	07H	TRISC	87H	X	107H		187H
PORTD	08H	**TRISD**	88H	X	108H		188H
PORTE	09H	**TRISE**	89H	X	109H		189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	PIE1	8CH	EEDATA	10CH	EECON1	18CH
PIR2	0DH	PIE2	8DH	EEADR	10DH	EECON2	18DH
TMR1L	0EH	PCON	8EH	EEDATH	10EH	**RESERV.**	18EH
TMR1H	0FH	X	8FH	EEADRH	10FH	**RESERV.**	18FH
T1CON	10H	X	90H	16 bytes	110H	16 bytes	190H
TMR2	11H	SSPCON2	91H		111H		191H
T2CON	12H		92H		112H		192H
SSPBUF	13H	SSPADD	93H		113H		193H
SSPCON	14H	SSPSTAT	94H		114H		194H
CCPR1L	15H	X	95H		115H		195H
CCPR1H	16H	X	96H		116H		196H
CCP1CON	17H	X	97H		117H		197H
RCSTA	18H	TXSTA	98H		118H		198H
TXREG	19H	SPBRG	99H		119H		199H
RCREG	1AH	X	9AH		11AH		19AH
CCPR2L	1BH	X	9BH		11BH		19BH
CCPR2H	1CH	X	9CH		11CH		19CH
CCP2CON	1DH	X	9DH		11DH		19DH
ADRESH	1EH	ADRESL	9EH		11EH		19EH
ADCON0	1FH	ADCON1	9FH		11FH		19FH
Registros de Propósito General 96 bytes	20H	Registros de Propósito General 80 bytes	A0H	Registros de Propósito General 80 bytes	120H	Registros de Propósito General 80 bytes	1A0H
			EFH		16FH		1EFH
		Mapeados con 70H-7FH	F0H	Mapeados con 70H-7FH	170H	Mapeados con 70H-7FH	1F0H
	7FH		FFH		17FH		1FFH

Tabla 3.3 Memoria RAM para los PIC16F876 7 con 368 bytes útiles.

3.2. Registros de control.

Para gobernar el funcionamiento de los recursos de los microcontroladores PIC, poseen un número de registros específicos cuyos bits soportan el control de los mismos. Generalmente o casi siempre estos registros están ubicados en las primeras posiciones de cada banco de memoria de datos RAM. Algunos de estos registros son compartidos tanto por la familia PIC16X84 como por la familia PIC16F87X.
A continuación se dará una breve descripción de los registros.

Registro de estado (STATUS).

Este es el registro más usado de todos, los bits que lo conforman se destinan al control de las funciones del procesador.
El registro se compone de la siguiente manera:

Figura 3.1 Estructura del registro STATUS.

Los tres bits de menos peso son señalizadores de ciertas condiciones en las operaciones lógico-aritméticas (cero y acarreos).

- **Z:** Señalizador de cero. Se pone a 1 cuando el resultado es cero.
- **C:** Acarreo/llevada. Se pone a 1 cuando existe acarreo en el bit de más peso en las instrucciones de suma. También actúa como señalizador de llevada de resta.
- **DC:** Acarreo/llevada. Funciona igual que el señalizador C, pero para el cuarto bit, y es muy útil para cifras expresadas en BCD.

Los señalizadores TO# y PD# sirven para indicar la causa de una reset de un procesador..

- **TO#:** Se activa a nivel bajo al desbordarse el *Watchdog Timer*. Toma el valor de 1 tras la conexión de la alimentación o al ejecutarse las instrucciones *clrwdt o sleep*.
- **PD#:** Se activa a cero al ejecutarse la instrucción *sleep*. Se pone a 1 automáticamente tras la conexión de la alimentación o al ejecutarse *clrwdt*.

RP0 y RP1 indican el banco de memoria que se selección.

BANCO	RP1	RP2
0	0	0
1	0	1
2	1	0
3	1	1

Tabla 3.4

Por último IRP se utiliza concatenado con el bit de más peso del registro FSR para el banco de RAM en el direccionamiento indirecto.

3.2.1. Registros para controlar las interrupciones.

En general se puede decir que todos los PIC pueden tener diversas causas que originen una interrupción. Los modelos PIC16X84 tienen 4 causas de interrupción (desbordamiento del TMR0, activación del pin de interrupción, cambio del estado de uno de los pines de menos peso del puerto B y finalización de la escritura de un byte en la EEPROM); mientras que los PIC16F87X tienen 13 posibles causas (además de las mencionadas para el 16X84, tenemos, el desbordamiento del TMR1, TMR2, captura o comparación en el módulo CCP1 y en el CCP2, transferencia del puerto serie, colisión de bus en el puerto serie, fin de la transmisión o recepción en el USART, fin de la conversión en el convertidor analógico/digital y por último, transferencia en el puerto paralelo esclavo).

Registro de control de interrupciones (INTCON).

Es un registro leible y escribible, en ambas familias de PIC aparece (16X84 y 16F87X), solo cambia en el bit 6, que en los nuevos PIC es PEIE (permiso de interrupción de los periféricos) y en el viejo PIC16F84 era EEIE (para permitir la interrupción cuando finalizase la escritura de un byte en la EEPROM). Este registro tiene la función de controlar las interrupciones provocadas por el TMR0, cambio de estado de las 4 líneas de más peso en el puerto B (RB4, 5, 6, 7) y activación del pin RB0.

En la figura siguiente se muestra el contenido de este registro para ambas familias de microcontroladores PIC, con cada uno de los bits que lo conforman teniendo un propósito de interrupción especial.

Figura 3.2 Estructura del registro INTCON.

- **GIE:** Bit de permiso global de interrupciones. 1 = permitido, 0 = prohibido.
- **PEIE o EEIE (en los PIC16F84):** Bit de permiso de los periféricos que no se controlan con *INTCON*.
- **TOIE:** Bit de permiso de interrupción del TMR0.
- **INTE:** Bit de permiso de la interrupción externa por RB0/INT.
- **RBIE:** Bit de permiso de la interrupción por cambio en RB4 – RB7.
- **TOIF:** Señalizador de desbordamiento en el TMR0.
- **INTF:** Señalización de activación de la patita RB0/INT.
- **RBIF:** Señalizador de cambio en Rb4 – RB7.

3.3. Otros registros.

Registro INDF.

El registro INDF, se encuentra en la posición 00H de la memoria de datos, sin embargo no es un registro que se encuentre implementado físicamente. Se emplea en el direccionamiento indirecto utilizando como puntero el contenido del registro FSR. Es decir, se accede a la posición señalada por el contenido de FSR.

Por ejemplo, supongamos que el registro general situado en 11H contiene el valor 3AH. Si cargamos el registro FSR con en valor 11H y a continuación realizamos una lectura del registro INDF, nos dará como resultado el valor 3AH.

Registro PCL.

Al igual que cualquier microprocesador o microcontrolador, los PIC para ejecutar el conjunto de instrucciones que constituyen su programa de aplicación, emplea un registro que va direccionando de forma adecuada las diferentes posiciones de memoria en las que se encuentran alojadas dichas instrucciones. Este registro especial recibe el nombre de Contador de Programa (PC). *La longitud total de este registro varía en función del componente de cada uno de los modelos de PIC.*

Vemos que la parte baja del PC se constituye con el registro PCL, mientras que los tres bits de más peso se obtienen del registro STATUS y de las palabras contenidas en las instrucciones empleadas en cada caso.
Si en un momento dado el PC está señalando la última dirección de una página de memoria determinada, al ejecutarse una instrucción ordinaria cualquiera (no de salto) se produce un autoincremento del PC señalando entonces a la primera posición de la página siguiente. Sin embargo, los bits del registro STATUS que nos sirven para preseleccionar la página de trabajo no se actualizan. Si en este momento aparece una instrucción GOTO o una instrucción CALL o cualquier otra instrucción que pretenda modificar el PC, nos enviará el

programa a la página que señalen los bits PA0 y PA1 del registro STATUS. Para prevenir esto, es conveniente actualizar dichos bits en el programa.

Registro FSR.

Se utiliza para poder llevar a cabo el direccionamiento indirecto de la memoria, en la ejecución de instrucciones del programa. Se usa en combinación con el registro INDF. El FSR es el puntero, es decir, su contenido nos indica la dirección de memoria a la que intentamos acceder de modo indirecto.

Registros PORTA, PORTB, PORTC, etc.

Estos registros pueden leerse y escribirse según nos interese mediante instrucciones del programa. Su misión esencial es servirnos de enlace entre el microcontrolador PIC y los elementos del mundo externo. Más adelante ampliaremos el uso de estos registros.

Registro OPTION.

Se trata de un registro que únicamente puede ser escrito, es decir, sobre él solamente pueden realizarse operaciones de escritura. Su misión principal es controlar el funcionamiento de Timer0 y del WDT. Dispone de varios bits de control que nos permiten manejar el WDT y el TIMER0.

Para escribir en este registro debemos utilizar la instrucción OPTION con la cual pasamos el contenido del registro de trabajo W al registro OPTION.

Un RESET nos coloca a nivel alto (1 lógico) los bits del registro OPTION.

Registro de trabajo W.

El registro de trabajo W, realiza funciones idénticas al Acumulador en los microprocesadores tradicionales. Se emplea para realizar operaciones aritméticas, lógicas,

etc. Se trata de un registro no perteneciente al área de memoria de datos. Se accede a este registro mediante determinadas instrucciones que hacen uso de él. En las instrucciones de dos operandos, uno de los dos datos siempre debe estar en el registro W, como ocurría en el modelo tradicional con el acumulador. En las instrucciones de un solo operando, el dato se toma de la memoria (también por convención). Las operaciones con constantes provenientes de la memoria de programa (literales) se realizan solo sobre el registro W.

NOTA: En cada unos de los modelos de microcontroladores PIC se tiene un determinado banco de memoria de datos, cada chip también contiene cierto número de los mismos bancos, los registros se sitúan dentro de estos ocupando una cierta dirección de memoria, en base a esto, algunos de los registros pueden ser los mismos en algunos de modelos de PIC, sin embargo no ocupan la misma dirección de memoria. Al usar un determinado modelo de microcontrolador debemos estar obligados a conocer las posiciones de cada uno de estos registros y los bits que lo componen para poder configurar bien nuestro circuito integrado.

3.4. Memorias EEPROM y FLASH.

Los PIC16X8X tienen 64 bytes de memoria EEPROM de datos, donde se pueden almacenar datos y variables que interesan que no se borren cuando se desconecta la alimentación del sistema. Soporta 1.000.000 de ciclos de escritura/borrado y es capaz de guardar la información sin alterarla más de 40 años. El PIC16F84A contiene una memoria flash de 1792 bytes.

Los PIC16F87X tienen 128 bytes (16F873/4) y 256 bytes (16F876/7) de memoria EEPROM respectivamente; además de contar también con memoria flash de 7168 bytes (16F873/4) y 14336 bytes (16F876/7).

Cabe mencionar que una memoria flash tiene una vida menor en cuanto a ciclos de borrado/escritura, a diferencia de una EEPROM con 1.000.000 de ciclos, la memoria flash tiene alrededor de solo 1.000 ciclos.

3.4.1. Lectura y escritura de las memorias EEPROM y FLASH.

En el PIC16F84 se puede escribir y leer la memoria de datos EEPROM. En los PIC16F87X también se puede escribir la memoria de código FLASH. En pocas palabras esto significa que un programa dinámicamente puede generar información que se puede grabar en la memoria FLASH directamente, sin necesidad de un programador externo.

Para manejar la memoria de EEPROM de 64 bytes del PIC16F84 bastan dos registros para proporcionar la dirección a acceder y para guardar el datos de 8 bits que se leía o se iba a grabar. Como solo existen 64 posiciones en la EEPROM, para contener la dirección bastaba con un registro de 8 bits: el EEADR. El dato leído o a escribir, de tamaño byte, se colocaba en el registro EEDATA.

Como en los PIC16F87X también se puede leer o escribir en la memoria FLASH y esta puede alcanzar un tamaño de 8 K de palabras de 14 bits cada una. No es suficiente con un solo registro para la dirección que alcanza los 13 bits, y lo mismo sucede para el dato, que tiene una longitud de 14 bits. Para cubrir esta necesidad el registro EEADR se liga al registro EEADRH (parte alta), que contiene los 5 bits de más peso de la dirección. El registro EEDATAH (parte alta) y se liga al registro EEDATA que contiene los 6 bits de más peso de la palabra leída o a escribir en la FLASH. Estos nuevos registros no se usan en operaciones que afectan a la EEPROM.

Para controlar la operación de lectura/escritura de las memorias EEPROM y FLASH hay dos registros denominados EECON1 y EECON2. El EECON1 ocupa la dirección 18Ch, mientras en el EECON2, como sucede con el PIC16F84, no está implementado físicamente y solo se usa en la delicada operación de escritura, que tiene una elevada duración de 2 milisegundos, aproximadamente.

Para evitar escrituras indeseadas en la memoria EEPROM motivadas por rebotes en la inicialización de l microcontrolador, se controla el bit WREN prohibiendo cualquier

operación de escritura mientras duran los 72 milisegundos que temporiza el timer de Power-up.

3.5. Puertos de entrada y salida E/S.

Entendemos como puertos de entrada/salida, las vías de comunicación o los medios de transmisión de datos, que tiene a su disposición el microcontrolador y que le permiten comunicarse con el exterior, es decir, con otros dispositivos periféricos.

Dependiendo de cada modelo de microcontrolador, ya sea el básico PIC16F84 que posee solamente entradas y salidas digitales; o la familia PIC16F87X que contiene líneas de entrada/salida con convertidor analógico-digital.

En el caso del microcontrolador PIC16F84 disponemos de dos puertos de comunicación entrada/salida definidos como PORTA y PORTB. Por el contrario, en el caso de la familia 16F87X dependiendo el modelo, en los de 28 pines se disponen de tres puertos; y en los modelos de 40 pines se disponen de 5 puertos.

Los puertos de E/S son tratados igual que cualquier otro registro, y por tanto podemos leer su contenido o bien escribir en ellos la información que deseemos. Estos procesos de lectura y escritura se realizan mediante las instrucciones adecuadas del programa.

Cada puerto lleva asociado, para llevar a cabo su control un registro (denominados TRISA, TRISB, TRISC, TRISD, TRISE, etc.) y cuando se produce un RESET todos los bits de dichos registros se ponen a nivel lógico alto (1 lógico), situándose entonces los puertos como entradas.

3.5.1. Los registros TRIS.

Como sabemos, cada puerto del PIC, lleva asociado un registro especial (TRIS) que utilizamos para cargar en él el código adecuado que nos permita programar el puerto correspondiente de acuerdo a nuestras necesidades.

Estos registros son controlados mediante software ejecutando la instrucción " TRIS f ", mediante la cual cargamos el registro TRIS con el contenido del registro W.

Si colocamos un 1 lógico en un determinado bit de un registro TRIS, nos determinará que la correspondiente línea de salida del puerto implicado se coloque en alta impedancia (modo entrada). Si colocamos un 0 lógico, la línea del puerto correspondiente trabajará como salida.

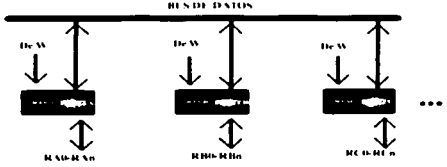


Figura 3.3 Registro TRIS

En la figura podemos ver la dependencia directa de los puertos de sus registros TRIS correspondientes y la dependencia de éstos del contenido del registro de trabajo W.

Cuando se realiza una operación de lectura, el puerto nos proporciona el dato presente en ese instante en las líneas físicas de entrada, de modo que si esas líneas cambian de valor un instante mas tarde, no lo apreciaremos a no ser que efectuemos otra nueva lectura. Sin embargo, cuando tenemos programada una línea como salida y escribimos en ella un dato, esta línea permanece de forma indefinida con el dato almacenado, hasta que en otra posterior operación de escritura nosotros lo cambiemos a voluntad.

Los puertos de estos modelos de microcontroladores los podemos programar ya sea como entradas o salidas independiente uno del otro; o podemos activar una función determinada, que se comporte como entrada analógica o digital.

3.5.2. Programación de los Puertos.

Como se ha mencionado anteriormente, los puertos se programan mediante el software del programa, utilizando para ello los registros TRIS.
En ocasiones se realizan algunas operaciones internas que llevan a cabo tareas de lectura seguida de escritura en los puertos. Por ejemplo, con las instrucciones BCF y BSF, se pueden leer todos los bits de un puerto, se realiza la operación con el bit indicado en la operación y finalmente se escribe el resultado. Estas operaciones deben realizarse con sumo cuidado cuando se trabaja con puertos cuyos bits están programados unos como entradas y otros como salidas.

Si realizamos una operación BSF sobre el bit 5 del PORTB, se producirá la lectura de los 8 bits del PORTB, a continuación el bit 5 se pone a 1 lógico y finalmente el valor del PORTB será escrito en los latches de salida. Si cualquier otro bit del PORTB se emplea en sentido bidireccional y está definido como entrada en ese instante, la señal de entrada presente en ese pin será leída por la CPU y reescrita en ese mismo pin. Como este pin permanece programado en modo entrada, no sucede nada anómalo. Sin embargo, si al realizar la operación BSF sobre el bit 5, cualquier otro bit está programado como salida, el contenido del latch de datos puede producir errores.

Mediante las instrucciones MOVF y MOVWF podemos leer y escribir respectivamente los puertos de los microcontroladores PIC.

Cuando se realizan una operación de lectura y otra de escritura seguidas, es recomendable dejar transcurrir unos instantes entre ambas operaciones, colocando una instrucción NOP entre las dos anteriores.

3.6. Otras características especiales.

Las características que conforman estos tipos de microcontroladores son entre otras cosas: disponen de circuitería especial que les permiten operar en tiempo real, además les permite optimizar los sistemas y minimizar los costes a través de la eliminación de componentes externos y permitiendo la inclusión de códigos de protección. Estas características pueden enumerarse del modo siguiente:

- Posibilidad de selección del oscilador del sistema.
- Módulo de Reset.
- Reset en la conexión de la alimentación (POR =Power-on Reset).
- Device Reset Timer (DRT).
- Watchdog Timer (WDT).
- Sleep (modo reposo).
- Código de protección.

3.6.1. Temporizadores.

Todos los dispositivos cuyo destino final o misión principal es controlar tiempos, reciben el nombre de timers o temporizadores. En realidad no son otra cosa que contadores ascendentes o descendentes que emiten una señal cuando se producen un determinado número de eventos.

Para realizar estas operaciones, se dispone de dos temporizadores básicos, el TMR y el WDT. El primero de ellos es el temporizador principal del sistema y se encarga de llevar a cabo el control de tiempos del sistema. El segundo (WDT = Perro guardián) se encarga de controlar el correcto funcionamiento del sistema impidiendo que se quede colgado (por ejemplo en la ejecución de un bucle infinito).

Ambos temporizadores pueden necesitar trabajar con tiempos elevados, elevando la duración de los pulsos, para ello se emplean las divisiones de frecuencia que nos permitirán

alargar los impulsos de reloj. Con el TMR0 la división de frecuencia se realiza antes de ser aplicada la señal al timer. Con el WDT el divisor de frecuencia actúa después.

3.6.2. Palabra de configuración.

Para ser honesto, en mi caso me encontré en un dilema al no poder hacer funcionar al principio un microcontrolador PIC16F84 cargado con un programa sencillo que solamente encendía unos leds al ser activados por unos switch *push-button*. El problema no radicaba en el hecho de que el programa estuviese mal editado o tuviera algún error en las instrucciones. Al analizar la señal que debía generarse por el cristal de cuarzo funcionando como oscilador, se descubrió que no existía ninguna señal de reloj que hiciera funcionar al PIC. Después de revisar bien algunos textos encontré que había omitido una parte fundamental en la construcción del programa, la palabra de configuración del microcontrolador.
Se debían de configurar dos bits de la palabra para seleccionar el tipo de oscilador que se quisiera utilizar (resistivo-capacitivo, cristal de cuarzo, etc.).

Según la configuración de bits, podremos adoptar varias modalidades de programación. Dos bits de configuración se utilizan para seleccionar el oscilador y otro bit más se emplea para habilitar el WDT. Otra serie de bits de configuración se emplean para disponer el código de protección, sin embargo éstos cambian en función del componente de la familia de microcontroladores PIC que estemos empleando.

CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	PWRTE	WDTE	FOSC1	FOSC0
13													0

Figura 3.4 Palabra de configuración.

- **CP:** Bits de protección de la memoria de código.
  1: No protegida.

0: Protegida. El programa no se puede leer, evitando copias. Tampoco se puede sobrescribir. Además evita que pueda ser accedida la EEPROM de datos y, finalmente si se modifica el bit CP de 0 a 1, se borra completamente la EEPROM.

- **PWRTE:** Bit de permiso para el timer de conexión de alimentación.

  1: Desactivado.

  0: Activado.

- **WDTE:** Bit de permiso del Timer del perro guardián.

  1: Activado.

  0: Desactivado.

- **FOSC0, 1:** Tipo de oscilador.

FOSC1	FOSC0	TIPO
0	0	LP (Baja potencia. 35 a 250 Khz).
0	1	XT(Estándar. Hasta 4Mhz).
1	0	HS (Alta velocidad. Más de 4 Mhz).
1	1	RC (Resistencia – condensador).

Tabla 3.5

3.6.3. Proceso de reinicialización (RESET).

El proceso de *reset* en un microcontrolador o microprocesador, consiste en obligarle a abandonar la ejecución del programa en el punto en el que se encuentre para reiniciar su ejecución desde el principio. En el caso de los microcontroladores en nuestro estudio el *reset* puede llevarse a cabo de cinco modos diferentes:

1. Al conectarse la alimentación (Power-On Reset, también llamado POR).
2. Activando a nivel lógico 0 la patilla MCLR (Master Clear Reset) con procedimiento normal.
3. Activando la patilla MCLR durante el estado de reposo (Sleep).
4. Mediante la activación desde el Perro Guardián (WDT) en funcionamiento normal.
5. Mediante la activación desde el Perro Guardián (WDT) desde el estado de reposo (Sleep).

En la generación del *reset* se produce un retardo de 18 ms para dar tiempo a que se estabilice la tensión de alimentación y la frecuencia del oscilador principal. El temporizador que genera este retardo está controlado por un oscilador RC independiente.

El POR (POWER-ON RESET).

Los circuitos integrados pertenecientes a algunos tipos de PIC, disponen en su interior de la circuitería necesaria para generar un Reset en los instantes de conexión del microcontrolador a la alimentación. La aplicación práctica de esta característica, se consigue conectando a la tensión de alimentación la patilla **MCLR** .

Cuando el microcontrolador comienza a funcionar normalmente, trata de ajustar los parámetros de tensión, frecuencia, temperatura, etc., para asegurar un correcto funcionamiento. Si no encuentra los parámetros correctos, el microcontrolador debe generar la señal de reset hasta que se produzcan las condiciones necesarias.

En la figura se presenta un circuito exterior (circuito RC) que debe ser conectado a **MCLR** para conseguir el reset POR. Normalmente se emplea cuando la tensión nominal de alimentación se alcanza de forma lenta. El diodo empleado en el circuito sirve para facilitar la descarga del condensador cuando se suprime la alimentación. Se recomienda una resistencia R menor de 40 K para asegurar las características eléctricas. El valor de R1 debe oscilar entre 100 ohmios y 1 K.

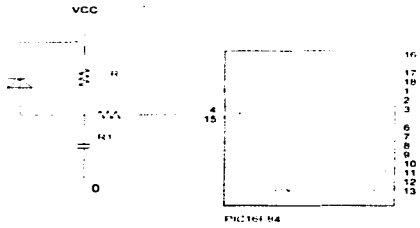


Figura 3.5

El DRT (DEVICE RESET TIMER).

El DRT (Device Reset Timer) proporciona un retardo nominal de 18 ms para las funciones de Reset. Este elemento (DRT) trabaja con su propio oscilador interno y se coloca en situación de actividad cuando se produce una situación de Reset. Su misión básica es conseguir la estabilidad de la tensión de alimentación permitiendo esperar a tener el menor rizado posible y alcanzar la estabilidad de la frecuencia de trabajo.

Los circuitos osciladores basados en cristales de cuarzo o en resonadores cerámicos, requieren cierto tiempo de retardo hasta alcanzar una oscilación estable.

El DRT mantiene la condición de reset durante 18 ms después de que el pin **MCLR** haya alcanzado el nivel lógico alto. Por tanto las redes externas RC de conexión en el pin **MCLR** no son necesarias en la mayoría de los casos, permitiendo un mayor ahorro y disponiendo de más espacio para las aplicaciones.

El WDT (WATCHDOG TIMER).

El Watchdog Timer (Perro Guardian) dispone de un oscilador interno RC que funciona de forma libre y no necesita componentes externos. Este oscilador continúa

funcionando aunque el oscilador principal se encuentre bloqueado, por ejemplo después de ejecutar una instrucción SLEEP.

En líneas generales podemos decir que el WDT es un contador interno de 8 bits que actúa como temporizador y su objetivo es generar un reset al sistema, cuando alcanza su valor máximo.

Cuando tiene lugar un Reset del WDT, se coloca a 0 el bit 4 (TO) del registro STATUS.

El WDT puede ser deshabilitado mediante la programación del bit WDTE con un 0 lógico.

El WDT posee una temporización inicial de 18 ms (sin emplear el divisor de frecuencia). Si en un momento dado necesitamos un periodo (o una temporización) mucho mayor, podemos asignar al divisor de frecuencias un valor tal que nos proporcione una frecuencia de hasta 1:128 veces menor mediante la escritura del valor adecuado en el registro OPTION.

El RESET BROWN-OUT.

Un Reset Brown-out es una condición producida porque la alimentación del componente (Vdd) baja o cae a un valor mínimo, sin llegar a cero, y posteriormente se recupera. El PIC se ve sometido a un Reset durante este proceso.

Para proteger de este problema a los componentes, se diseña un circuito exterior de protección, como el mostrado en la figura, que basado en un diodo zener y un transistor PNP, trata de eliminar esta circunstancia.

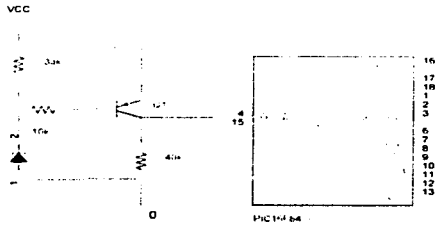


Figura 3.6.

3.6.4. El modo de bajo consumo (SLEEP).

Estos tipos de microcontroladores entran en el modo de trabajo de bajo consumo, cuando se ejecuta la instrucción SLEEP. En estas condiciones se producen las acciones siguientes:

a. Si está habilitado el WDT, su contenido será borrado iniciándose nuevamente su proceso de conteo.
b. El bit **TO** (bit 4 del registro STATUS) se activa (se pone a 1)
c. El bit **PD** (bit 3 de STATUS) se borra (se pone a 0).
d. El oscilador principal del sistema se bloquea.
e. Los puertos de E/S mantienen el mismo estado que tenían antes de iniciarse el modo SLEEP.

Debemos darnos cuenta que si en el WDT se produce un RESET por un exceso de tiempo, no se produciría un nivel bajo en la patilla **MCLR** .

Para disminuir el consumo al máximo es conveniente conectar la patilla TOCKI a Vdd o a masa, y la patilla **MCLR** a nivel lógico alto.

Para salir del estado de reposo (SLEEP) hay dos posibilidades:

1. Activar exteriormente la patilla **MCLR** y generar un reset.
2. El WDT que estaba activo cuando se ejecuto SLEEP, se desborda y genera un reset.

Los bits **PD** y **TO** pueden utilizarse para determinar la causa del reset.

3.6.5. El convertidor A/D en los PIC16F87X.

Una desventaja que se presenta en los "viejos" microcontroladores PIC16F84, es que se debe trabajar con señales completamente digitales para ser procesadas por el microcontrolador. Si se deseaba trabajar con señales en tiempo continuo era necesario implementar circuitería extra para darle tratamiento para hacer la conversión posterior en señales discretas.

Los microcontroladores PIC16F87X poseen un convertidor analógico – digital (ADC) de 10 bits de resolución. Los modelos 873/6 cuentan con 5 pines de entrada y los 874/7 tienen 8 pines.
La resolución que tiene cada bit procedente de la conversión tiene un valor que es función de la tensión de referencia *Vref*, de acuerdo con la relación siguiente:

$$\text{Resolución} = (Vref+ - Vref-)/1.024 = Vref/1.024$$

Por ejemplo: Si el Vref+ = 5 V y Vref- es tierra, la resolución es de 4.8 mV/bit. Por lo tanto, a la entrada analógica de 0 V le corresponde una digital de 00 0000 0000 y para la de 5 V una de 11 1111 1111. La tensión de referencia determina los límites máximo y mínimo de la tensión analógica que se puede convertir. El voltaje diferencial mínimo es de 2 V.

A través del canal de entrada seleccionado, se aplica la tensión analógica a un condensador de muestreo y retención y luego se introduce al convertidor, el cual

proporciona un resultado de 10 bits de resolución usando la técnica de aproximaciones sucesivas.

Registros que manejan el convertidor analógico – digital.

El funcionamiento del ADC requiere la manipulación de varios registros:

- **ADRESH:** Parte alta del resultado de la conversión.
- **ADRESL:** Parte baja del resultado de la conversión.
- **ADCON0:** Registro de control 0.
- **ADCON1:** Registro de control 1.

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

El registro *ADCON0* controla la operación de AD, mientras que el *ADCON1* sirve para configurar los pines del puerto A como entradas analógicas o como E/S digitales.

Figura 3.7 Configuración de bits para los registros ADCON0 y ADCON1.

Los bits 7 y 6 de registro ADCON0 sirven para seleccionar la frecuencia de reloj que se emplea en la conversión, con la siguiente asignación:

ADCS1:0	FRECUENCIA
00	$F_{OSC}/2$
01	$F_{OSC}/8$
10	$F_{OSC}/32$
11	F_{RC} (Procede del oscilador interno)

Tabla 3.6

El bit GO/DONE es el que indica el estado de conversión. Poniéndolo a 1 se inicia la conversión y mientras esté a 1 está realizándose dicha operación. Cuando este bit se pone a 0 confirma el final de la conversión y la puesta del resultado en la pareja de registros ADRESH:L.
El bit ADON sirve para activar el ADC poniéndolo a 1 y para inhibir su funcionamiento poniéndolo a 0.

ADFM del registro ADCON1 selecciona el formato del resultado de la conversión. Si vale 1, el resultado está justificado en el registro ADRESH, que tiene sus 6 bits de más peso a 0; mientras que si vale 0 la justificación se realiza sobre el registro ADRESL, que tiene sus 6 bits de menos peso a 0.

Los bits PCFG3 – 0 de ADCON1 se usan para configurar los pines de los canales de entrada al convertidor como analógicas o E/S digitales.

CAPÍTULO 4

REPERTORIO DE INSTRUCCIONES

REPERTORIO DE INSTRUCCIONES.

4.1. Introducción.

En el estudio que nos ocupa sobre los microcontroladores PIC16F84 y PIC16F87X, debemos considerar la parte más importante que estos circuitos, el cual es su lenguaje y las instrucciones que conforman su repertorio. Tengo la obligación de mencionar que el repertorio de ambas familias de PIC es prácticamente el mismo. Las instrucciones de los PIC están formadas por palabras de 12 bits que pueden ser ejecutadas por los microcontroladores.

Cuando, en la terminología empleada, usamos la letra "f" nos referimos a un registro genérico, mientras que la letra "d" representa el destino del resultado de la instrucción que vaya a ejecutarse. El registro f especificará uno de los 32 registros de los bancos del PIC.

En las instrucciones orientadas a trabajar con bits, la letra "b" indica el bit que será afectado por la operación, mientras "f" representa el registro en el que se encuentra dicho bit.

En las instrucciones de control y las que trabajan con valores numéricos, la letra "k" representa una constante o un valor literal de 8 ó 9 bits.

Todas las instrucciones se ejecutan durante un ciclo simple, a menos que una determinada condición sea cierta o bien el contador de programa resulte alterado al ejecutar una instrucción. En este caso, la ejecución se realiza durante dos ciclos de instrucción. (Como sabemos, un ciclo de instrucción consta de cuatro períodos de oscilador). Así, para una frecuencia de oscilador de 4 MHz, el tiempo normal de ejecución de una instrucción sería 1 µs. Si la ejecución es de dos ciclos se emplean 2 µs.

Lista de palabras empleadas en la nomenclatura:

f	Localización de memoria en el microcontrolador.
W	Registro de trabajo.
b	Posición de un bit en el registro "f".
k	Valor literal, constante o etiqueta
x	Valor desconocido (0 ó 1)
d	Bit de destino.
label	Etiqueta.
TOS	Parte alta de la pila
PC	Contador de programa *(program counter)*.
WDT	Perro guardián.

En las siguientes tablas se muestran el repertorio de instrucciones para los microcontroladores de las familias 16X84 y 16F87X.

Nemónicos	Parámetros	Operación.	Ciclos	Formato 14 bits	Señalizadores
addwf	f,d	Suma de W con f	1	00 0111 dfff ffff	C,DC,Z
andwf	f,d	AND de W con f	1	00 0101 dfff ffff	Z
clrf	f	Borrado de f	1	00 0001 1fff ffff	Z
clrw	ae	Borrado de W	1	000001 0xxxxxxx	Z
comf	f,d	Complemento de f	1	00 1001 dfff ffff	Z
decf	f,d	Decremento de f	1	00 0011 dfff ffff	Z
incf	f,d	Incremento de f	1	00 1010 dfff ffff	Z
iorwf	f,d	OR de W con f	1	00 0100 dfff ffff	Z
movf	f,d	Movimiento de f	1	00 1000 dfff ffff	Z
movwf	f	Movimiento de W a f	1	00 0000 1fff ffff	
nop	ae	No operación	1	0000000xx00000	
rlf	f,d	Rotación de f a izquierda con carry	1	00 1101 dfff ffff	C
rrf	f,d	Rotación de f a derecha con carry	1	00 1100 dfff ffff	C
subwf	f,d	Resta de W a f	1	00 0010 dfff ffff	C,DC,Z
swapf	f,d	Intercambio de bits	1	00 1110 dfff ffff	
xorwf	f,d	Orex de W con f	1	00 0110 dfff ffff	Z
bcf	f,b	Puesta a 0 del bit b de f	1	01 00bb bfff ffff	
bsf	f,b	Puesta a 1 del bit b de f	1	01 01bb bfff ffff	
btfsc	f,b	Testeo del bit b de f, brinco si 0	2	01 10bb bfff ffff	
btfss	f,b	Testeo del bit b de f, brinco si 1	2	01 11bb bfff ffff	
decfsz	f,d	Decremento de f	2	00 1011 dfff ffff	
incfsz	f,d	Incremento de f	2	00 1111 dfff ffff	
addlw	k	Suma de literal con W	1	11 111x kkkkkkkk	C,DC,Z
andlw	k	AND de literal con W	1	11 1001 kkkkkkkk	Z
iorlw	k	OR de literal con W	1	11 0000 kkkkkkkk	Z
movlw	k	Movim. de literal a W	1	11 00xx kkkkkkkk	
sublw	k	Resta W de literal	1	11 110x kkkkkkkk	C,DC,Z
xorlw	k	OREX de literal con W	1	11 1010 kkkkkkkk	Z
call	k	Llamada a subrutina	2	10 0kkk kkkkkkkk	
clrwdt		Borrado del perro guardián	1	0000000100100	#TO, #PD
goto	k	Salto a una dirección	1	10 1kkk kkkkkkkk	
retfie		Retorno de interrupción	2	0000000001001	
retlw	k	Retorno devolviendo literal en W	2	11 01xx kkkkkkkk	
return		Retorno de subrutina	2	0000000001000	
sleep		Puesta del microprocesador en reposo	1	0000000100011	#TO, #PD

Tabla 4.1 Instrucciones de los microcontroladores PIC16X8X y PIC16F87X

4.2. Análisis de las instrucciones.

4.2.1. Instrucciones que manejan registros.

ADDWF

Suma el contenido del registro W al contenido del registro f, y almacena el resultado en W si d = 0, y en el registro f si d = 1. Si se produce acarreo la bandera C se pone a "1". Si se genera un acarreo del bit 3 al 4 la bandera DC se pone a 1. Si el resultado de la operación es cero, la bandera Z se pone a 1.

ANDWF

Efectúa la operación AND lógica entre el contenido del registro W y el contenido del registro f. Almacena el resultado en W si d=0, y en f si d=1. Afecta la bandera Z.

CLRF

Coloca a cero el contenido del registro f y afecta a la bandera Z que se coloca a 1 lógico.

CLRW

Coloca a cero el contenido del registro W (acumulador) y afecta a la bandera Z que se coloca a 1 lógico.

COMF

Complementa el contenido de registro f. Si d=0 el resultado se almacena en W, y si d=1 el resultado se almacena en f. Afecta a la bandera Z.

DECF

Decrementa (en 1) el contenido del registro f. Si d=0 el resultado se almacena en W. y si d=1 el resultado se almacena en f.

INCF

Se incrementa el contenido del registro f en una unidad. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. Afecta a la bandera Z.

IORWF

Realiza una operación OR entre el contenido del registro W y el contenido del registro f. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. Afecta a la bandera Z.

MOVF

El contenido del registro f se copia en el registro señalado por d. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en f.

MOVWF

Copia el contenido del registro W en el registro f. No altera ningún flag.

NOP

NO realiza ninguna operación y no afecta a ninguna bandera.

RLF

El contenido del registro f rota hacia la izquierda 1 bit pasando su bit de mayor peso al Carry, y el bit que había en el Carry pasa al bit de menor peso del registro f. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. Afecta a la bandera C.

RRF

El contenido del registro f rota hacia la derecha 1 bit pasando su bit de menor peso al Carry, y el bit que había en el Carry pasa al bit de mayor peso del registro f. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. Afecta a la bandera C.

SUBWF

Resta el contenido de los registros f y W por el método del complemento a dos. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. Afecta a las banderas C, DC y Z.

SWAPF

Se intercambian los cuatro bits de mas peso con los cuatro bits de menos peso del registro f. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. No afecta a ninguna bandera.

XORWF

Se realiza una operación OR exclusiva entre el contenido del registro W y el contenido del registro f. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. Afecta al bandera Z.

4.2.2. Instrucciones que manejan bits.

BCF

Coloca a cero el bit número b del registro f. No afecta a ninguna bandera de estado.

BSF

Coloca a uno el bit número b del registro f. No afecta a ninguna bandera de estado

4.2.3. Instrucciones de salto.

BTFSS

Si el bit número b del registro f está a 1, la instrucción que sigue a ésta se ignora y se trata como un NOP (*no operation*). En este caso, y sólo en este caso, la instrucción BTFSS precisa dos ciclos para ejecutarse.

BTFSC

Si el bit número b del registro f está a 0, la instrucción que sigue a ésta se ignora y se trata como un NOP (*no operation*). En este caso, y sólo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.

DECFSZ

Se decrementa el contenido del registro f. Si d=0 el resultado se almacena en el registro W y f no varía. Si d=1 el resultado se almacena en el registro f. Si el resultado es cero, se ignora la siguiente instrucción (salta una instrucción) y, en ese caso la instrucción tiene una duración de dos ciclos.

INCFSZ

Se incrementa el contenido del registro f en una unidad. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro f. Si el resultado es cero, se ignora la siguiente instrucción (salta una instrucción) y, en ese caso la instrucción tiene una duración de dos ciclos.

4.2.4. Instrucciones que manejan operandos inmediatos.

ADDLW

ANDLW

Efectúa la operación AND lógico entre el contenido del registro W y el literal k, y almacena el resultado en W. Afecta la bandera Z.

IORLW

Realiza una operación OR lógica ente el contenido del registro W y el valor del literal k. El resultado se almacena en el registro W. Afecta a la bandera Z.

MOVLW

El valor del literal k se almacena en el registro W. No afecta a ninguna bandera.

SUBLW

XORLW

Se realiza una operación XOR entre el contenido de W y el valor del literal k. El resultado se almacena en W. Afecta a la bandera Z.

4.2.5. Instrucciones de control y especiales.

CALL

Instrucción de llamada a subrutina. En primer lugar incrementa el PC (*contador de programa*) en 1 y almacena dicho valor en la cima de la pila. A continuación carga en los 8 bits menos significativos del PC el valor del literal k. Los dos bits más significativos del PC (bits 10 y 9) se cargan con el valor de los bits 6 y 5 del registro de estado respectivamente (STATUS). Finalmente el bit 8 del PC se pone a cero. Esta instrucción emplea dos ciclos en su ejecución. No afecta a ninguna bandera de estado.

CLRWDT

La instrucción CLRWDT resetea el WDT. Se borra tanto el registro WDT (Watchdog) como su preescalar. Los bits T0 y PD del registro de estado se ponen a "1".

GOTO

Se trata de un salto incondicional. Los nueve bits menos significativos del PC son cargados con el valor de k. Los bits 10 y 9 del PC son cargados con el valor de los bits 6 y 5 de STATUS, respectivamente. Emplea dos ciclos de instrucción y no afecta a ninguna bandera de estado.

RETFIE

RETLW

El valor del literal k se almacena en el registro W y el contador de programa (PC) se carga con el valor que tiene la parte alta de la pila. Emplea dos ciclos de instrucción y no afecta a ninguna bandera.

RETURN

SLEEP

El WDT y su preescalar se ponen a cero. Pone al circuito en modo Sleep (bajo consumo) con parada del oscilador. Pone a 0 el flag PD (Power Down) y el flag TO (Timer Out) se pone a 1. Se puede salir de este estado por: 1. Activación de MCLR para provocar un Reset. 2. Desbordamiento del Watchdog si quedó operativo en el modo reposo 3. Generación de una interrupción que no sea TMR0 ya que ésta se desactiva con la instrucción SLEEP.

CAPÍTULO 5

MPLAB

MPLAB.

5.1. Introducción al MPLAB IDE.

La empresa Microchip nos presenta MPLAB como una herramienta de trabajo, que distribuye de forma gratuita desde su página web, con todo tipo de información y ayuda.

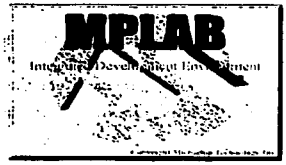


Figura 5.1

MPLAB es un sistema de desarrollo integrado (IDE, Integrated Development Environment) para la tecnología de Microchip, que permite desarrollar sistemas basados en sus microcontroladores.

MPLAB es un programa para ambiente Windows que hace posible crear y desarrollar un programa fácilmente.

MPLAB consiste de diversas e importantes partes:

- Agrupa los archivos dentro de un proyecto (Project Manager).
- Genera y procesa un programa (Text Editor).
- Simulador de programas.

Así, este software, también soporta sistemas para productos de Microchip como son el *PicStart Plus* (programador de dispositivos de fabricación de la propia empresa) y el *ICD (In Circuit Debugger)*.

La organización de herramientas de MPLAB por funciones, con menús desplegables, ayuda en las tareas de programación y facilita el trabajo permitiendo, entre otras cosas:

- Ensamblar, compilar y ligar el código fuente.
- Agrupar diferentes archivos en un proyecto común.
- Ejecutar el programa desarrollado a través del simulador o en tiempo real mediante el empleo del emulador MPLAB-ICE.
- Realizar las medidas oportunas.
- Inspeccionar la evolución de las variables del programa mediante ventanas.
- Inspeccionar variables en ventanas de Reloj.

MPLAB constituye una rica herramienta de trabajo que nos permitirá abrir una serie de ventanas, en las que podemos examinar de forma detallada la evolución de nuestro programa durante su ejecución. Podremos visualizar todos y cada uno de los registros importantes del microcontrolador con el que deseemos trabajar y la forma en que varían en función del punto de programa en el que nos encontremos, pudiendo ejecutarlo paso a paso o de forma continua, tendremos acceso al listado del programa y veremos la evolución del mismo, podremos visualizar un reloj que nos indicará el número de ciclos realizados por nuestro programas y el tiempo empleado en su ejecución. Podremos establecer según nuestras conveniencias, puntos de ruptura o de control para efectuar los análisis que consideremos convenientes, etc.

El MPLAB IDE incorpora un editor de textos que nos permite crear nuestros propios programas (nosotros lo haremos en lenguaje ensamblador). Este editor no funciona de forma independiente a MPLAB IDE, sino que se accede a él desde el sistema de desarrollo integrado.

La instalación se efectúa básicamente como cualquier otro programa en el ambiente Windows, donde aparece la ventana de instalación siguiente:

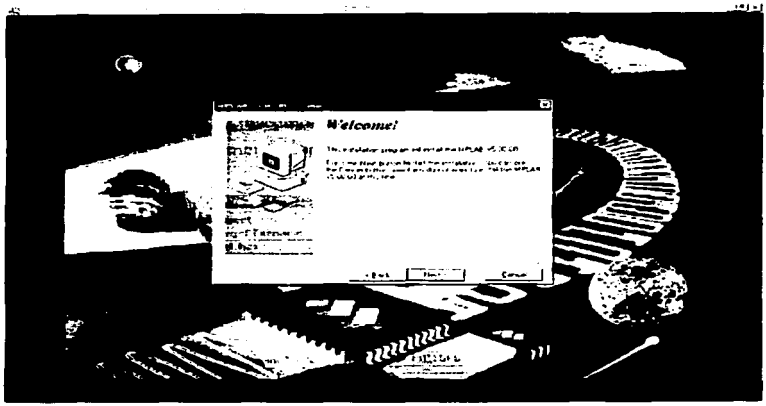


Figura 5.2 Entorno de instalación del MPLAB

5.2. Iniciación de proyectos en el MPLAB.

Se dará a continuación de forma breve, aunque lo más clara posible, la forma de iniciar un proyecto nuevo y la forma de abrir un proyecto ya existente con MPLAB.

En primer lugar a arrancar el programa MPLAB, haciendo doble click sobre el icono que lo representa, tras lo cual el programa se iniciará y se presentará la pantalla de nuestro ordenador en blanco, quedando accesible, en la parte superior de la pantalla, la barra de funciones que nos permitirá el acceso a todas las posibles tareas que podamos llevar a cabo con MPLAB. Inmediatamente debajo de esta barra, aparece una segunda barra con un conjunto de iconos, que no son más que accesos directos de determinadas funciones, que nos permitirán agilizar nuestro trabajo.

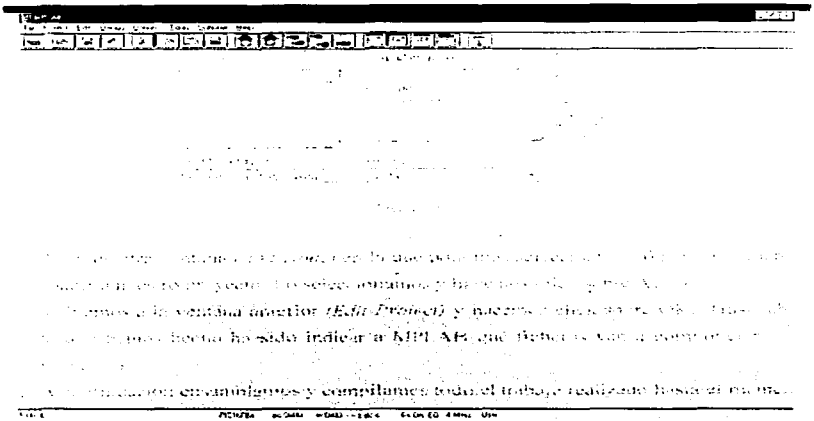


Figura 5.3 Aspecto de la ventana de inicio.

Para iniciar un nuevo proyecto en el que trabajar, se procede del modo siguiente:

1. En la barra de funciones se elige *Options*, haciendo click con el puntero del ratón.
2. En la ventana desplegada que aparece elegimos *Development Mode*, con lo que aparecerá una nueva ventana de diálogo, con diversas pestañas para seleccionar.
3. Se abre la pestaña *Tools* y activamos *MPLAB-SIM* (simulador). En el espacio que señala *Processor* hacemos click sobre la flecha para desplegar una lista con las denominaciones de los diferentes procesadores de Microchip y seleccionamos el modelo a usar.
4. Se hace click sobre *OK*.
5. En este momento se abre la pantalla de trabajo limpia y únicamente se ha indicado al MPLAB el modo de trabajo que se va a manejar. Hacemos ahora click sobre *Projects*.
6. En el menú desplegado, hacer click sobre *New Projects*.

7. Aparece ahora una ventana que muestra por un lado el directorio en el que nos encontramos y la carpeta seleccionada para almacenar el nuevo proyecto (podemos elegir otra) y un espacio denominado *File Name* en el que se debe introducir el nombre al nuevo proyecto, con la extensión "pjt".
8. Introducido el nombre, hacemos click sobre OK.
9. Aparece ahora otra nueva ventana con una serie de espacios que nos indicarán entre otras cosas el modo de trabajo elegido, dando además la opción de poder cambiarlo
10. En este momento ya es posible trabajar sobre el nuevo proyecto. Si deseamos guardar el proceso realizado hasta el momento, tendremos que hacer click sobre *Project* y a continuación accionar *Save Project* en el menú desplegado.

Si se dispone de un proyecto, que ya ha sido iniciado o incluso finalizado, y deseamos abrirlo para realizar en él cualquier modificación, procederemos del modo siguiente:

1. Se hace click sobre la función *Project* en la barra de funciones. Se abre entonces una ventana en la que aparecen diversas opciones.
2. Se hace click sobre la opción *Open Project* y aparece ahora otra ventana dividida en dos partes claramente diferenciadas.
3. Sobre la parte derecha de dicha ventana seleccionamos la carpeta en la que se encuentra el archivo correspondiente al proyecto que deseamos abrir.
4. Sobre la parte izquierda de la misma ventana aparecen los nombres de los diferentes proyectos que tengamos disponibles en la carpeta seleccionada. Entre ellos estará el proyecto que buscamos. Lo seleccionamos haciendo click sobre su nombre.
5. A continuación aceptamos la opción señalada haciendo click sobre OK.

En este instante estaremos en condiciones de realizar las modificaciones que estimemos oportunas en nuestro proyecto seleccionado.

5.2.1. Crear un fichero con MPLAB.

Una vez iniciado o seleccionado el proyecto en el que debemos trabajar, se procede a escribir las líneas, en ensamblador, de las diferentes partes que constituirán el programa que pretendemos realizar.

Para esto, MPLAB dispone de un editor que nos permite de una forma muy sencilla, llevar a cabo dicho trabajo.

Los ficheros generados por el programa editor, en nuestro caso se realizarán en ensamblador y por tanto llevarán siempre la extensión ".asm". Para generar un fichero de estas características se procederá del modo siguiente:

1. Se abre el proyecto en el que vamos a incluir el fichero que deseamos generar.
2. Se hace click sobre la función *File* de la barra de funciones.
3. Sobre el menú desplegado seleccionamos la opción *New*. Aparece una nueva ventana (en blanco) en la que ya podemos escribir el programa en ensamblador.

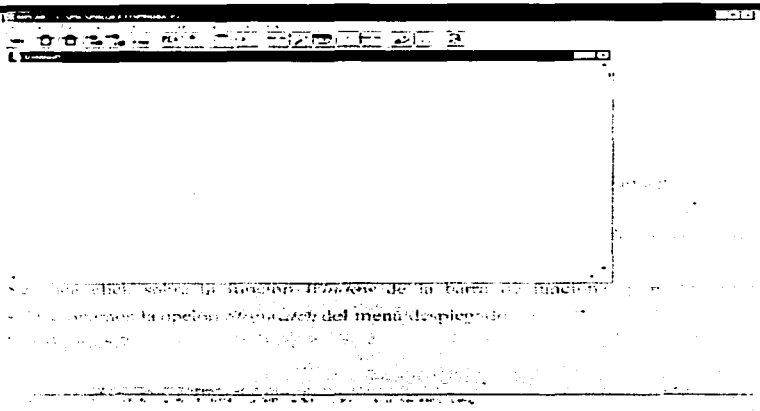


Figura 5.4 Apariencia del editor.

4. Escribimos con el editor.
5. Una vez escrito lo se guarda en la carpeta donde deseemos almacenar todos los ficheros que compondrán el proyecto. Para ello, hacemos click sobre *File* y a continuación, en el menú desplegado, elegimos la opción *Save As*.
6. Aparece una nueva ventana que permite seleccionar la carpeta de almacenamiento y a continuación se da el nombre al nuevo fichero en la casilla denominada *File Name* (siempre con la extensión ".asm").
7. Finalmente hacemos clik sobre *OK*.

5.3. Ensamblar y compilar en MPLAB.

Una vez que ha sido editado en lenguaje ensamblador el programa, debemos llevar a cabo las operaciones necesarias para convertirlo en ejecutable, de modo que lo podamos hacer funcionar en el simulador de MPLAB.

Los pasos a seguir son los siguientes:

1. Los primeros pasos a seguir son siempre, seleccionar el proyecto que ha de ejecutarse, escribir en ensamblador los ficheros que van a constituir nuestro proyecto y guardarlos.
2. A continuación indicar a MPLAB qué ficheros van a pertenecer al proyecto seleccionado. Para ello, hacemos click sobre *Project* y a continuación elegimos la opción *Edit Project* del menú desplegado.

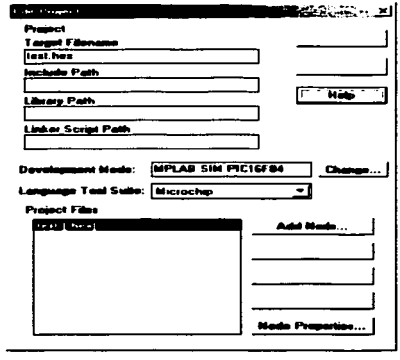


Figura 5.5

3. El software muestra la ventana *Edit Project* en la que aparecen una serie de casillas. Seleccionamos aquí el nombre del proyecto (ahora con extensión ".hex") y a continuación hacemos click sobre *Add Node*.

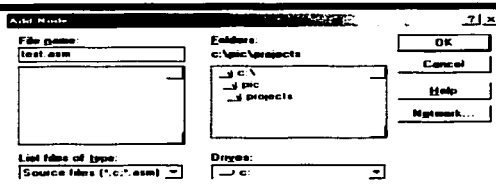


Figura 5.6

4. Aparece otra ventana *(Add Node)* en la que podemos seleccionar el fichero o ficheros a añadir a nuestro proyecto. Lo seleccionamos y hacemos click sobre Aceptar.
5. Volvemos a la ventana anterior *(Edit Project)* y hacemos click sobre OK. (Hasta ahora lo que hemos hecho ha sido indicar a MPLAB qué ficheros van a componer nuestro proyecto).
6. A continuación ensamblamos y compilamos todo el trabajo realizado hasta el momento. Para ello hacemos click sobre *Project* y a continuación sobre *Built All*. (Si todos los pasos se han desarrollado correctamente, MPLAB indicará que los procesos se han realizado de forma satisfactoria y ya estaremos en condiciones de ejecutar nuestro programa en el simulador de MPLAB. Si nos indica que se ha producido algún error, tendremos que volver al punto erróneo y repetir el proceso anterior).

5.3.1. Ejecución del programa.

Realizadas ya de forma correcta todas las operaciones indicadas en las fases anteriores, no queda más que ejecutar el programa para comprobar que se ejecuta de manera correcta.

Existen diversos métodos para ejecutar el programa en MPLAB.

Para ello se realizan los siguientes pasos:

1. Se selecciona la función *Debug* de la barra de funciones.
2. Hacemos click sobre la opción *Run* del menú desplegado.
3. Y se selecciona *Step* en el submenú que aparece.
4. MPLAB presenta una ventana con el programa en ensamblador y muestra una línea resaltada, indicando que el programa se está ejecutando en esa línea. Si ahora hacemos click con el puntero del ratón sobre la casilla *Step* (marcada por dos pisadas en la línea de iconos que está debajo de la línea de funciones), veremos que la ejecución del programa avanza hasta otra línea y así cada vez que hagamos click sobre el icono *Step*. Esta es la ejecución del programa paso a paso y en ella podremos observar de forma precisa si nuestro programa se ejecuta correctamente o si por el contrario realiza alguna acción no deseada.
5. Cuando lleguemos al final de la ejecución del programa la acción *Step* se detiene.

En la figura 5.7 se puede observar la ejecución de un programa ya ensamblado y compilado, la línea que aparece en un recuadro de color negro, indica que el programa esta siendo ejecutado en ese instante.

5.4. Como visualizar registros especiales.

En la ejecución paso a paso, tal se ha realizado en el apartado anterior, se ve la evolución del programa instrucción a instrucción, pero no podemos apreciar el efecto que dicha ejecución tiene sobre los principales registros del microcontrolador. MPLAB dispone de una opción que nos permite visualizar estos parámetros.

Se deben de ejecutar los siguientes pasos:

1. Seleccionar el proyecto, escribir sus ficheros en ensamblador, convertirlo en ejecutable y pasar a la ejecución paso a paso.
2. Hacer click sobre la función *Window* de la barra de funciones y seleccionamos la opción *Special Function Registers* del menú desplegado.

3. Aparece una ventana en la que nos presenta todos los registros especiales del microcontrolador y el contenido actual de cada uno. Esta ventana podemos desplazarla a la posición de la pantalla que nos interese sin más que seleccionarla con el puntero del ratón y arrastrar sin soltar.

El aspecto que puede presentar la ventana de registros especiales, es similar a la mostrada a continuación en la tabla En ella podemos apreciar una primera columna encabezada por *SFR Name*, en la que nos muestra los nombres de todos los registros especiales. En la segunda columna nos muestra el contenido de cada uno de los registros expresado en hexadecimal. En la columna número tres tenemos los mismos valores que en la anterior, pero expresados en decimal y finalmente en la cuarta columna, nos presenta el mismo dato, pero ahora expresado en binario.

Observando esta ventana de registros especiales, veremos que a medida que vamos ejecutando paso a paso nuestro programa, el contenido de dichos registros va variando en función de las instrucciones ejecutadas. Esto nos permite observar la evolución de determinados registros a lo largo del programa y comprobar si se obtienen los resultados deseados.

Ver figura 5.7 y 5.8.

5.4.1. Como visualizar todos los registros.

Al igual que en el caso anterior, en primer lugar debemos poner en marcha nuestro programa en modo de trabajo paso a paso. Si ahora deseamos visualizar el contenido de todos los registros de nuestro microcontrolador, a medida que se va ejecutando el programa, debemos proceder del modo siguiente:

1. Hacer doble click sobre la función Window de la barra de funciones y a continuación seleccionar la opción File Registers del menú desplegado.
2. Veremos aparecer una ventana en la que se mostrará el contenido de todos los registros, los especiales y los de trabajo normal.

Ver figura 5.7.

5.4.2. Como visualizar el reloj (Clock).

Una opción interesante, en el momento de depurar programas y mejorarlos, es poder visualizar un reloj que nos permita controlar el número de ciclos que se invierten en llevar a cabo una parte de un programa o una subrutina, y el tiempo que se emplea en ejecutarla.

Para esto, MPLAB dispone de una herramienta a la que se accede del modo siguiente:

1. Se selecciona nuestro proyecto, escribimos sus ficheros en ensamblador, se convierte en ejecutable y se pasa a la ejecución paso a paso.
2. Se hace click sobre la función *Window* de la barra de funciones y posteriormente seleccionamos la opción *Stopwatch* del menú desplegado
   Ver figura 5.7.

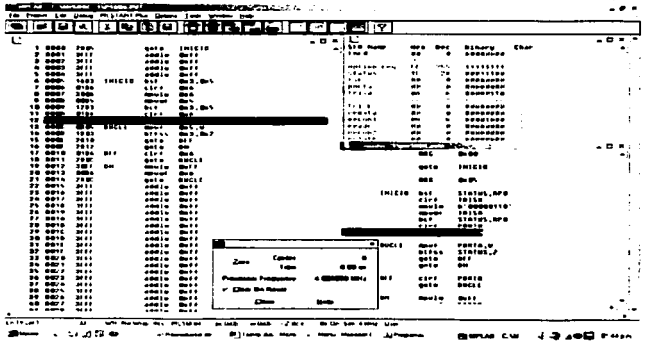


Figura 5.7 Vista de la ejecución paso a paso en un programa, y su desplazamiento a través de los registros, así como las condiciones del reloj

3. Se ve una ventana en la que se nos muestran el número de ciclos empleados en la ejecución, el tiempo invertido en los mismos y la frecuencia de trabajo del microcontrolador empleado.

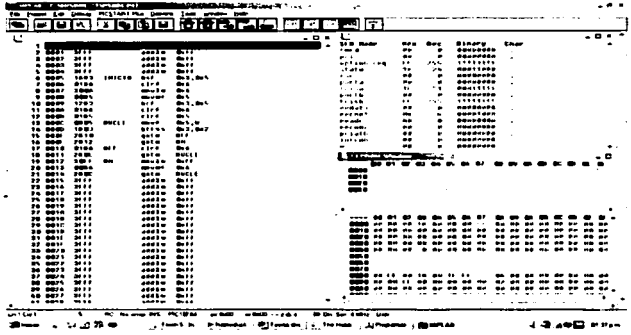


Figura 5.8 Vista de la memoria EEPROM de Datos y Registros especiales, de un programa ensamblado en el MPLAB.

CAPÍTULO 6

PRÁCTICAS

PRÁCTICAS.

6.1. Práctica No.1. El programador de PIC's.

Objetivos.

- Construir un programador sencillo para los modelos de microcontroladores PIC 16F84 y 16F87X (usados en las prácticas) conociendo el protocolo y tensiones de programación del circuito
- Optimizar recursos recurriendo a dispositivos de muy bajo costo para la construcción del programador.

Materiales.

- Un conector DB25 macho.
- 7 Transistores de efecto de campo (FET) BS170.
- 3 Resistencias de 470 KΩ.
- 6 Resistencias de 1 KΩ.
- 3 Resistencias de 820 Ω.
- 3 Capacitores de .1 μF.
- 1 Diodo 1N4148.
- 3 Diodos emisores de luz.
- Bases para circuitos integrados de 18, 28 y 40 pines (puede ser usado una base ZIF).

Para la fuente de alimentación.

- 1 Regulador de tensión 7805.
- 1 Regulador de tensión 78012.
- 1 Resistencia de 10 KΩ.
- 1 Resistencia de 820 Ω.
- 1 Capacitor de 100 μF.

- 1 Capacitor de 47µF.
- 2 Capacitores de 10 µF.
- 4 Capacitores de .1µF.
- 1 Diodo 1N4148.
- 1 Diodo emisor de luz.

Uno de los graves problemas con los que se enfrenta la mayoría de los estudiantes al elaborar sus tesis relacionadas con proyectos en los que se debe se disponer de equipo para la elaboración de los mismos, es el aspecto económico. Generalmente en estos trabajos se buscan temas relevantes, más en la carrera de Ingeniería.

En el caso de esta tesis, el elevado costo no se refleja en los circuitos (microcontroladores), ya que son relativamente baratos, ni en el software que es proporcionado por la empresa Microchip a través de su página de internet gratuitamente; sin embargo, tarjetas de desarrollo y el propio programador de chip tienen un precio alto.

De lo anterior surge la necesidad de buscar nuevas fuentes, para elaborar uno mismo un circuito programador de los de tipo casero, adaptables a la PC, de elaboración sencilla, baratos, confiables y sobretodo eficientes en su desempeño.

Después de investigar en diversas fuentes, y tomando en cuenta los modelos de microcontroladores que había decidido usar (PIC16X84 y PIC16F87X), encontré un sitio en internet de un programador desarrollado por Miguel Scapolla llamado PIC800, teniendo las características necesarias para programar nuestros modelos de PIC a usar.

Características y funcionamiento del programador.

PIC800 es un programador de microcontroladores PIC de Microchip. Soporta los modelos 16C84, 16F84, 16F84A, 16F873, 16F874, 16F876 y 16F877. El hardware se conecta al puerto paralelo de la impresora, y el software corre bajo DOS o en una ventana DOS bajo Windows 3.x o 9x. Debido a los sistemas de protección que tienen, PIC800 no

funciona bajo Windows 2000. Windows NT ni Linux. ya que accede directamente al puerto paralelo de la impresora y dichos sistemas operativos no lo permiten. La información de como diseñarlo fue extraída de las notas de aplicación de Microchip números DS39025. DS30189 y AN589. La primera trata sobre la programación de los PIC16F87x. la segunda sobre los PIC 16x84. y la tercera explica como armar un prototipo de programador básico. En ambas familias. el sistema de programación es el mismo. Lo único que cambia es la cantidad de memoria de programa y datos. por lo que las dos primeras notas de aplicación son muy similares. Las tres notas de aplicación explican como deben ser las señales que le deben llegar al PIC durante la fase de programación y lectura. y que debe hacer el software del programador.

El hecho de controlar estas señales en forma adecuada dan el dominio sobre el PIC. Con esto quiero decir que no es necesario que se programe el PIC usando el puerto paralelo de la PC y el software y hardware del PIC800. sino que mientras se generen las señales adecuadas. el PIC puede ser programado correctamente. Por eso. es posible usar un puerto serie RS-232. placas especiales de entrada y salida. y porque no. otro microcontrolador (sea o no de Microchip). dando pie a cambios y actualizaciones del software en forma remota.

Por los costos de hoy en día. es recomendable contar con una computadora para programarlos. ya que existen compiladores de lenguaje C. ensambladores y programadores que necesitan una. La alternativa es hacer el programa en código de máquina y utilizar un programador empotrado e introducir el programa directamente en hexadecimal. pero este sistema (si lo hay) es muy engorroso durante la fase de desarrollo y depuración. En resumen. al PIC no le interesa como se generan las señales. Mientras se respete el protocolo de programación. es lo mismo que sean generadas por una computadora. por un microcontrolador. o por simples pulsadores manuales.

Debido a que Microchip hizo público el protocolo de programación de sus microcontroladores. existe una gran cantidad de programadores. tanto comerciales como caseros. Los programadores comerciales se caracterizan por soportar gran cantidad de modelos de micros. incluso algunos soportan modelos de otras marcas. Obviamente. las

empresas no distribuyen los circuitos esquemáticos. En cambio, los programadores caseros se caracterizan por tener un bajo costo y por disponer los circuitos esquemáticos y programas en forma libre y gratuita. Incluso algunos distribuyen el código fuente del programador. Debido al bajo costo y por ende a la pequeña cantidad de componentes que necesitan, soportan una cantidad limitada de modelos de microcontroladores que pueden programar. A su favor, tienen la ventaja que para los PIC16x84 y 16F87x son fácilmente adaptables para usarlos con el sistema ICSP (programación serie en el circuito). Este sistema permite programar al PIC en el mismo circuito en el que va a funcionar, por lo que no es necesario sacarlo de su base para ponerlo en el programador cada vez que hay que reprogramarlo. Esto es muy útil durante la fase de desarrollo y depuración del software del micro.

Este programador PIC800 es un programador casero. No es un programador comercial, ya que no tiene implementado la verificación de la grabación a distintas tensiones de alimentación. En un programador comercial, luego de la grabación, el PIC es verificado a la tensión de alimentación mínima y máxima, con lo que se puede asegurar el correcto grabado de la memoria. PIC800 hace la verificación solo a la tensión nominal de alimentación (+5V), con lo que por lo menos se asegura el buen estado de grabación a esa tensión.

Hardware

El hardware de PIC800 es sencillo. El circuito está dividido en tres partes: alimentación, control y señalización.

La alimentación del circuito se obtiene de una fuente externa de entre 15 a 24Vcd y 500mA. Un regulador 7805 se encarga de alimentar al PIC con los +5V, y un regulador 7812 con dos diodos 1N4148 conectados en serie a su masa se encarga de proporcionar la tensión de programación Vhh (+13.4V). Los capacitores eliminan los ruidos eléctricos y previenen las oscilaciones. Es conveniente que el switch de encendido - apagado sea

bipolar para evitar el pasaje de ruidos eléctricos desde el transformador a la PC a través de la masa cuando el programador está apagado.

La parte de control utiliza transistores de efecto de campo BS170 (siempre tomando en cuenta el costo de estos, que oscila entre .80 pesos y 1.20 pesos); los cuales toman las señales que envía la PC, las invierten y las envían al PIC. De esta manera se hace una separación eléctrica entre la PC y el PIC. En el caso del bit de *Data*, hay otro transistor que toma la señal emitida por el PIC, lo invierte y lo envía a la PC (en líneas punteadas). Ese bit es utilizado por el programador para leer el PIC.

La señalización consiste en otros tres transistores que toman las señales recibidas por el PIC y manejan leds para indicar el estado de cada señal: ***Data, Clock y Vpp.*** Obviamente esto es opcional y la falta de señalización no afecta el funcionamiento del programador.

En caso de no poder usar o no conseguir transistores BS170, se puede experimentar utilizando transistores bipolares del tipo BC548 o BC549. Tienen la desventaja de necesitar corriente de base para la excitación y de no tener una tensión emisor - colector de 0V cuando está en saturación. Esto puede provocar errores en la programación si es una tensión lo suficientemente elevada para que el PIC lo tome como un uno lógico. Por eso, lo conveniente es buscar un transistor que tenga una VCES lo mas chica posible (no mas de 0,5V) y una ganancia HFE elevada (mas de 200) para asegurar la saturación con una corriente de base pequeña. Los parámetros necesarios son:

Polaridad = NPN
VCEO >= 30V
IC >= 50mA
VCES <= 0,5V
β >= 200

La construcción del programador no deberá presentar ningún problema. Se deber tener cuidado al soldar los transistores y no calentarlos mucho ya que son muy sensibles. El conector para el cable que va a la PC debe tener cinco pines como mínimo. Puede ser una ceetor DB9 (como del puerto serie), un RJ45 (como las de red Ethernet) o alguna otra que sea cómoda para conectar. Para el cable de conexión con la PC se recomienda utilizar cable multipar apantallado. La contraparte del cable debe ir conectada al chasis de la PC a través de u conector DB25. No se debe unir el chasis de la PC con el negativo de la fuente de alimentación. El switch es opcional, aunque es recomendable instalarlo y que corte ambos polos de la fuente para poder desenchufar el transformador y que los ruidos eléctricos no afecten a la PC cuando está conectada.

El único problema que se presenta es la elección de la base para el PIC. Si se van a programar ambos tipos de PIC (16x84 y 16F87x), esto es un problema porque no coinciden los pines de alta tensión de programación (Vpp). Las soluciones son dos:

- Usar una base ZIF *(Zero Input Force)* que acepte ambos tipos de PIC y un jumper para seleccionar a que pin se le hace llegar la tensión de programación (13,4V). Es la elección menos adecuada ya que si por error se coloca el jumper en la posición equivocada, es bastante probable que se dañe el PIC.

- Usar dos (o tres) bases separadas: uno de 18 pines (2 x 9), otro de 28 pines (2 x 14), y otro de 40 pines (2 x 20). Tiene la desventaja de que de esta manera se agranda el tamaño de la placa impresa, pero se minimiza la posibilidad de errores.

Software.

Instalación.

Este programador no trae un programa instalador porque no es necesario. Para instalarlo, simplemente copie el archivo PIC800.EXE a un disco y subdirectorio que está.

en la variable de entorno PATH, para que DOS lo pueda ejecutar sin necesidad de especificar su ruta. Inclusive se puede copiar a una unidad de red o a un disco protegido contra escritura, ya que el software no genera ningún archivo de configuración.

Modo de uso.

Todos los parámetros se le deben pasar a través de la línea de comandos del DOS. El modo de uso es el siguiente:

PIC800 <comando> [-opción ...] <archivo> [-opción ...]

Donde:

PIC800 es el programa ejecutable del programador. n comando puede ser uno de los siguientes:

P: Programa al PIC desde un archivo HEX.
L: Lee el PIC y genera un archivo HEX.
V: Verifica el PIC con un archivo HEX.
B: Borra el contenido del PIC.
T: Testea el hardware del programador.

"*Archivo*" es un archivo hexadecimal en formato generado por MPASM, y es leído al programar o verificar un PIC, o generado al leer un PIC. Las opciones son:

-L x: Número de puerto LPT a usar (1, 2 o 3). Por defecto, usa LPT1.
-V x: Factor de corrección de la velocidad de procesamiento (1 a 1000). Por defecto es 100.
-P x: Fuerza el tipo de PIC a utilizar, evitando la auto detección. Los valores permitidos son: 84, 873, 874, 876 y 877.

La detección del tipo de PIC se hace leyendo la dirección 0x2006 que contiene un valor único para cada modelo. El factor de corrección de velocidad es un porcentaje que se

usa para acelerar o ralentizar la lectura y escritura del PIC. La velocidad nominal es de 100 (100%). Con 300 (300%) se triplica la velocidad, con 50 (50%) se baja la velocidad a la mitad, etc. Esto es útil cuando hay problemas de grabación con máquinas muy rápidas o cuando se quiere acelerar el proceso con máquinas lentas.

Depende del entorno que se este, usando, hay varias formas de utilizarlo:

Si sólo se usa desde DOS (con el compilador MPASM para DOS), conviene crear un programa BAT para invocar al programador sin tener que teclear todos los parámetros cada vez. Por ejemplo:

```
rem pgm.bat
@echo off
cls
pic800 P prueba.hex -L 2
```

Entonces, PGM.BAT graba el PIC con PRUEBA.HEX utilizando LPT2. También se pueden crear los archivos BORRA.BAT y VERI.BAT para borrar el contenido y verificar respectivamente.

Si se esta bajo Windows 3.1x, quizá lo mas conveniente es crear un grupo de iconos (en el Administrador de Programas) con el proyecto. Por ejemplo, podrían ser los iconos del MPLAB, del Administrador de Archivos, y un icono del PIC800 por cada tarea a realizar y con un nombre descriptivo. Por ejemplo, el icono para programar se podría llamar "Programar PIC" y en la línea de comandos (en las propiedades del icono) se pueden poner los par metros. O también, que ese icono llame a PGM.BAT descrito antes.

Bajo Windows 95, lo mas fácil es crear accesos directos en el Escritorio, y en las propiedades, llamar al programador con los parámetros necesarios.

No funciona en Windows 2000 ni en Windows NT, ya que PIC800 accede en forma directa al puerto de la impresora, y dichos sistemas operativos no lo permiten.

Deber usar el comando de borrado (B) con cuidado. PIC800 no pregunta si está de acuerdo o no en borrar el PIC, sino que lo borra sin advertencia previa. Al ser un borrado por hardware de la memoria de programa, datos y configuración, es imposible recuperar el código del PIC.

PIC800 detecta automáticamente el tipo de PIC conectado, por lo que no es necesario especificar el modelo en la línea de comandos. También verifica la consistencia del archivo hexadecimal y que este archivo se corresponda con el PIC. De no corresponder, ya sea porque el archivo es para otro modelo o porque tiene direcciones de memoria inválidas, PIC800 aborta la grabación presentando un mensaje de error.

La programación de los PIC's de las series 16x84 y 16F87x es relativamente sencilla. Se utiliza una comunicación serie síncrona con dos hilos: dato y reloj. De esta manera, se le envían comandos para que el PIC los ejecute y datos para que los almacene. Lo que sigue está tomado de las hojas de programación de la EEPROM de los PIC con números DS39025, DS30189 y DS30262 de Microchip.

Modo de programación.

Para poner al PIC en modo programación, solo hay que mantener en nivel bajo los pines RB6 (reloj) y RB7 (dato) mientras se produce el flanco ascendente de bajo (0V) a Vpp (13,4V) del pin MCLR. Una vez en este estado, se puede acceder a la memoria de programa. RB6 es usado como entrada de reloj, y RB7 es usado para entrada de bits de comandos y para entrada y salida de bits de datos durante la operación serie. Para ingresar un comando se necesitan 6 ciclos de reloj. Cada bit de comando es almacenado en el flanco de bajada del reloj, con el bit menos significativo (LSB) ingresando primero. En caso de que el comando lleve un dato asociado, se ingresa dicho dato en 16 ciclos de reloj, también

con el bit menos significativo primero. Los datos son de 14 bits, por lo cual, el primer bit es un bit de comienzo y el último es un bit de parada. Si el comando es para leer un dato, el pin RB7 actúa como salida y envía cada bit de dato en el flanco de subida del reloj, también con 16 bits: bit de comienzo, 14 bits de datos y un bit de parada. El dato sale con el bit menos significativo primero.

Comandos

Load Configuration: Después de recibir este comando y 16 ciclos de reloj, el contador de programa (PC) se sitúa en la dirección 0x2000. De esta manera se puede programar la memoria ID Loc, la de configuración y la de datos. El único modo de volver a poder programar o leer la memoria del programa (0x0000 a 0x1FFF) es reseteando el PIC con el pin MCLR a bajo, saliendo del modo de programación y volviendo a entrar en .l.

Load Data for Program Memory: Después de recibir este comando, el PIC almacena el dato que sigue con los próximos 16 ciclos de reloj para ser grabado en la memoria del programa, ID Locs o configuración.

Load Data for Data Memory: Después de recibir este comando, el PIC almacena el dato que sigue con los próximos 16 ciclos de reloj para ser grabado en la memoria de datos. Nótese que la memoria de datos es de 8 bits por byte, por lo tanto solo se tendrán en cuenta los 8 bits menos significativos de los 14 que componen el dato recibido.

Read Data from Program Memory: Después de recibir este comando, el PIC transmite los bits de datos de la memoria del programa, ID Locs, identidad o configuración, según a donde está apuntando el contador de programa (PC). El pin RB7 (dato) cambia a modo salida en el segundo flanco ascendente del reloj y vuelve al modo entrada (alta impedancia) después del 16vo flanco ascendente del reloj.

Read Data from Data Memory: Después de recibir este comando, el PIC transmite los bits de datos de la memoria de datos (EEPROM) en la que está apuntando el

contador de programa (PC). El pin RB7 (dato) cambia a modo salida en el segundo flanco ascendente del reloj y vuelve al modo entrada (alta impedancia) después del 16vo flanco ascendente del reloj. Nótese que la EEPROM de datos tiene 8 bits por byte, por lo que sólo los 8 bits menos significativos deberán ser tenidos en cuenta.

Increment Address: Después de recibir este comando, el contador de programa (PC) es incrementado en uno. Si está apuntando a la memoria de programa (0x0000 a 0x1FFF), cuando su valor es 0x1FFF, pasa a apuntar a la dirección 0x0000 (no a 0x2000). Si está apuntando a la memoria de datos (0x2000 a 0x3FFF) cuando su valor es 0x3FFF, pasa a apuntar a la dirección 0x2000 (no a 0x0000). El único modo de pasar de la memoria de datos a la memoria de programa es reseteando el PIC saliendo del modo de programación y volviendo a entrar en el *Begin Programming*: Programa la memoria del PIC según el último comando "*Load Data for Program Memory*" o "*Load Data for Data Memory*" ingresado. Es necesario ejecutar uno de los dos comandos anteriores para programar la posición de memoria. No se requiere ningún comando de final de programación

Bulk Erase Program Memory: Después de recibir este comando, con el próximo comando "*Begin Programming*" se borra toda la memoria de programa. Para realizar el borrado en forma correcta se deben cumplir los siguientes pasos:

1 - Comando *Load Data for Program Memory* con 0xFFFF como valor
2 - Comando *Bulk Erase Program Memory*
3 - Comando *Begin Programming*
4 - Esperar 10 ms

Si el PIC tiene la protección de código activado, la memoria de programa no se borra.

Bulk Erase Data Memory: Después de recibir este comando, con el próximo comando *"Begin Programming"* se borra toda la memoria de datos (EEPROM). Para realizar el borrado en forma correcta se deben cumplir los siguientes pasos:

1 - Comando *Load Data for Data Memory* con
0xFFFF como valor
2 - Comando *Bulk Erase Data Memory*
3 - Comando *Begin Programming*
4 - Esperar 10 ms

Protección de código.

Si el PIC tiene la protección del código activado, al leerlo se obtienen todas las posiciones de memoria en cero, y la programación de esa parte de la memoria esta deshabilitada. Para poder reutilizar el PIC, es necesario deshabilitar esta protección antes de intentar grabar en el. Los pasos para deshabilitarla son:

- Enviar el comando Load Configuration con 0xFFFF como dato.
- Incrementar la dirección (PC) con el comando *Increment Address* a la dirección 0*2007 (configuración).
- Enviar el comando 0*01.
- Enviar el comando 0*07.
- Enviar el comando *Bejin Programming.*
- Esperar 10 ms.
- Enviar el comando 0*01.

Archivo Hexadecimal.

El archivo hexadecimal contiene el programa compilado por el MPASM u otro compilador para PIC's. Este archivo es leido por el programador (en este caso es PIC800) y es transferido al PIC. El formato de este archivo est documentado en el ap.ndice A del

manual del MPASM *(Hex File Format)*. PIC800 sólo reconoce el formato Intel INHX8M que se describe a continuación.

Este formato produce un archivo hexadecimal de 8 bits con una combinación de byte bajo - alto. Como cada dirección contiene sólo 8 bits, todas las direcciones están duplicadas. Cada registro de datos comienza con un prefijo de 9 caracteres y termina con 2 caracteres de checksum. Cada registro tiene el siguiente formato:

:BBAAAATTHHHHHHH....HHHHCC

Donde:

: es el caracter dos puntos, y representa el comienzo de un registro.

BB es un byte de dos dígitos hexadecimales que representan la cantidad de bytes de datos que contiene la línea.

AAAA Es una dirección de cuatro dígitos hexadecimales que representan la dirección de comienzo del registro de datos (multiplicada por dos).

TT es un byte de dos dígitos hexadecimales que siempre es '00', excepto en el registro de final de archivo, en donde es '01'.

HH...HH son pares de bytes de dos dígitos hexadecimales, de la forma byte bajo - byte alto, y representan los datos a grabar en el PIC.

CC es un byte de dos dígitos hexadecimales que representan el checksum de verificación de errores del registro de datos. Se calcula como el complemento a dos de la suma de todos los bytes anteriores en el registro.

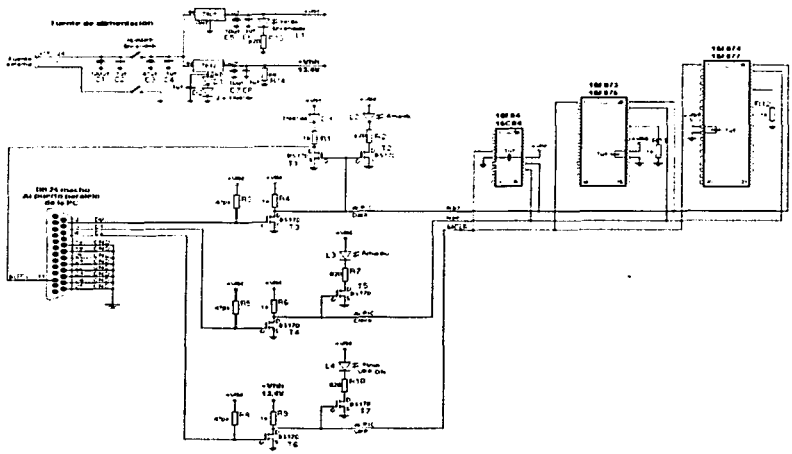


Figura 6.1 Esquemático del circuito programador.

```
PIC800 - Programador de PIC 16x84 y 16F87x de Microchip - Versión 1.41b
(C) 2001 por Miguel Scapolla, miguelo@miguelo.com.ar, http://www.miguelo.com.ar
******* PIC800 no ofrece ninguna garantía. Vea la licencia GNU adjunta *******

Modo de uso:
  PIC800 <comando> [-opción ...] [Archivo.HEX] [-opción ...]

  Comando:                    Opciones:
   P: Programa el PIC.          -L x: Nro de LPT a usar (1, 2 o 3).
   L: Lee el PIC.               -V x: Escala de velocidad (1% a 1000%).
   V: Verifica el PIC.          -P x: No detecta el modelo de PIC.
   B: Borra el PIC.                   Valores: 84, 873, 874, 876 y 877.
   T: Testeo del programador.         (Fuerza el modelo de PIC).

Error: Falta la tarea a realizar. Especifiquela en la línea de comandos.
```

Figura 6.2 Forma en que se presenta la pantalla bajo MS-DOS el entorno del programa PIC800

6.2. Práctica No. 2. Programa de prueba para los microcontroladores PIC.

Objetivos.

- Estructurar un programa sencillo para los microcontroladores PIC16F84 y PIC16F876, configurando sus puestos como entradas o salidas.
- Plantear las diferencias que se presentan en un programa para dos modelos de PIC que comparten el mismo repertorio de instrucciones, pero que poseen distintos recursos y diferencias en sus registros.

Materiales.

- 1 Microcontrolador PIC16F84.
- 1 Microcontrolador PIC16F876.
- 1 Cristal de cuarzo a 4 Mhz.
- 2 Interruptores.
- 1 Switch pushbutton.
- 3 Resistencias de 10 KΩ.
- 2 Resistencias de 330 Ω.
- 1 Resistencia de 100 Ω.
- 1 Diodo 1N4148.
- 2 Diodos emisores de luz.

En esta primera práctica compete conocer el funcionamiento de los microcontroladores PIC específicamente el PIC16F84 que es el más conocido y rudimentario, pero a la vez, tiene un buen número de recursos. El mismo problema se plantea para uno de los más recientes microcontroladores PIC16F87X, concretamente el PIC16F876 conteniendo éste mucho más posibilidades como se vio en los capítulos anteriores.

El problema es el siguiente: se tiene un microcontrolador trabajando a 4 Mhz, con dos interruptores conectados en el puerto A en las líneas RA1 y RA2, y dos leds en el puerto B líneas RB0 y RB1. El funcionamiento es sencillo, cuando los dos interruptores

otorgan a las líneas de entrada un valor de 0 volts los dos leds puestos en las líneas del puerto B deben de encender. Para cualquier otra combinación (0-1, 1-0 y 1-1), los leds deben de permanecer apagados.

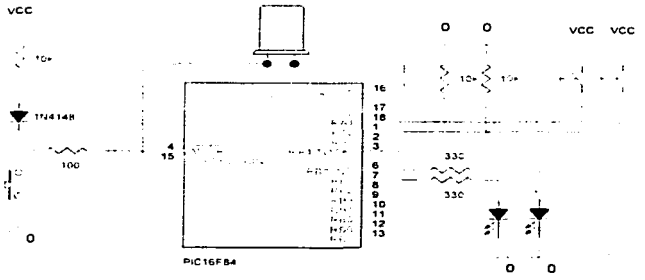


Figura 6.3

Programa.

```
        list    p=16f84         ;tipo de procesador
        radix   hex             ;sistema de numeración hexadecimal
        include "p16f84.inc"    ;registros internos de una libreria
        ORG     0x00            ;se va al vector de reset
        goto    INICIO
        ORG     0x05            ;se va al vector de interrupcion
INICIO  bsf     STATUS,RP0      ;cambio al banco 1
        clrf    TRISB           ;puerto B como salida
        movlw   b'00000110'     ;se definen las lineas RA1 y RA2 como entradas
        movwf   TRISA
        bcf     STATUS,RP0      ;cambia al banco 0
        clrf    PORTB           ;borra los leds
        clrf    PORTA           ;borra las lineas no utilizadas del puerto A
BUCLE   movf    PORTA,W
        btfss   STATUS,Z
        goto    OFF             ;si RA1 y RA2 no son 0
```

```
        goto    ON              ;si RA1 y RA2 son 0

OFF     clrf    PORTB           ;se apagan los leds
        goto    BUCLE           ;comprobar

ON      movlw   0xff            ;se encienden los leds
        movwf   PORTB
        goto    BUCLE           ;vuelve a comprobar

        END                     ;fin del programa
```

Mismo programa para el PIC16F876.

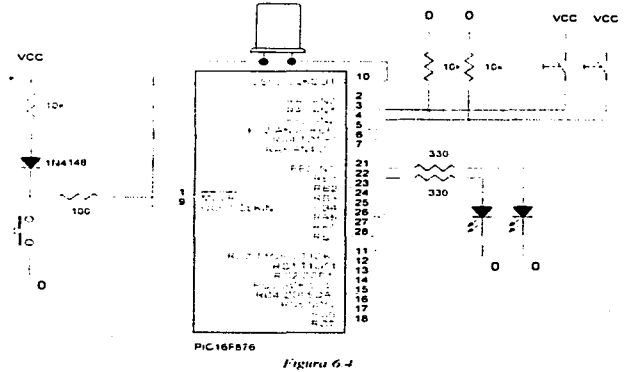


Figura 6.4

Programa.

```
*       list    p=16F876        ;tipo de procesador

        radix   hex
*       include "p16f876.inc"   ;registros internos de libreria

        org     0x00
        goto    INICIO
        ORG     0x05

INICIO  bsf     STATUS,RP0
        bcf     STATUS,RP1
```

```
*       clrf   TRISB                ;puerto B como salida
        movlw  b'00000110'          ;Puerto A E/S digitales
*       movwf    ADCON1
*       movlw  b'00000110'          ;RA1 y RA2 entradas
        movwf  TRISA
        bcf      STATUS,RP0
clrf             PORTB
clrf             PORTA

BUCLE  movf    PORTA,W
       btfss       STATUS,Z
       goto        APAGAR
       goto        ENCENDER

APAGAR clrf    PORTB
        goto   BUCLE

ENCENDER       movlw  0xFF
               movwf  PORTB
               goto   BUCLE

       END
```

Las líneas que aparecen con un asterisco al principio, definen los cambios realizados en el programa para este tipo de PIC, en general, las instrucciones y la estructura del programa es el mismo para el PIC16F86 y el PIC16F876.

6.3. Practica No 3. Relevadores.

Objetivos.

- Comprender el funcionamiento de circuitos digitales asociados con dispositivos electromecánicos que manejan tensiones altas en corriente alterna.
- Generar las señales necesarias para activar en optoacoplador y un relevador controlando una carga.

Materiales.

- 1 Microcontrolador PIC16F84.
- 1 Cristal de cuarzo a 4 Mhz.
- 1 Optoacoplador H11B1.
- 1 Relevador a 12 VDC, 10 Amp.
- 1 Lámpara o foco a 115 VAC.
- 1 Switch pushbutton.
- 1 Resistencia de 10 KΩ.
- 1 Resistencia de 330 Ω.
- 1 Diodo 1N4004.

El relevador es un dispositivo electromecánico, que transforma una señal eléctrica en un movimiento mecánico. Este consiste de una bobina arrollada en un núcleo de metal, y una armadura de metal con uno o más contactos.

Cuando un voltaje es aplicado en la bobina, la corriente debe fluir y un campo magnético es producido moviendo así la armadura para abrir una serie de contactos y abrir otra.

Cuando el voltaje es removido del relevador, el flujo magnético en la bobina colapsa y produce un voltaje alto en dirección opuesta.

Conexión de un optoacoplador y un relevador al microcontrolador.

Un relevador puede ser activado vía un optoacoplador . El opto es un dispositivo que contiene en su interior un diodo emisor de luz y un transistor (este puede ser en configuración normal o darlington). Básicamente el LED se activa mediante una señal proveniente del exterior, y a su vez otorga a la base del transistor la señal necesaria para que éste entre en saturación. Como se sabe, un transistor es un amplificador de corriente y la demanda de ella depende de la carga conectada, en este caso el motor, lámpara o foco, así lo requieren. En este caso la señal antes pasa por el relevador el cual activa la carga.

En esencia, el optoacoplador maneja un relevador y el relevador a su vez activa al motor, lámpara, foco, etcétera.

A continuación se presenta el esquemático de una conexión con relevador y optoacoplador de una carga; así como el programa descrito y los macros utlizados.

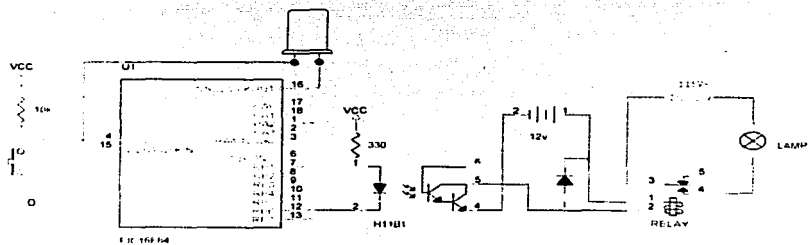


Figura 6.5

¿Qué es un macro?

Un macro es un subprograma que involucra rutinas para el manejo de elementos o dispositivos en un programa; es decir, son segmentos de programación que encajan en un programa y que facilitan la elaboración del mismo.

Muchos programadores expertos realizan los macros y los ponen a disposición del público para su uso.

En este caso utilizaremos algunos macros obtenidos mediante páginas WEB para el control de dispositivos (display LCD) y el control de señales.

Macros utilizados.

Wait.inc

Este archivo wait.inc contiene 2 macros **wait** y **waitx.** Con estos macros es posible asignar tiempos de barrido en diferentes intervalos. Ambos macros utilizan el desbordamiento del contador TMR0 como un intervalo básico. Para poder cambiar el valor del preescalar debemos cambiar la longitud del intervalo del TMR0.

```
;***** Declarando constantes *****

    CONSTANT PRESCstd = b'00000001' ; valor estandar del preescalar del TMR0

;***** Macro *****

WAIT   macro  timeconst_1
       movlw  timeconst_1
       call   WAITstd
       endm

WAITX  macro  timeconst_2, PRESCext
       movlw  timeconst_2
       movwf  WCYCLE
       movlw  PRESCext
       call   WAIT_x
       endm

;***** Subprogramas *****

WAITstd  movwf  WCYCLE
         movlw  PRESCstd
WAIT_x   clrf   TMR0
         BANK1
         movwf  OPTION_REG     ;asigna el preescalar al timer
         BANK0
```

```
WAITa   bcf    INTCON,T0IF     ; borra la bandera de desbordamiento del TMR0
WAITb   btfss  INTCON,T0IF     ; checa cuando este borrado

        goto   WAITb           ; espera salto
        decfsz WCYCLE,1        ; repite el salto si el periodo de barrido no ha sido corrido
        goto   WAITa
        RETURN
```

Si nosotros usáramos el cristal de 4Mhz. para valores de preescalares de 0, 1 y 7 que dividen el reloj básico del oscilador, el intervalo siguiente por un desbordamiento del timer. TMR0 debe ser 0.512, 1.02 y 65.3ms.

El valor estándar asignado al preescalar de este macro es 1 (1.02ms), y este no puede ser cambiado.

Taster.inc

```
TASTER  macro HiLo, Port, Bit, Delay, Address

        Local    Exit          ; etiquetas locales
        Local    Loop

        if HiLo == 0           ; esta presionada la llave?
        btfsc Port,Bit         ; linea de entrada en bajo?
        else
        btfss Port,Bit         ; linea de entrada en alto?
    endif
        goto Exit              ; si la llave no ha sido presionada salir del macro

        WAIT Delay             ; espera retraso
Loop
        if HiLo == 0
        btfss Port,Bit         ; la llave se ha disparado ?
        else
        btfsc Port,Bit
        endif
        goto Loop

        WAIT Delay             ; espera retraso

        call Adrress           ; llama a subprograma
Exit                           ; salir del macro
        endm                   ; Kfin del macro
```

Hacerca de este macro hay que hacer referencia a algunos argumentos para que quede claro la función que realiza este dentro de un programa.

HiLo.- *Puede ser '0' o '1' cuando representa subida o bajada donde el subprograma debe ser ejecutado cuando se presiona la llave.*

Port.- Representa un puerto del microcontrolador cuando un switch es conectado a este. En el caso de un Pic16f84, este puede ser l puerto A o el puerto B.

Bit.- *Representa una sola línea de algún puerto donde la llave o switch es conectado.*

Delay.- *Es un número de 0 hasta 255, usado para asignar el tiempo de detección de la llave para detenerse. El valor es calculado de la siguiente forma:* ***TIME=Delay x 1ms.***

Address.- *Es la dirección donde el microcontrolador donde la llave es detectada.*

EJEMPLO: TESTER0, PORTA, 3, .100, Tester1_below

- El switch 1 es conectad0 en RA0 (primer salida del puerto A) con un retraso de 100 microsegundos y una respuesta para 0 lógico.

Bank.inc

```
;    Macro estandar para PIC16FXXX
;    ==============================

BANK0  macro
         bcf    STATUS,RP0     ; selecciona banco de memoria 0
         endm

BANK1  macro
         bsf    STATUS,RP0     ; selecciona banco de memoria 1
         endm
```

Programa.

```
    PROCESSOR 16f84
    #include "p16f84.inc"

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

***** Variables *****

        Cblock  0x0C
        WCYCLE
        PRESCwait
```

```
  endc

;***** Declarando hardware *****

        #define RELAY PORTA,3   ;relevador en el cuarto pin del puerto A

;***** Estructura de la memoria de programa *****

        ORG     0x00                    ; vector de reset
        goto    Main

        ORG     0x04                    ; vector de interrupcion
        goto      Main                  ; rutina de no interrupcion

        #include "bank.inc"            ; macros
        #include "taster.inc"
        #include "wait.inc"


Main                            ; inicio de programa
        BANK1
        movlw   0x17            ;inicializa el puerto A
        movwf   TRISA           ; TRISA <- 0xff
        movlw   0x00            ; inicializa el puerto B
        movwf   TRISB           ; TRISB <- 0x00
        BANK0

        clrf        PORTB   ; PORTB <- 0x00

Loop
        TASTER 0, PORTA, 0, .100, On        ; Taster 0
        TASTER 0, PORTA, 1, .100, Off       ; Taster 1

        goto    Loop

On
        bsf RELAY           ; relevador encendido
        return

Off
        bcf RELAY           ; relevador apagado
        return
  End
```

6.4. Práctica No. 4. Conexión de un display inteligente (LCD).

Objetivos.

- Manejar un display LCD mediante la generación de impulsos por medio del PIC16F84 y el control de las señales de control (lectura/escritura y habilitación).
- Mostrar en la pantalla del display una secuencia de caracteres y mensajes utilizando el código del propio dispositivo.

Materiales.

- 1 Microcontrolador PIC16F84.
- 1 Cristal de cuarzo a 4 Mhz.
- 1 Display LCD de 16 columnas x 2 líneas.
- 1 Potenciómetro de 10 KΩ.

Muchos microcontroladores son utilizados para el manejo de displays inteligentes o displays LCD para visualizar información referente a algún sistema. En este apartado se plantea la posibilidad de utilizar un microcontrolador PIC para manejar un display de este tipo. El display que se utilizará en este caso será un display de 16 * 2 (16 columnas, 2 líneas), que tiene la capacidad de visualizar 8 * 80 pixeles. Posee un código ASCII estándar, pudiéndose así visualizar un gran número de letras y símbolos.

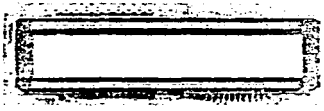


Figura 6.6 Display LCD

Las mayoría de estos módulos de visualización poseen características estándar: en este caso 14 pines, de los cuales 3 son para la alimentación y contraste, 3 líneas de control (RD/WR, RS y E) y 8 líneas para datos.

Pin No.	Symbol	Level	Description
1	Vss	0V	Ground
2	Vcc	5.0V	Power supply voltage for logic and LCD( )
3	Vee	0-3V	Power supply voltage for LCD( )
4	RS	H/L	Selects registers
5	R/$\overline{W}$	H/L	Selects read or write
6	E	H/L	Starts data read/write
7	DB0	H/L	Data bit0
8	DB1	H/L	Data bit1
9	DB2	H/L	Data bit2
10	DB3	H/L	Data bit3
11	DB4	H/L	Data bit4
12	DB5	H/L	Data bit5
13	DB6	H/L	Data bit6
14	DB7	H/L	Data bit7
15	NC		
16	NC		

Tabla 6.1 Función de los pines del display LCD

La tabla anterior se refiere a la configuración de los pines del módulo LCD.

Las líneas de control conectadas al microcontrolador se comportarían de la siguiente manera:

*La línea **ENABLE** permite el acceso de datos al display a través de las líneas R/W y RS. Cuando esta línea esta en bajo (0 volts) el display esta deshabilitado.*

*La línea **RS** permite la transferencia de datos hacia el display cuando está en nivel alto.*

*La línea de lectura/escritura **R/W** escribe los datos en el display cuando está en nivel alto, los datos escritos se transfieren sólo en la transición de alto a bajo de esta señal. Para el caso de lectura, los datos permanecen disponibles después de la transición de bajo a alto y permanecen ahí hasta que la señal baja nuevamente.*

Las ocho líneas de datos del display pueden transferir información hacia o desde el display. Cabe mencionar que los datos pueden ser transferidos como un byte o como dos nibbles; en este caso en modo de 4 bits. Nosotros utilizaremos nibbles para el manejo del microcontrolador.

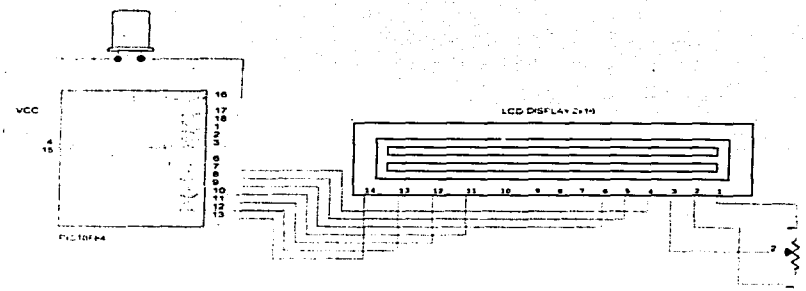


Figura 6.6

Macro para el manejo del display.

Declaracion de hardware

```
RS    equ   1   ; Register Select
RW    equ   2   ; Read/Write
EN    equ   3   ; Enable Output / "CLK"
```

Comandos para display

```
CONSTANT  LCDEM8 = b'00110000'  ; modo de 8 bits, 2 lineas
CONSTANT  LCDDZ  = b'10000000'  ; escribe en la DDRAM
CONSTANT  LCDEM4 = b'00100000'  ; modo de 4 bits, 2 lineas
```

```
;Comandos para la inicializacion del display

      CONSTANT  LCD2L     = b'00101000'   ; funcion: 4 bits 2 lineas
      CONSTANT  LCDCONT = b'00001100'   ; control del display, display encendido
                                        ; cursor apagado
      CONSTANT  LCDSH     = b'00101000'   ; modo de display, autoincremento
                                        ; cursor

;Comandos estándar para el display

      CONSTANT  LCDCLR  = b'00000001' ; limpia display, reinicializa el cursor
      CONSTANT  LCDCH   = b'00000010' ; cursor
      CONSTANT  LCDCR   = b'00000110' ; el cursor se mueve a la derecha
      CONSTANT  LCDCL   = b'00000100' ; el cursor se mueve a la izquierda
      CONSTANT  LCDSL   = b'00011000' ; mueve el contenido del display hacia la izquierda
      CONSTANT  LCDSR   = b'00011100' ;
      CONSTANT  LCDL1   = b'10000000' ; selecciona linea 1
      CONSTANT  LCDL2   = b'11000000' ; selecciona linea 2

;***** Macro *****

LCDinit macro
     call   LCD_init      ; inicializa LCD

LCDchar macro  LCDarg       ; escribe el carácter en el display
   movlw  LCDarg
     call   LCDdata
     endm

LCDw   macro
     call   LCDdata
     endm

LCDcmd  macro  LCDcommand     ;manda el comando al LCD
     movlw  LCDcommand
     call   LCDcomd
     endm

LCDline macro  line_num
     IF (line_num == 1)
          LCDcmd  LCDL1   ; inicia el macro con la instruccion "first line"
     ELSE
        IF (line_num == 2)
          LCDcmd  LCDL2   ; inicia el macro con la instrucción "second line"
        ELSE
        ENDIF
     ENDIF
     endm

LCD_DDAdr macro DDRamAddress
     Local   value = DDRamAddress | b'10000000'
     IF (DDRamAddress > 0x67)
          ERROR "Wrong DDRAMaddressin LCD_DDAdr "
     ELSE
          movlw   value
```

```
        call   LCDcomd
    ENDIF
    endm

LCD_CGAdr macro CGRamAddress
    Local  value = CGRamAddress | b'01000000'    ;
    IF (CGRamAddress > b'00111111')
        ERROR "Wrong DDRAM address in LCD_CGAdr"
    ELSE
        movlw  value
        call   LCDcomd
    ENDIF
    endm

;***** Subprogramas *****

LCDcomd clrf   LCDbuf       ; borra la bandera de dato
    goto   LCDwr

LCDdata clrf   LCDbuf
    bsf    LCDbuf,RS      ; va a la bandera de dato

LCDwr  movwf  LCDtemp
    andlw  b'11110000'
    iorwf  LCDbuf,0
    movwf  LCDport
    call   LCDclk
    clrf   LCDport
    swapf  LCDtemp,0
    andlw  b'11110000'
    iorwf  LCDbuf,0
    movwf  LCDport
    call   LCDclk
    clrf   LCDport
    RETURN

LCDclk WAITX  0x01, 0x00    ; habilita el acceso de datos y comandos en el LCD
    bsf    LCDport,EN
    bcf    LCDport,EN
    WAIT   0x01
    RETURN

LCD_init
    clrf   LCDport       ; prepara LCDport
    BANK1
    clrf   OPTION_REG
    movlw  b'00000000'
    movwf  LCDtris
    BANK0
    WAIT   0x01

    movlw  LCDEM8        ; inicializacion
    movwf  LCDport       ; inicia con modo de 8 bits
    call   LCDclk
    clrf   LCDport
```

```
    WAIT   0x01

    movlw  LCDDZ        ; escribe un 0 en la DDRAM
    movwf  LCDport
    call   LCDclk
    clrf   LCDport

    movlw  LCDEM4       ; va al modo de 4 bits
    movwf  LCDport
    call   LCDclk
    clrf   LCDport

    LCDcmd LCD2L        ; funcion: modo 4 bits 2 lineas
    LCDcmd LCDCONT      ; display encendido sin cursor
    LCDcmd LCDSH
    LCDcmd LCDCLR       ; limpia display
    call   LCDspecialChars

RETURN

LCDspecialChars         ; maximo numero de caracteres

    LCD_CGAdr 0x00
    LCDchar b'00001010'
    LCD_CGAdr 0x01
    LCDchar b'00000100'
    LCD_CGAdr 0x02
    LCDchar b'00001110'
    LCD_CGAdr 0x03
    LCDchar b'00010001'
    LCD_CGAdr 0x04
    LCDchar b'00010000'
    LCD_CGAdr 0x05
    LCDchar b'00010001'
    LCD_CGAdr 0x06
    LCDchar b'00001110'
    LCD_CGAdr 0x07
    LCDchar b'00000000'

    LCD_CGAdr 0x08
    LCDchar b'00000010'
    LCD_CGAdr 0x09
    LCDchar b'00000100'
    LCD_CGAdr 0x0A
    LCDchar b'00001110'
    LCD_CGAdr 0x0B
    LCDchar b'00010001'
    LCD_CGAdr 0x0C
    LCDchar b'00010000'
    LCD_CGAdr 0x0D
    LCDchar b'00010001'
    LCD_CGAdr 0x0E
```

```
    LCDchar b'00001110'
    LCD_CGAdr 0x0F
    LCDchar b'00000000'

    LCD_DDAdr 0x00      ; reset de la DDRAM

    RETURN
```

El anterior macro se utiliza para el control de un display inteligente LCD de 2x16 líneas.

En esta práctica implementaremos un circuito en el cual se controle un módulo de LCD y sacando datos por el mismo. En este caso vamos a escribir en el display dos palabras una que aparezca situada en la primera líneas y otra que se encuentre en la segunda. Lo principal en esta práctica es manejar bien la línea de comandos para el display habilitando las líneas que se requieran para hacer funcionar el display.

Programa.

```
;***** Declarando configuración del microcontrolador *****

    PROCESSOR 16f84
    #include "p16f84.inc"

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

        Cblock 0x0C
        LCDbuf
        LCDtemp
        WCYCLE
        PRESCwait
        Pointer
        endc

;***** Declarando hardware*****

        LCDtris  equ     TRISB
        LCDport equ    PORTB

        ORG    0x00      ; vector de reset
        goto   Main

        ORG    0x04      ; vector de interrupción
        goto     Main

Poruke

    movwf  PCL
```

```
Poruka1 dt "uNiVeRsIdAd"
Poruka2 dt "bla, bla"
Poruka3 dt "primer"

Kraj

      #include "bank.inc"
      #include "wait.inc"
      #include "lcd.inc"
          #include "print.inc"

Main                          ; Pocetak programa

   bcf PORTB,2

          LCDinit             ; Incijalizacija LCD-a

   LCDchar 'c'      ; Ispisi karakter na LCD-u
   LCDchar 'a'
   LCDchar 'r'
   LCDchar 'a'
   LCDchar 'c'
   LCDchar 't'
   LCDchar 'e'
   LCDchar 'r'
   LCDchar 'i'
          LCDchar ':'
          LCDchar ' '

          LCDchar 0x00
          LCDchar 0x01

          LCDline 2


          PRINT    Poruke, Poruka1, Poruka2, Pointer, LCDw

Loop      goto     Loop

          End
```

6.5. Práctica No. 5. Control de un motor a pasos.

Objetivos.

- A partir de un modelo en específico de un motor a pasos, y de su secuencia para su funcionamiento, proporcionar las combinaciones y secuencias para la activación y giro del mismo mediante el microcontrolador.
- Controlar el motor de manera total, implementando circuitería para el control del sentido de giro y de la velocidad.

Materiales.

- 1 Microcontrolador PIC16F84.
- 1 Cristal de cuarzo a 4 Mhz.
- 1 Motor a pasos de 4 fases a 5VDC.
- 1 C.I. L293D.
- 1 Transistor 2SC1815.
- 1 Potenciómetro de 10 KΩ.
- 3 Resistencias de 10 KΩ.
- 1 Resistencia de 3.3 KΩ.
- 1 Resistencia de 1 KΩ.
- 3 Switch pushbutton.

Existen diversos tipos de motores a pasos, sin embargo, el principio de operación es el similar para todos los casos. El funcionamiento de estos se basa en las fuerzas electromagnéticas generadas entre el estator y el rotor: a diferencia de otro tipo de motores, consta de más de dos polos, esto hace que necesite conjuntos de impulsos para generar fuerzas que lo hacen girar, este giro, en términos simples lo hace en cierto número de grados.

Dependiendo el tipo de motor a pasos, se necesitan uno o más pulsos para generar movimiento en un sentido; es decir pueden darse en pares y no en uno solo. Las capacidades de potencia, tensión y corriente vienen dadas por el fabricante.

La única condición necesaria para el movimiento continuo en el motor, es que se mande una secuencia correcta de impulsos representados, si se puede mencionar de este modo, por un código binario (señales 1's o 0's).

En ésta práctica se plantea el manejo de un motor a pasos por medio de un microcontrolador PIC16F84 mandando una secuencia de impulsos mediante el puerto RA del microcontrolador. El problema se plantea no solo la posibilidad de hacer girar un motor, si no que poder tener dominio de la velocidad y sentido de giro del motor, así como el paro intempestivo del motor, para ello se utilizan tres switch pushbutton normalmente abiertos, los cuales se conectan al puerto RB.

Explicación de los circuitos de control de velocidad y sentido de giro.

-Velocidad: Este circuito permite el control de la velocidad rotacional del motor. El transistor llega a estar en condición de encendido cuando el pin RR7 del microcontrolador se encuentra en nivel alto, la carga eléctrica del capacitor fluye a través del transistor y el voltaje medido en ambos extremos del capacitor llega a ser casi cero. Cuando RB7 llega a tener un nivel bajo en su salida, el transistor tiene la condición de apagado. En esta condición, la corriente eléctrica fluye a través del potenciómetro y la resistencia hacia el capacitor puesto en serie y lo carga. El voltaje, ahora, en el capacitor gradualmente aumenta.

El voltaje del capacitor es detectado por RB5. Una parte del programa con el que se graba al PIC interrumpe el control del motor hasta que este checa los estados lógicos de las patas RB5 y RB7 del chip. Cuando el valor de la resistencia variable es pequeña, el tiempo de carga del capacitor es relativamente corto, y el control del motor llega a ser rápido.LO opuesto sucede cuando la resistencia es alta en el potenciómetro. El rango de control de

velocidad del motor puede ser cambiado dependiendo del valor del capacitor que se tenga: esto en términos de ecuación se define así:

$$T = RC$$

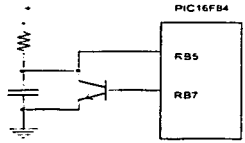


Figura 6.7

-Sentido de giro: El sentido de giro se controla mediante software, es decir, se forza al motor a tomar los pulsos de manera inversa que para un sentido determinado, el switch al ser accionado, hacer que el microcontrolador tome en uno de sus pines un valor y hace que el sentido cambie (derecha, izquierda o se detenga).

Las resistencias *pull up* del microcontrolador son usadas para que el puerto pueda tener un nivel alto cuando el switch este apagado. Hablando de lo anterior, el puerto B del PIC16F84 contiene un arreglo de resistencias *pull up*.

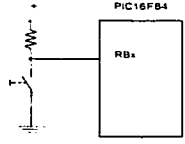


Figura 6.8

Como es bien sabido, los pulsos digitales otorgados por un chip no otorgan la corriente necesaria a un motor para ser manejado apropiadamente. Por ello de todas las posibilidades que se presentan para amplificar la corriente provenientes de señales

digitales: entre ellos transistores o drivers, se eligió un driver, el CI L293 que es muy práctico en el momento de presentar un diseño. De otra forma se tendrían que usar transistores en configuración darlington y otra serie de dispositivos como diodos de protección y resistencias.

X	$\overline{X}$	Y	$\overline{Y}$	Ángulo	X	$\overline{X}$	Y	$\overline{Y}$	Ángulo
0	1	0	1	0.0°	0	1	0	1	180.0°
1	0	0	1	7.5°	1	0	0	1	187.5°
1	0	1	0	15.0°	1	0	1	0	195.0°
0	1	1	0	22.5°	0	1	1	0	202.5°
0	1	0	1	30.0°	0	1	0	1	210.0°
1	0	0	1	37.5°	1	0	0	1	217.5°
1	0	1	0	45.0°	1	0	1	0	225.0°
0	1	1	0	52.5°	0	1	1	0	232.5°
0	1	0	1	60.0°	0	1	0	1	240.0°
1	0	0	1	67.5°	1	0	0	1	247.5°
1	0	1	0	75.0°	1	0	1	0	255.0°
0	1	1	0	82.5°	0	1	1	0	262.5°
0	1	0	1	90.0°	0	1	0	1	270.0°
1	0	0	1	97.5°	1	0	0	1	277.5°
1	0	1	0	105.0°	1	0	1	0	285.0°
0	1	1	0	112.5°	0	1	1	0	292.5°
				120.0°					300.0°
				127.5°					307.5°
				135.0°					315.0°
				142.5°					322.5°
0	1	0	1	150.0°	0	1	0	1	330.0°
1	0	0	1	157.5°	1	0	0	1	337.5°
1	0	1	0	165.0°	1	0	1	0	345.0°
0	1	1	0	172.5°	0	1	1	0	352.5°

Tabla 6.2 Secuencia que debe de seguir el motor para girar

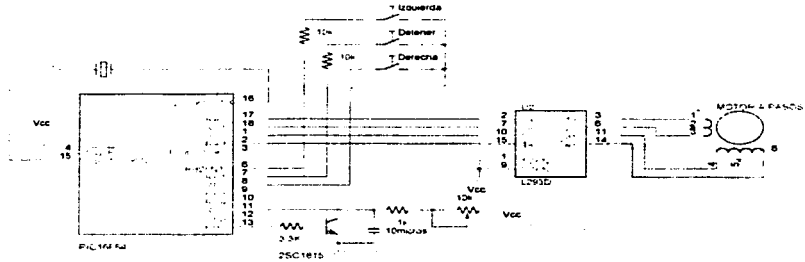


Figura 6.9

Programa.

```
list   p=pic16f84A
       include p16f84A.inc

          __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

;****************** Etiquetas **********************

          cblock  h'0c'
mode        ;Modo de operacion
          ;0=parar 1=derecha 2=izquierda
count1
count2
          endc
rb0    equ   0
rb1    equ   1
rb2    equ   2
rb5    equ   5
rb7    equ   7

;****************** Inicio ************************

          org    0            ;Vector de reset
          goto   init
          org    4            ;Vector de inerrupcion
          clrf   INTCON       ;Borra registro de interrupcion

;****************** Proceso ***********************

init
          bsf    STATUS,RP0    ;Cambia al banco 1
          clrf   TRISA         ;Puerta A como salida
          movlw  b'00100111'   ;RB0,1,2,5=entradas RB7=salida
```

```
        movwf   TRISB           ;Ponte en RB0
        movlw   b'10000000'     ;RBPU=1 resistencias pull up no usadas
        movwf   OPTION_REG      ;Poner OPTION_REG
        bcf     STATUS,RP0      ;Cambia al banco 0
        clrf    mode            ;modo = parar
        clrf    count1          ;Limpia contador
        clrf    count2          ;Limpia contador
        movlw   b'00000101'
        movwf   PORTA           ;Escribe en puerto A
        bsf     PORTB,rb7       ;Pon RB7 = 1
        btfsc   PORTB,rb5       ;RB5 = 0 ?
        goto    $-1             ;No. Espera

start
;************* Checa condicion de switch *****************
        btfsc   PORTB,rb1       ;RB1 = encendido ?
        goto    check1          ;No. Sigue
        clrf    mode            ;Si. Pon modo de parar
        goto    drive           ;No. Salta al drive del motor
check1  btfsc   PORTB,rb2       ;RB2(derecha) = encendido?
        goto    check2          ;No. Sigue
        movlw   d'1'            ;Si. Pon modo derecha
        movwf   mode            ;Salvar modo
        goto    drive           ;No. Salta al drive del motor
check2  btfsc   PORTB,rb0       ;RB0(izquierda) = encendido?
        goto    drive           ;No. Salta al drive del motor
        movlw   d'2'            ;Si. Pon modo derecha
        movwf   mode            ;Salvar modo
;******************** Drive del Motor **********************
drive
        movf    mode,w          ;Lee modo
        bz      start           ;modo = parar
        bsf     PORTB,rb7       ;Pon RB7 = 1
        btfsc   PORTB,rb5       ;RB5 = 0 ?
        goto    $-1             ;No. Espera
        movlw   d'5'
        movwf   count1
loop    call    timer           ;Espera 1msec
        decfsz  count1,f        ;count - 1 = 0 ?
        goto    loop            ;No. Continua
        bcf     PORTB,rb7       ;Pon RB7 = 0
        btfss   PORTB,rb5       ;RB5 = 1 ?
        goto    $-1             ;No. Espera
        movf    PORTA,w         ;Lee PORTA
        sublw   b'000000101'    ;Checa posición de motor
        bnz     drive2
        movf    mode,w          ;Lee modo
        sublw   d'1'            ;Derecha ?
        bz      drive1          ;Yes.Derecha
        movlw   b'00001001'     ;No  Pon dato izquierda
        goto    drive_end       ;Salta al puerto A
drive1
        movlw   b'00000110'     ;Pon dato derecha
        goto    drive_end       ;Salta al puerto a
;-------
```

```
drive2
        movf   PORTA,W      ;Lee pueto a
        sublw  b'000000110' ;Checa posicion de motor
        bnz    drive4
        movf   mode,W       ;Lee modo
        sublw  d'1'         ;Derecha ?
        bz     drive3       ;Si. Derecha
        movlw  b'00000101'  ;No. Pon dato izquierda
        goto   drive_end    ;Salta al puerto a
drive3
        movlw  b'00001010'  ;Pon dato derecha
        goto   drive_end    ;Salta al puerto a
;------
drive4
        movf   PORTA,W      ;Lee puerto a
        sublw  b'000001010' ;Checa posicion de motor
        bnz    drive6
        movf   mode,W       ;Lee modo
        sublw  d'1'         ;Derecha ?
        bz     drive5       ;Si. Derecha
        movlw  b'00000110'  ;No. Pon dato laquierda
        goto   drive_end    ;Salta al puerto A
drive5
        movlw  b'00001001'  ;Pon dato derecha
        goto   drive_end    ;Salta al puerto a
;------
drive6
        movf   PORTA,W      ;Leer puerto a
        sublw  b'000001001' ;Checa posicion de motor
        bnz    drive8
        movf   mode,w       ;Lee modo
        sublw  d'1'         ;Derecha ?
        bz     drive7       ;Si. Derecha
        movlw  b'00001010'  ;No. Pon dato izquierdo
        goto   drive_end    ;Salta al puerto a
drive7
        movlw  b'00000101'  ;Pon dato derecha
        goto   drive_end    ;Salta al puerto a
;------
drive8
        movlw  b'00000101'
drive_end
        movwf  PORTA        ;Escribe en puerto A
        goto   start        ;Salta a inicio
;************ Subrutina de 1msec ****************
timer
        movlw  d'200'
        movwf  count2
tmlp
        nop                 ;Ajuste de tiempo
        nop                 ;Ajuste de tiempo
        decfsz count2,f     ;count - 1 = 0 ?
        goto   tmlp         ;No. Continua
        return
;*************************************************************
```

```
;        FIN de control de motor a pasos
;*****************************************************
         end
```

6.6. Práctica No.6. Utilización de recursos especiales en el PIC16F876. Módulos CCP y convertidor analógico/digital.

Objetivos.

- Haciendo uso específico de opciones distintas del microcontrolador PIC16F876, implementar un controlador de luces en corriente directa variando la intensidad de estas.
- Conocer los módulos de comparación, captura y modulación por ancho de pulsos, así como el convertidor analógico digital del PIC16F876.

Materiales.

- 1 Microcontrolador PIC16F876.
- 1 Cristal de cuarzo de 4 Mhz.
- 1 Transistor 2SC1815.
- 1 Transistor de efecto de campo 2SJ471.
- 1 Potenciómetro de 10 KΩ.
- 2 Resistencias de 10 KΩ.
- 1 Resistencia de 1 KΩ.
- 1 Foco a 5VDC.

Los módulos CCP.

Los microcontroladores PIC16F87X disponen de dos módulos CCP, llamados CCP1 y CCP2, que son idénticos excepto en lo referente a la modalidad de disparo especial.

Dada la similitud, la explicación que sigue, se refiere solo al módulo CCP1.

1ro. Módulo captura. Una pareja de registros de un módulo CCPx captura el valor que tiene el TMR1 cuando ocurre un evento especial en el pin RC2/CCP1 (para el módulo CCP1) o en la RC1/1OSI/CCP2 (para el módulo CCP2).

2do.Módulo comparación. Se compara el valor de 16 bits del TMR1 con oto valor cargado en una pareja de registros de un módulo CCPx y cuando coinciden se produce un evento en los pines RC2/CCP1 y/o RC1/1OSI/CCP2.

3ro. Modo modulación por anchura de pulsos (PWM). Dentro del intervalo del periodo de un impulso controla la anchura en que la señal vale nivel alto.

El módulo CCP1 utiliza un registro de trabajo de 16 bits que esta formado con la concatenación de los registros CCPR1H-CCPR1L. El registro de control de CCP1 es CCP1CON.
El módulo CCP2 tiene como registros de trabajo a CCPR2H-CCPR2L y como registro de control a CCP2CON.

Figura 6.10 Registro CCPx

CCPxM3-0	Modo de trabajo del módulo.
0000	Modulo CCPx desconectado.
0100	Modo captura con cada flanco descendente en RCy/CCpx.
0101	Modo captura con cada flanco ascendente en RCy/CCpx.
0110	Modo captura cada 4 flancos ascendentes en RCy/CCpx.
0111	Modo captura cada 16 flancos ascendentes en RCy/CCpx.
1000	Modo comparación que activa el pin RCy/CCPx al coincidir valores.
1001	Modo comparación que desactiva al pin RCy/CCPx al coincidir valores.
1010	Modo comparación que genera una interrupción software (no afecta RCy/CCPx).
1011	Modo comparación en el que se produce un disparo especial diferente para cada módulo.
11xx	Modo PWM.

Tabla 6.3 Descripción general para los valores de los bits de menos peso del registro CCPx

Modo captura. En este modo, la pareja de registros CCPxH-L del módulo CCPx captura el valor de 16 bits que contiene el TMR1 cuando sucede un evento en el pin RCy/CCPx del puerto C, que previamente ha sido configurado como entrada poniendo a 1 el bit correspondiente del registro TRISC.

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

- Un flanco ascendente.
- Un flanco descendente.
- Cada 4 flancos ascendentes.
- Cada 16flancos ascendentes.

Modo comparación. En esta forma de trabajo los registros CCPR1H-L comparan su contenido, de forma continua, con el valor del TMR1. Cuando coinciden ambos valores, al pin RC2/CCP1, que se halla configurada como salida, ocurren uno de los siguientes eventos, de acuerdo con la programación de los bits CCP1M3-0:

- Pasa a nivel alto.
- Pasa a nivel bajo.
- No cambia su estado pero se produce una interrupción.

Si con los bits CCP1M3-0 se selecciona el modo de trabajo de disparo especial, el módulo CCP1 se pone a 0 y el CCPR1 funciona como un registro de periodo, capaz de provocar periódicamente interrupción. En este modo de disparo especial, el CCP2 pone a 0 el TMR1, y además, inicia una conversión en el conversor A/D, con lo que también y, con carácter periódico, puede realizar conversiones analógico/digitales sin el control del programa de instrucciones.

Explicación del circuito controlador de luces.

El objetivo es realizar un controlador de luces, y para ello el circuito se compone básicamente de dos partes escenciales.

Control del voltaje de entrada al circuito.

El voltaje de entrada es cambiado desde 0 V hasta 5 V por un resistor variable. El voltaje es introducido a través de un puerto analógico del PIC. En este circuito RA0/AN0 es usado para la entrada analógica.
El voltaje mínimo de conversión analógica/digital es Vss=0V y el máximo límite es Vdd=+5V.
El valor del resistor variable que controla este voltaje es de 10 KΩ.
El voltaje de control es tomado en un periodo de 1 milisegundo por CCP2 en el modo de comparación.

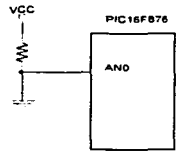


Figura 6.11

Circuito de control de lámpara.

La corriente eléctrica debe fluir de la lámpara, que es a su vez controlada por un transistor FET que funciona como switch.
El FET usado tiene una capacidad de manejar una corriente de 30 Amperes y una potencia de 300W.

Ya que el voltaje de alimentación del PIC es de 5 VDC, es necesario contar con otra fuente de 12 VDC para polarizar el FET.

Como se mencionó el inicio de esta práctica, la lámpara es controlada por el PWM (modulación por anchura de pulsos) de CCP1.

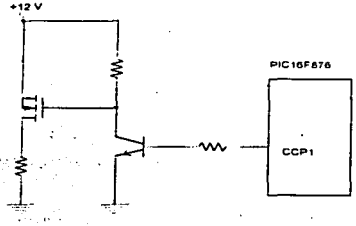


Figura 6.12

A continuación se presenta el circuito esquemático correspondiente al controlador de luces.

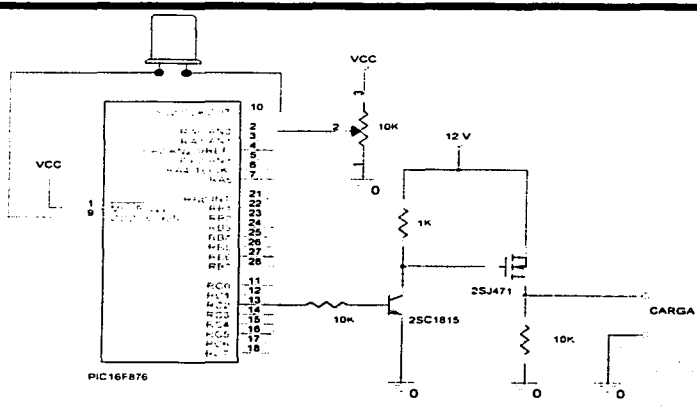


Figura 6.13

Programa.

```
    list        p=pic16f873
     include     p16f873.inc
     __config _hs_osc & _wdt_off & _pwrte_on & _cp_off


     org   0           ;Vector de reset
     goto  init
     org   4           ;Vector de interrupcion
     goto  int

;*************** Proceso inicial ********************

init      bsf   status,rp0    ;Cambio al banco 1
     movlw  b'00000001'   ;AN0 como entrada
     movwf  trisa
     clrf   trisc
     bcf    status,rp0    ;Cambio al banco 0

;A/D inicializacion
     movlw  b'10000001'   ;ADCS=10 CHS=AN0 ADON=ON
     movwf  adcon0        ;Pone el registro ADCON0
     bsf    status,rp0    ;Cambio al banco 1
     movlw  b'00001110'   ;ADFM=0 PCFG=1110
     movwf  adcon1        ;Pone el registro ADCON
```

```
    bcf   status,rp0    ;Cambia al banco 0

;*** PWM initializacion
    clrf  tmr2          ;Limpia el registro TMR2
    clrf  ccpr1l        ;Limpia el registro CCPR1L
    bsf   status,rp0    ;Cambia al banco 1
    movlw d'255'        ;Periodo=1638.4us(610Hz)
    movwf pr2           ;Pone el registro PR2
    bcf   status,rp0    ;Cambia al banco 0
    movlw b'00000110'   ;Pst=1:1 TMR2=ON Pre=1:16
    movwf t2con         ;Pone el registro T2CON registe
    movlw b'00001100'   ;CCP1XY=0 CCP1M=1100(PWM)
    movwf ccp1con       ;Pone CCP1CON r

;*** Inicia modo de comparación
    clrf  tmr1h         ;Limpia TMR1H
    clrf  tmr1l         ;Limpia TMR1L
    movlw h'09'         ;H'9C4'=2500
    movwf ccpr2h        ;Pone el registro CCPR2H
    movlw h'c4'         ;2500*0.4usec = 1msec
    movwf ccpr2l        ;Pone el registro CCPR2L
    movlw b'00000001'   ;Pre=1:1 TMR1=Int TMR1=ON
    movwf t1con         ;Pone T1CON
    movlw b'00001011'   ;CCP2M=1011(Compara)
    movwf ccp2con       ;Pone CCP2CON

;*** Control de interrupcion
    bsf   status,rp0    ;Cambia al banco 1
    movlw b'00000001'   ;CCP2IE=Enable
    movwf pie2          ;Pone el registro PIE2
    bcf   status,rp0    ;Canbia al banco 0
    movlw b'11000000'   ;GIE=ON PEIE=ON
    movwf intcon        ;Pone INTCON

wait
    goto  $             ;Espera

;********** Proceso de interrupcion **********
int     clrf  pir2      ;Limpia bandera de interrupcion

ad_check
    btfsc adcon0,go     ;A/D termino?
    goto  ad_check      ;No.
    movf  adresh,w      ;Lee el registro ADRESH
    movwf ccpr1l

    retfie

    end
```

6.7. Práctica No. 6. Medición de temperatura mediante el PIC16F876.

Objetivos.

- Utilizar el recurso de conversión analógico digital de este modelo de PIC para medir temperatura ambiente mediante el sensor LM35D.
- Visualizar el valor de temperatura mediante un display LCD.
- Fijar valores de temperatura máxima y mínima para activar 2 dispositivos electromecánico. un relevador emulando un sistema de calefacción y un motor haciendo la función de un ventildor.

Material.

- 1 Microcontrolador PIC16F876.
- 1 Cristal de cuarzo a 4 Mhz.
- 1 Relevador a 5VDC, 1 Ampere.
- 1 Motor a 5 VDC estándar.
- 1 C.I. L293D.
- 1 Sensor de temperatura LM35D.
- 1 Transistor BC547.
- 1 Potenciómetro de 10 KΩ.
- 2 Resistencias de 10 KΩ.
- 1 Diodo 1N4004.
- 1 Diodo emisor de luz.

Uno de los problemas con los que mayor frecuencia nos enfrentamos, es el de traducir señales analógicas en digitales, es decir, manipular algunos parámetros como pueden ser temperatura, presión, luminosidad, etc.
Como se había mencionado anteriormente, uno de los recursos que incorporan los PIC16F87X es el convertidor analógico – digital.

El problema que se plantea a continuación para realizar la práctica es el de tomar lectura de la temperatura mediante un sensor LM35D. Estos valores se visualizaran en un display LCD. Además de lo anterior, se fijaran un par de valores de temperatura (Máximo y

Mínimo) para los cuales se accionaran ciertos sistemas: para cuando el valor de temperatura pase por debajo del valor mínimo se accionará un relevador que emulara la puesta en marcha de la calefacción; para cuando el valor de temperatura exceda el valor máximo se encenderá un motor que emula a un ventilador.

Sensor LM35D.

Este sensor es uno de los más ampliamente utilizados en algunos diseños. La decisión de usar este sensor se baso en que es uno de los más sencillos de utilizar, además es relativamente barato y siempre está disponible en el mercado.

SENSOR	RANGO	INCREMENTO	PROPIEDADES
LM35D	0 – 100 grados C.	10 mV/grado C.	

Tabla 6.4 Valores del sensor LM35D

Además de las características mostradas en la tabla, tenemos que:

- Su tensión de funcionamiento Vs está comprendido entre +4 VDC y +30 VDC.
- Su precisión es de $\pm 0.9^{\circ}C$.

Figura 6.14

La patita VS+ se debe conectar a una tensión comprendida entre +4 y +30 VDC. VOUT es la salida que otorga el sensor que en este caso se conectará al microcontrolador.

El relevador es estándar *(SUN HOLD)* con caracteristicas de 5VDC y una corriente de 1 ampere.

El motor motor es de C:C. a 6 VDC.

Estos dispositivos se tomaron de forma aleatoria solo para realizar la emulación de un sistema de calefacción y un ventilador.

Conexiones.

La salida del sensor de temperatura se conecta a la patita RA5 del PIC. se debe de tomar en cuenta que esta señal proveniente del sensor es analógica y a la patita a la que se va a introducir es un canal de conversión A/D de microcontrolador. La tensión de referencia del convertidor puede ser la propia de alimentación de PIC (5 VDC). esta tensión se introduce por el pin RA3.

El relevador es conectado en RC0 y el motor en RC1 y RC2. El display inteligente ocupa las líneas desde RB0 a RB7para datos y RA0, RA1 y RA2 para control.

El encendido y apagado de los sistemas de ventilación depende de los valores fijados como máximo y mínimo en el programa principal.

En este caso, se usa este modelo de PIC por tener incluido (como ya se había dicho) un ADC. Si se usará el PIC16F84, se tendría que agregar circuitería externa y hacer el tratamiento apropiado con la señal analógica. Sin embargo con el 876 la conversión se hace por software.

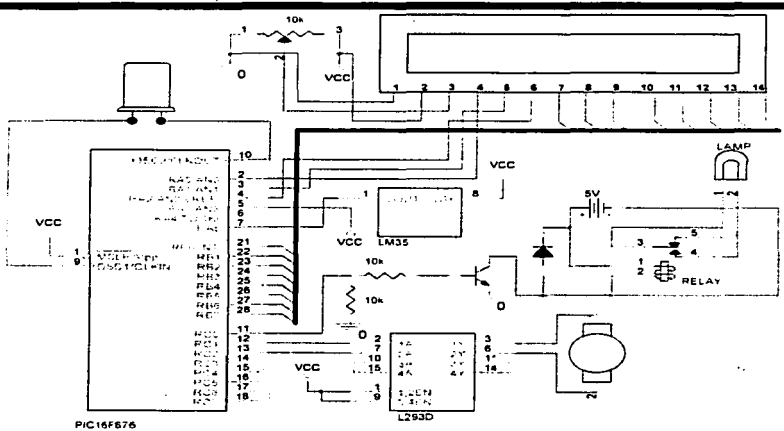


Figura 6.15

Como se puede ver en el esquemático, se conecto una carga a la salida del relevador, esto para comprobar que efectivamente se acciona este dispositivo.

Programa.


```
                list            p=16f876
                radix           hex
                include         "p16f876.inc"
                __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC
T_MIN           EQU             0x25            ;Temperatura minima
T_MAX           EQU             0x35            ;temperatura máxima
Lcd_var         EQU             0x20            ;Variables para LCD
DUTY_H          EQU             0x22            ;Ancho de pulso
DUTY_L          EQU             0x23
DATOA_H         EQU             0x24            ;Multiplicando de 16 bits
DATOA_L         EQU             0x25
```

```
DATOB_H     EQU     0x26        ;Multiplicando de 16 bits
DATOB_L     EQU     0x27
DATOD_H     EQU     0x28        ;Resultado de 32 bits
DATOD_L     EQU     0x29
DATOC_H     EQU     0x2A
DATOC_L     EQU     0x2B
CONTADOR    EQU     0x2C
BCD_2       EQU     0x2D        ;Variables BCD
BCD_1       EQU     0x2E
BCD_0       EQU     0x2F
TEMPORAL    EQU     0x30
DELAY       EQU     0x31        ;Variable de temporización

            ORG     0x00        ;Vector de reset

            goto    INICIO

            ORG     0x05

            include "lcd_cxx.inc"

MUL16x16    movlw   .16         ;Inicia contador
            movwf   CONTADOR
            clrf    DATOD_H             ;Borra parte alta del resultado

            clrf    DATOD_L

MUL_LOOP    rrf     DATOB_H,F
            rrf     DATOB_L,F
            btfss   STATUS,C
            goto    NO_SUMA
            movf    DATOA_L,W
            addwf   DATOD_L,F
            btfsc   STATUS,C
            incf    DATOD_H,F
            movf    DATOA_H,W
            addwf   DATOD_H,F
NO_SUMA     rrf     DATOD_H,F
            rrf     DATOD_L,F
            rrf     DATOC_H,F
            rrf     DATOC_L,F
            decfsz  CONTADOR,F
            goto    MUL_LOOP
            return

;**********Conversión de numero binario de 16 bits a 5 digitos BCD****

BITS16_BCD  bcf     STATUS,C
            clrf    CONTADOR
            bsf     CONTADOR,4
            clrf    BCD_0
            clrf    BCD_1
            clrf    BCD_2
LOOP_16     rlf     DATOC_L,F
            rlf     DATOC_H,F
```

```
                rlf     BCD_2,F
                rlf     BCD_1,F
                rlf     BCD_0,F
                decfsz  CONTADOR,F
                goto    AJUSTE
                return
AJUSTE          movlw   BCD_2
                movwf   FSR
                call    AJUSTE_BCD
                incf    FSR,F
                call    AJUSTE_BCD
                incf    FSR,F
                call    AJUSTE_BCD
                goto    LOOP_16
AJUSTE_BCD      movf    INDF,W
                addlw   0x03
                movwf   TEMPORAL
                btfsc   TEMPORAL,3
                movwf   INDF
                movf    INDF,W
                addlw   0x30
                movwf   TEMPORAL
                btfsc   TEMPORAL,7
                movwf   INDF
                return
```



```
;*********Inicio de programa principal***********
INICIO          clrf    PORTB           ;Limpia salidas
                clrf    PORTA
                clrf    PORTC
                bsf     STATUS,RP0      ;Cambia al banco 1
                bcf     STATUS,RP1
                movlw   b'00000110'     ;Puerto A E/S digitales
                movwf   ADCON1
                clrf    TRISB           ;Salidas
                clrf    TRISA           ;Salidas
                clrf    TRISC           ;Salidas
                movlw   b'11000111'     ;Configuracion del TMR0
                movwf   OPTION_REG
                bcf     STATUS,RP0      ;Cambio al banco 0

                call    UP_LCD          ;Configuración de lineas para display

                call    LCD_INI         ;Inicializacion de display, cursor on
                movlw   b'00001100'
                call    LCD_REG
BUCLE           bsf     STATUS,RP0      ;Banco 1
                bcf     STATUS,RP1
                movlw   b'00101000'     ;RA3 y RA5 entradas
                movwf   TRISA
                movlw   b'10000001'
                movwf   ADCON1          ;RA3 y RA5 entradas analógicas
                bcf     STATUS,RP0      ;Banco 0
```

```
                movlw       b'10100001'     ;Configuracion y activacion del convertidor
                movwf       ADCON0

                movlw       .20
                movwf       DELAY
DELAY_1         movlw       ~.195
                movwf       TMR0
                bcf         INTCON,T0IF

DELAY_2         clrwdt
                btfss       INTCON,T0IF
                goto        DELAY_2
                decfsz      DELAY,F
                goto        DELAY_1

                bcf         PIR1,ADIF       ;restaurar bandera del convertidor

                bsf         ADCON0,GO       ;Iniciar conversion

ADC_WAIT        btfss       PIR1,ADIF
                goto        ADC_WAIT


                bsf         STATUS,RP0      ;Banco 1
                bcf         STATUS,RP1
                movf        ADRESL,W
                bcf         STATUS,RP0
                movwf       DATOB_L
                movf        ADRESH,W
                movwf       DATOB_H
                movlw       .243
                movwf       DATOA_L
                clrf        DATOA_H
                call        MUL16x16
                call        BITS16_BCD

                bsf         STATUS,RP0      ;banco 1
                bcf         STATUS,RP1
                movlw       b'00000110'     ;Puerto A digital
                movwf       ADCON1
                bcf         STATUS,RP0      ;Banco 0

                call        UP_LCD          ;Configura LCD
                movlw       0x80
                call        LCD_REG
                movf        BCD_0,W                 ;Visualiza decenas de grado
                andlw       0x0f
                iorlw       0x30
                call        LCD_DATO
                swapf       BCD_1,W                 ;Visualiza unidades de grado
                andlw       0x0f
                iorlw       0x30
                call        LCD_DATO
                movlw       '.'             ;Visualiza punto decimal
                call        LCD_DATO
```

```
                movf        BCD_1,W             ;Visualiza decimas de grado
                andlw       0x0f
                iorlw       0x30
                call        LCD_DATO
                movlw       ' '                 ;Visualiza espacio en blanco
                call        LCD_DATO
                movlw       0xdf                ;Vsualiza simbolo de grados
                call        LCD_DATO
                movlw       'C'                 ;visualiza C
                call        LCD_DATO

                swapf       BCD_0,W
                andlw       0xf0
                movwf       TEMPORAL
                swapf       BCD_1,W
                andlw       0x0f
                iorwf       TEMPORAL,F

;*********Se compara temperatura con maximo y minimo para activar relevador o motor**********

                movlw       T_MAX
                subwf       TEMPORAL,W
                btfsc       STATUS,C            ;¿Mayor que la maxima?
                goto        SUP_MAX             ;si
                bcf         PORTC,2             ;Deshabilitar motor

                movlw       T_MIN
                subwf       TEMPORAL,W
                btfss       STATUS,C            ;¿Inferior a la minima?
                goto        INF_MIN             ;si
                bcf         PORTC,0             ;No, desconectar relevador

                goto        BUCLE
SUP_MAX         bsf         PORTC,2             ;Encender motor
                goto        BUCLE
INF_MIN         bsf         PORTC,0             ;Encender relevador
                goto        BUCLE

        END
```

FALTA

PAGINA

134

CONCLUSIONES.

Generalmente los temas óptimos en los que se deben basar la tesis en licenciaturas relacionadas con cualquiera de las ramas de la Ingeniería, deben ser versátiles e innovadores o al menos destacables en lo que proponen. Ciertas condiciones ocasionan que los temas no alcancen un grado lo suficientemente alto en lo que se refiere a propuestas realmente consistentes.

Los temas que puedan ser generados dentro del ámbito académico, deben contener al menos las bases para la generación de nuevas ideas para un posible tema de estudio posterior, aplicaciones dentro de entornos externos (industria) o que ayuden al desarrollo de estrategias de estudio.

Actualmente las técnicas de investigación en temas de electrónica, no solo abarcan proyectos en laboratorios, si no que también hay que tener en consideración nuevos mecanismos de estudio y de aplicaciones, sobretodo el surgimiento de nuevas tecnologías que demandan más investigación. Estas técnicas se refieren al conocimiento de hardware (implementación de sistemas cada día más complejo) y software (programación, que hoy en día, cada vez son más los circuitos que demandan el conocimiento de un lenguaje).

En este caso, el estudio de los microcontroladores es necesario considerando las demandas que se tienen hoy en día la industria, además de que este campo de estudio se puede utilizar para generar nuevas propuestas en aplicaciones más específicas dentro de proyectos académicos.

El hecho de haber escogido a los microcontroladores PIC de Microchip, se basó en la idea de realizar un juego de prácticas de bajo costo y sencillas, además de aprovechar los recursos de los que disponen estos dispositivos, los cuales son los adecuados para estos fines.

El lenguaje propio de programación de estos microcontroladores así como su repertorio de instrucciones es relativamente sencillo de comprender y manejar, mediante el cual se pueden generar secuencias de programación tan elaboradas tanto como el ingenio del programador llegue, donde se consideren diseños de distintos tipos.

Del planteamiento proporcionado en esta tesis, pueden derivarse diseños más hábiles en el campo de los microcontroladores, y no solo de los PIC's, sino que teniendo en cuenta que el funcionamiento en la mayoría de los microcontroladores es similar, independientemente del fabricante, pueden usarse una infinidad de dispositivos como actuadores.

Además del avance tecnológico a pasos agigantados, los nuevos microcontroladores que vayan surgiendo comprenderán en su estructura distintos y mejores recursos, sin embargo, será de relativa importancia contar con un antecedente que fomente la comprensión del tema.

Se debe admitir que durante el desarrollo de este trabajo, se presentaron problemas en diversos aspectos, pero, como en toda investigación la idea primordial es concluirla presentando resultados (independientemente de que sean favorables o no); en este caso específicamente se llegó a una comprensión más concreta sobre el tema generando prácticas que realmente puedan ser implementadas dentro de un laboratorio.

Esta tesis esta orientada en dar una introducción en el tema de microcontroladores, y se espera que sirva como un antecedente a los alumnos de electrónica.

APÉNDICE A

LA FAMILIA DE MICROCONTROLADORES PIC

PIC12C5XX Family	8-Pin, 8-Bit CMOS Microcontrollers
PIC12CE5XX Family	8-Pin, 8-Bit CMOS Microcontrollers with EEPROM Data Memory
PIC12C67X Family	8-Pin, 8-Bit CMOS Microcontrollers with A/D Converter
PIC12CE67X Family	8-Pin, 8-Bit CMOS Microcontrollers with A/D Converter and EEPROM Data Memory
PIC14000 Family	28-Pin Programmable Mixed Signal Controller
PIC16C5X Family & PIC16HV540	EPROM/ROM-Based 8-Bit CMOS Microcontroller Series
PIC16C55X Family	EPROM-Based 8-Bit CMOS Microcontrollers
PIC16C6X Family	8-Bit CMOS Microcontrollers
PIC16C64X & PIC16C66X Families	8-Bit EPROM Microcontrollers with Analog Comparators
PIC16X62X Family	18-Pin EPROM-Based 8-Bit CMOS Microcontrollers
PIC16CE62X Family	8-Bit CMOS Microcontrollers with Analog Comparators and EEPROM Data Memory
PIC16C7X Family	8-Bit CMOS Microcontrollers with A/D Converter
PIC16C71X Family	20, 28 and 40-Pin, 8-Bit CMOS Microcontrollers with 12-Bit

PIC16C745/765 Family	**8-Bit CMOS Microcontrollers with A/D Converter for USB, PS/2 and Serial Device Applications**
PIC16C77X Family	**20, 28 and 40-Pin, 8-Bit CMOS Microcontrollers with 12-Bit A/D Converter**
PIC16F87X Family	**28/40-Pin, 8-Bit CMOS FLASH Microcontrollers**
PIC16X8X Family	**8-Bit CMOS Flash/EEPROM Microcontrollers**
PIC16C9XX Family	**8-Bit CMOS Microcontroller with LCD Driver**
PIC17C4X Family	**High-Performance 8-Bit CMOS EPROM/ROM Microcontrollers**
PIC17C7XX Family	**High-Performance 8-Bit CMOS EPROM Microcontrollers**
PIC18CXXX Family	**Enhanced Architecture 8-Bit Microcontrollers**

PICmicro MCU Overview and Roadmap

Microchip PICmicro MCUs combine high-performance, low-cost, and small package size, offering the best price/performance ratio in the industry. More than 120 million of these devices ship each year to cost-sensitive consumer products, computer peripherals, office automation, automotive control systems, security and telecommunication applications.

Microchip offers four families of 8-bit MCUs to best fit your needs: PIC16C5X 12-bit program word, PIC16CXXX 14-bit program word, PIC17CXXX 16-bit program word, PIC18CXXX enhanced 16-bit program word and PIC12CXXX

8-pin 12-bit/14-bit program word MCU families.

All families offer OTP, low-voltage and low-power options, with a variety of package options. Selected members are available in ROM, EEPROM or reprogrammable Flash versions.

The widely-accepted PIC16C5X, PIC16CXXX and PIC17CXXX MCU families employ a modified RISC architecture. Today, the industry's first 8-pin MCU family – the PIC12CXXX, joins these families. The PIC12CXXX family combines the 8-bit high-speed RISC architecture of the PICmicro MCUs with the smallest footprint MCU. Microchip pioneered the use of RISC architecture to obtain high speed and instruction efficiency.

PIC12CXXX: 8-Pin, 8-Bit Family

The PIC12CXXX family packs Microchip's powerful RISC-based PICmicro architecture into 8-pin DIP and SOIC packages. These PIC12CXXX products are available with either a 12-bit or 14-bit wide instruction set, a low operating voltage of 2.5V, small package footprints, interrupt handling and a deeper hardware stack. All of these features provide an intelligence level not previously available in applications because of cost or size considerations.

PIC16C5X: 12-Bit Architecture Family

The PIC16C5X is the well-established base-line family that offers the most cost-effective solution. These PIC16C5X products have a 12-bit wide instruction set and are currently offered in 18-, 20- and 28-pin packages. In the SOIC and SSOP packaging options, these are among the smallest footprint MCUs. Low-voltage operation, down to 2.0V for OTPs, makes this family ideal for battery operated applications. Additionally, support is available for PIC16HV5XX that can operate directly from a battery at wide

voltage ranges up to 15 volts.

PIC16CXXX: 14-Bit Architecture Family

With the introduction of new PIC16CXXX family members, Microchip now provides the industry's highest performance Analog-to-Digital Converter capability at 12-bits for an 8-bit MCU. The PIC16CXXX family offers a wide-range of options, from 18-pin to 68-pin packages as well as low to high levels of peripheral integration. This family has a 14-bit wide instruction set, interrupt handling capability and a deep, 8-level hardware stack. The PIC16CXXX family provides the performance and versatility to meet the more demanding requirements of today's cost-sensitive marketplace for mid-range 8-bit applications.

PIC17CXXX: 16-Bit Architecture Family

The PIC17CXXX family offers the world's fastest execution performance of any 8-bit MCU family in the industry. The PIC17CXXX family extends the PICmicro MCU's high-performance RISC architecture with a 16-bit instruction word, enhanced instruction set and powerful vectored interrupt handling capabilities. A powerful array of precise on-chip peripheral features provides the performance for the most demanding 8-bit applications.

PIC18CXXX: 16-Bit Architecture Family

The PIC18Cxx2 is a family of high performance, CMOS, fully static, 16-bit MCUs with integrated analog-to-digital (A/D) converter.

All PIC18Cxx2 MCUs employ an advanced RISC architecture. The PIC18Cxxx has enhanced core features, 32 level-deep stack, and multiple internal and external interrupts sources. The separate instruction and data busses of the Harvard architecture allow a 16-bit wide instruction word with the separate 8-bit wide data. The two-stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches, which require two cycles. A

total of 68 instructions (reduced instruction set) are available. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance of 10MIPs for an 8-bit MCU.

The PIC18CXXX family has special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. These include programmable Low Voltage Detect (LVD) and Brown-Out Detect (BOR).

APÉNDICE B

HOJAS TÉCNICAS DE DATOS

PIC16F84

PIC16F84 Rev. A Silicon Errata Sheet

The PIC16F84 Rev. A parts you have received conform functionally to the Device Data Sheet (DS30430C), except for the anomalies described below.

All the problems listed here will be addressed in future revisions of the PIC16F84 silicon.

1. Module: CPU (STATUS bit)

   The operation of the power-down (PD) bit in the STATUS register may not function correctly for temperatures below -20°C.

   Work Around

   None.

2. Module: Data EEPROM

   Do not perform a modify (set or clear a bit) of the EECON1 register one instruction cycle after an EEPROM read. This will corrupt the EEDATA register.

   Example:

```
BSF     EECON1, RD
BCF     EECON1, EEIF
```

   Work Around

   Use either of the following two code segment in place of the above example.

```
BSF     EECON1, RD
NOP
BCF     EECON1, EEIF
```

   or

```
BCF     EECON1, EEIF
BSF     EECON1, RD
```

3. Module: Timer0

   The TMR0 register may increment when the WDT postscaler is switched to the Timer0 prescaler. If TMR0 = FFh, this will cause TMR0 to overflow, setting T0IF.

   Work Around

   Follow the following sequence:

a) Read the 8-bit TMR0 register into the W register
b) Clear the TMR0 register
c) Assign WDT postscaler to Timer0
d) Write W register to TMR0

Note: As with any windowed EPROM device, please cover the window at all times, except when erasing.

Clarifications/Corrections to the Data Sheet:

In the Device Data Sheet (DS30450C), the following clarifications and corrections should be noted.

1. Module: Data EEPROM

In the PIC16F8A Data Sheet (DS30430B), the following clarifications and corrections should be noted.

a) Erase/Write Cycle Time for Data EEPROM should be changed from 10 ms maximum to 20 ms maximum, as shown in Table 1.

TABLE 1: DC SPECIFICATION LIMITS THAT VARY FROM DATA SHEET

Parameter No.	Sym.	Characteristic	Actual Data		Data Sheet		Units
			Typ	Max	Typ	Max	
D122	TDEW	Erase/Write Cycle Time	10	20*	—	10	ms

* This parameter is characterized but not tested.

2. Module: Device IDD

a) The maximum device IDD in the LP oscillator mode (at 32kHz, 2.0V, and WDT disabled) should be changed from 32 µA maximum to 45 µA maximum, as shown in Table 1.

TABLE 2: DC SPECIFICATION LIMITS THAT VARY FROM DATA SHEET

Parameter No.	Sym.	Characteristic	Actual Data		Data Sheet		Units
			Typ	Max	Typ	Max	
D014	IDD	Supply Current	15	45	15	32	µA

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

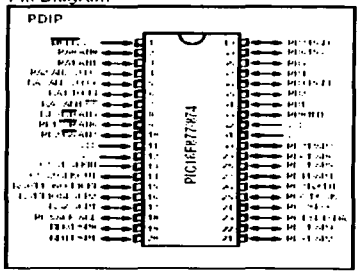
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pin out compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, Indirect and Relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS FLASH/EEPROM technology
- Fully static design
- In-circuit Serial Programming (via two pins)
- Single 5V In-Circuit Serial Programming capability
- In-circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

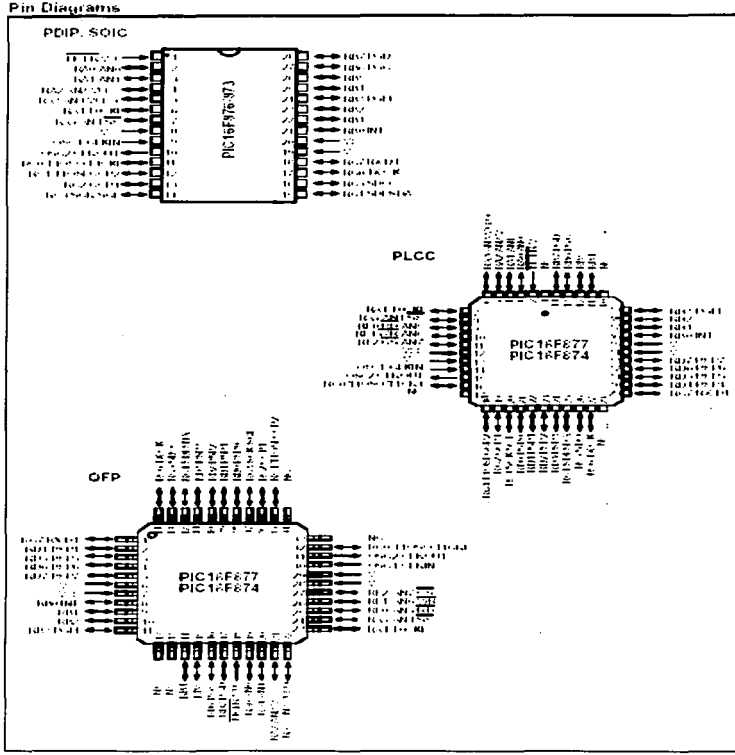
PDIP



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two capture, compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Pin Diagrams



Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory 14-bit words	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications		PSP		PSP
10-bit Analog-to-Digital Module	5 Input Channels	8 Input Channels	5 Input Channels	8 Input Channels
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions

MICROCHIP

PIC16F84 — PIC16F84A Migration

DEVICE MIGRATIONS

This document is intended to describe the functional differences and the other considerations when migrating from one device to another. Table 1 lists the functional differences that exist between the PIC16F84 and the PIC16F84A. Table 2 shows the differences in the electrical specifications between these devices.

Note:	This device has been designed to perform to the parameters of its data sheet. It has been tested to an electrical specification designed to determine its reliability when operating at these parameters. Due to process differences in the manufacture of this device, this device may have different performance characteristics than its earlier version. These differences may cause this device to perform differently in your application than the earlier version of this device.

Note:	The user should verify that the device oscillator starts and performs as expected. Adjustment of the loading capacitor values used in the oscillator circuit may be required.

TABLE 1: PIC16F64 - PIC16F64A FUNCTIONAL DIFFERENCES

No.	Module	Differences from PIC16F64	H/W	S/W
1	Oscillator	The PIC16F84 oscillator can run up to 10 MHz. The PIC16F84A oscillator can run up to 20 MHz.	Yes.	Yes.

Legend: H/W = Feature, may need to be tested in the application circuit.
S/W = Feature, may need to be tested in the user program.

OSCILLATOR

The PIC16F84A can have crystals up to 20 MHz, resulting in double the execution speed. For higher frequency crystals, other load capacitors may be required. Ceramic resonators often require additional capacitors as well. These load capacitors typically may need to be adjusted for the higher speed crystal. Full verification of oscillator operation at the same speed as the application should be done when migrating from the PIC16F84 to the PIC16F84A.

TABLE 2: PIC16F64 - PIC16F84A SPECIFICATION DIFFERENCES

Param No.	Symbol	Characteristic		PIC16F64			PIC16F84A			Units
				Min	Typ†	Max	Min	Typ†	Max	
Core										
	FOSC	External CLKIN Frequency (DC – 4 MHz oscillator/clock input) External CLKIN Frequency (DC – 10 MHz oscillator/clock input)		DC 1		4 10	DC 1		20 20	MHz MHz
D001 D001A	VDD	Supply Voltage (XT, RC, LP oscillator) Supply Voltage (HS oscillator)		4.0 4.5		6.0 6.0	4.0 4.5		5.5 5.5	V V
40	TDLY	Power-up Timer Period		1			2			ms
D004A	VDD	VDD rise rate to ensure internal Power-on Reset signal (POR)		N A	N A	N A	TBD			V/ms
D010 D010A	IDD	Supply current during XT, RC, LP oscillator configuration (FOSC = 4 MHz, VDD = 5.5V)			2	4		2.0	4	mA
D013 D013A	IDD	Supply current during HS oscillator configuration (FOSC = 10 MHz, VDD = 5.5V)	XT, RC, LP (16F84) (FOSC = 10 MHz)		2	4				mA
			HS (16F84A) (FOSC = 20 MHz)					10	20	mA
D021 D021A	IPD	Power-down current (WDT disabled, VDD = 4.0V) (WDT disabled, VDD = 4.5V)	Commercial Industrial		10 10	14 16		1 1.0	7.0	µA µA
D022	ΔIWDT	Watchdog Timer Current: Differential current (WDT enabled minus WDT disabled)	Commercial Industrial	N A N A	N A N A	N A N A		4.0 4.0	20 25	µA µA
D030 D030A D031	VIH	Input High Voltage I/O Ports: with TTL buffer with Schmitt Trigger buffer MCLR, RA4/T0CKI	4.5V ≤ VDD ≤ 5.5V VDD = 4.0V to 4.5V 4.5V ≤ VDD ≤ 5.5V VDD = 4.0V to 4.5V	2.0 (0.8)VDD (0.8)VDD		VDD VDD VDD	2.0 (0.25VDD+0.8) 0.8VDD		VDD VDD VDD	V V V
D032		OSC1 (XT, RC and LP configurations)		0.85VDD		VDD	0.8VDD		VDD	V
D033 D033A		OSC1 (HS configuration)		0.7VDD N A	N A	VDD N A	0.7VDD 0.7VDD		VDD VDD	V V
D034	VHYS	Hysteresis of Schmitt Trigger inputs		0.15				0.1		V
EEPROM Data Memory										
D120	VDRW	VDD for read/write		VMIN		6.0	VMIN		5.5	V
D122	TDEW	Erase/Write cycle time			10	20		4	8	ms
FLASH Program Memory										
D130	VPR	VDD for read		VMIN		6.0	VMIN		5.5	V
D132	TDEW	Erase/Write cycle time			10			4	8	ms

† Data in Typ column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: These must always be the lesser of the two specifications.

LM34/LM35 Precision Monolithic Temperature Sensors

National Semiconductor
Application Note 460
October 1985

Introduction

[illegible]

Forerunners to the LM34

[illegible]

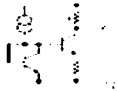


FIGURE 1.

[illegible]

FAIRCHILD
SEMICONDUCTOR™

Discrete POWER & Signal Technologies

BC547
BC547A
BC547B
BC547C

E
B
C

TO-92

NPN General Purpose Amplifier

This device is designed for use as general purpose amplifiers and switches requiring collector currents to 500 mA. Sourced from Process 10. See PN100A for characteristics.

Absolute Maximum Ratings*

Symbol	Parameter	Value	Units
V_{CEO}	Collector-Emitter Voltage	45	V
V_{CBO}	Collector-Base Voltage	50	V
V_{EBO}	Emitter-Base Voltage	6.0	V
I_C	Collector Current - Continuous	500	mA
T_J, T_{stg}	Operating and Storage Junction Temperature Range	-55 to +150	°C

* These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

NOTES:
1) These ratings are based on a maximum junction temperature of 150 degrees C.
2) These are steady state limits. The factory should be consulted on applications involving pulsed or low duty cycle operations.

Thermal Characteristics

Symbol	Characteristic	Max		Units
		BC547 A B C		
P_D	Total Device Dissipation Derate above 25°C	625 5.0		mW mW/°C
$R_{\theta JC}$	Thermal Resistance, Junction to Case	83.3		°C/W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient	200		°C/W

NPN General Purpose Amplifier

(continued)

Electrical Characteristics

Symbol	Parameter	Test Conditions	Min	Max	Units
	Collector-Emitter Breakdown Voltage				
	Collector-Base Breakdown Voltage				
	Collector-Base Breakdown Voltage				
	Emitter-Base Breakdown Voltage				
	Collector Cut-off Current				
	DC Current Gain				
	Collector-Emitter Saturation Voltage				
	Base-Emitter On Voltage				
	Small-Signal Current Gain				
NF	Noise Figure				

FAIRCHILD

April 1996

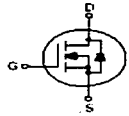
BS170 / MMBF170
N-Channel Enhancement Mode Field Effect Transistor

General Description

These N-Channel enhancement mode field effect transistors are produced using Fairchild's proprietary, high cell density, DMOS technology. These products have been designed to minimize on-state resistance while provide rugged reliable, and fast switching performance. They can be used in most applications requiring up to 500mA DC. These products are particularly suited for low voltage, low current applications such as small servo motor control, power MOSFET gate drivers, and other switching applications.

Features

- High density cell design for low R
- Voltage controlled small signal switch.
- Rugged and reliable.
- High saturation current capability.



Absolute Maximum Ratings

Symbol	Parameter	BS170	MMBF170	Units
V_{DS}	Drain-Source Voltage	60		V
V_{DG}	Drain-Gate Voltage ($R_{GS} \leq 1M\Omega$)	60		V
V_{GS}	Gate-Source Voltage	±20		V
I_D	Drain Current - Continuous	500	500	mA
	- Pulsed	1200	800	
P_D	Maximum Power Dissipation	830	300	mW
	Derate Above 25°C	6.6	2.4	mW/°C
T_J, T_{STG}	Operating and Storage Temperature Range	-55 to 150		°C
T_L	Maximum Lead Temperature for Soldering Purposes, 1/16" from Case for 10 Seconds	300		°C
THERMAL CHARACTERISTICS				
$R_{\theta JA}$	Thermal Resistance, Junction-to-Ambient	150	417	°C/W

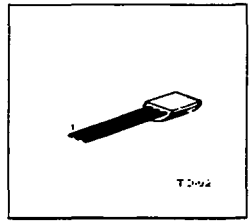
Electrical Characteristics (T_A = 25°C unless otherwise noted)

Symbol	Parameter	Conditions	Type	Min	Typ	Max	Unit
OFF CHARACTERISTICS							
BV_{DSS}	Drain-Source Breakdown Voltage	V_{GS} = 0 V, I_D = 100 µA	All	60			V
I_{DSS}	Zero Gate Voltage Drain Current	V_{DS} = 25 V, V_{GS} = 0 V	All			0.5	µA
I_{GSSF}	Gate - Body Leakage, Forward	V_{GS} = 15 V, V_{DS} = 0 V	All			10	nA
ON CHARACTERISTICS (Note)							
$V_{GS(th)}$	Gate Threshold Voltage	V_{DS} = V_{GS}, I_D = 1 mA	All	0.8	2.1	3	V
$R_{DS(on)}$	Static Drain-Source On-Resistance	V_{GS} = 10 V, I_D = 200 mA	All		1.2	5	Ω
g_{FS}	Forward Transconductance	V_{DS} = 10 V, I_D = 200 mA	BS170		320		mS
		$V_{DS} \geq 2 V_{DS(on)}$, I_D = 200 mA	MMBF170		320		
DYNAMIC CHARACTERISTICS							
C_{iss}	Input Capacitance	V_{DS} = 10 V, V_{GS} = 0 V, f = 1.0 MHz	All		24	40	pF
C_{oss}	Output Capacitance		All		17	30	pF
C_{rss}	Reverse Transfer Capacitance		All		7	10	pF
SWITCHING CHARACTERISTICS (Note)							
t_{on}	Turn-On Time	V_{DD} = 25 V, I_D = 200 mA, V_{GS} = 10 V, R_{GEN} = 25 Ω	BS170			10	ns
		V_{DD} = 25 V, I_D = 500 mA, V_{GS} = 10 V, R_{GEN} = 50 Ω	MMBF170			10	
t_{off}	Turn-Off Time	V_{DD} = 25 V, I_D = 200 mA, V_{GS} = 10 V, R_{GEN} = 25 Ω	BS170			10	ns
		V_{DD} = 25 V, I_D = 500 mA, V_{GS} = 10 V, R_{GEN} = 50 Ω	MMBF170			10	

Note: Pulse test: pulse width ≤ 300 µs, duty cycle ≤ 2.0%.

UTC2SC1815 NPN EPITAXIAL SILICON TRANSISTOR

AUDIO FREQUENCY AMPLIFIER
HIGH FREQUENCY OSC NPN
TRANSISTOR



1 EMITTER 2 COLLECTOR 3 BASE

FEATURES

*Collector-Emitter voltage
V(BR)CEO=50V
*Collector current up to 150mA
*High hFE linearity
*Complimentary to 2SA1015

ABSOLUTE MAXIMUM RATINGS: (Ta=25°C unless otherwise specified)

PARAMETER	SYMBOL	RATING	UNIT
Collector-base voltage	$V_{(BR)CBO}$	60	V
Collector-emitter voltage	$V_{(BR)CEO}$	50	V
Emitter-base voltage	$V_{(BR)EBO}$	5	V
Collector dissipation Ta=25°C	P_C	400	mW
Collector current	I_C	150	mA
Base current	I_B	50	mA
Junction Temperature	T_J	125	°C
Storage Temperature	T_{stg}	-55°C~+125	°C

ELECTRICAL CHARACTERISTICS: (Ta=25°C unless otherwise specified)

Parameter	Symbol	Test conditions	MIN	TYP	MAX	UNIT
Collector-emitter cut-off current	I_{CBO}	V_{CB}=60V, I_E=0			100	nA
Emitter cut-off current	I_{EBO}	V_{EB}=5V, I_C=0			100	nA
DC current gain note	h_{FE1}	V_{CE}=6V, I_C=2mA	70		700	
	h_{FE2}	V_{CE}=6V, I_C=150mA	25			
Collector-emitter saturation voltage	$V_{CE(sat)}$	I_B=10mA, I_C=100mA		0.1	0.25	V
Base-emitter saturation voltage	$V_{BE(sat)}$	I_B=10mA, I_C=100mA			1.0	V
Current gain bandwidth product	f_T	V_{CE}=10V, I_C=50mA	80			MHz
Output capacitance	C_{ob}	V_{CB}=10V, I_E=0, f=1MHz		2.0	3.5	pF
Noise Figure	NF	I_C=0.1mA, V_{CE}=6V, R_G=10kΩ, f=1kHz		1.0	10	dB

Liquid Crystal Display Model LCD1621
(16 Characters x 2 Lines)

Environmental Specifications

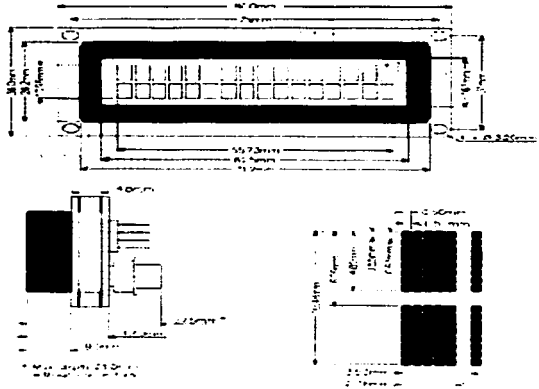
	Standard Temp.	*Extended Temp.*
Operating Temperature	0 to +50 C	-20 to +70 C
Storage Temperature	-20 to +70 C	-30 to +85 C
Operating Relative Humidity	90 max non-condensing	

Electrical Specifications (Ta = 25° C, Vin = 5V)

Supply Voltage	4.75 - 5.25 Vdc (optional +3.15Vdc)
Supply Current	9mA typical
Backlight Supply Current	40mA typical

Optical Characteristics

Number of characters	32 (16 Characters x 2 Lines)
Matrix format	5 x 7 with underline
Display area	62.5 x 16.1mm (XxY)
Character size	2.78 x 4.89mm (XxY), not including underline
Character pitch	3.53mm
Line pitch	6.09mm
Dot size	0.50 x 0.55mm (XxY)
Dot pitch	0.57 x 0.62mm (XxY)
LED Backlight life	100,000 hours typical
Color of illumination	Yellow green



Bibliografía.

- Microcontroladores "PIC". Diseño práctico de aplicaciones.
Autor: José Ma. Angulo U.
Editorial: McGraw Hill.
Segunda edición.
España.

- Microcontroladores "PIC". Diseño práctico de aplicaciones. Segunda parte: PIC16F87X.
Autores: José Ma. Angulo U., Susana Romero Sosa, Ignacio Angulo Ramírez.
Editorial: Mc Graw Hill.
Segunda edición.
España.

- The PIC Microcontrollers.
Autores: Nebojsa Matic, Dragan Andric.
Microchip.
Yugoslavia.

- Microcontroladores PIC16C5X.
Autor: Celestino Benítez Vázquez.
Colegio Oficial de Peritos de Ingenieros Técnicos de Alicante (COPITAL).

- Microcontroladores.
Autor: Vicente Torres.
Universidad Politécnica de Valencia.
Servicio de publicaciones.
España.

- PIC 16/17 Microcontroller Data Book.
Microchip.
1996 – 1997.

- Microcontrollers: A architecture, implementation and programming.
Autores: Kenneth J. Hintz, Daniel Tabak.
Editorial: Mc Graw Hill.

- Microcontrollers and microcomputers: principles of software and hardware engineering.
Autor: Fredrick M. Cady.
Oxford University Press. 1997.

- Microcontroladores de 4 y 8 bits.
Autor: Christian Tavernier.
Editorial: Paraninfo.

- *Microcontroladores PIC.*
Autor: Christian Tavernier.
Editorial: Paraninfo.

Referencias en Internet.

www. microchip.com
www.netcom.es/celes
www.mikroelektronika.co.yu
www.hobby-elec.com
www.tianma.com
www.national.com
www.sagitron.es
www.fairchild.com