UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

MANUAL PARA PROGRAMAR PIC`s ENFOCADO AL LABORATORIO DE MICROPROCESADORES QUE SE IMPARTE EN LA FES ARAGON

T E S I S

P A R A O B T E N E R E L T I T U L O D E

I N G E N I E R O M E C A N I C O E L E C T R I C I S T A

P R E S E N T A :

M A R C O A N T O N I O S O L O R I O A V I L A

ASESOR: ING. ARTURO OCAMPO ALVAREZ

MEXICO 2007

AGRADECIMIENTOS

A mis padres: *Víctor e Irma Lilia muy especialmente por darme la oportunidad de tener una educación, porque cada uno ha sido un motivo de superación, por el amor, el gran apoyo, por ser mis primeros maestros, porque son una parte fundamental en mi vida y gracias a ellos he logrado cumplir una meta muy importante en mi vida, así que esta tesis esta dedicada a ellos con amor y agradecimiento.*

A mi hermana: *Lic. Nancy Gabriela por sus correcciones.*

Al Ing. Marco Antonio Avila Oliva*: Desde que era niño el a sido mi ejemplo a seguir y en momentos difíciles el ha estado conmigo.*

A mi novia: *A la persona que representa el éxito de este trabajo mariana mismo que dedico todo mi amor por su apoyo, decisión y compañía sinónimo de ayuda sin esperar nada a cambio mas que la ilusión de un futuro de felicidad el cual quiero demostrar con este granito de arena que solo ella y yo sabemos lo que significa.*

A mis familiares: *A mi abuela, tíos, primos, sobrinos, y a todos mis amigos que ya son parte de mi familia, no escribo nombres porque no quisiera olvidar a nadie, a todos y cada uno de ustedes por que han estado siempre al pendiente, dándome ánimos a lo largo de mi carrera, por todos sus consejos muchas gracias.*

Al Señor Fidel Arenas Flores: *Que fue una parte clave para el desarrollo de esta tesis, gracias por su paciencia, por brindarme mucho de su valioso tiempo y sabiduría.*

A mi asesor de tesis: *Ing. Arturo Ocampo Álvarez, Por su ayuda para la realización de esta tesis y por el apoyo y la atención, que me brindo durante la carrera.*

Gracias

Marco Antonio Solorio Avila

INDICE GENERAL

Capítulo 2. PRACTICAS PROPUESTAS

Capítulo 3. PROPUESTA DE PRÁCTICAS EXTRAS

INTRODUCCIÓN

Es necesario reconocer que la actualización del plan de estudios de la FES Aragón constituye un problema a nivel académico, se sabe que los laboratorios de electrónica del plantel cuentan ya con algunas carencias; los instructores del plantel deben hacer uso de su ingenio para cubrir dicha situación.

Este trabajo tiene como finalidad ahondar en el diseño de nuevas prácticas del laboratorio de microprocesadores, poniendo énfasis en los microcontroladores PIC´s de microchip, el avance de estos dispositivos abre la posibilidad de que con un solo microcontrolador se puedan realizar todas las prácticas del laboratorio. Este trabajo muestra una alternativa para lograr dicho fin.

El supuesto hipotético que se pretende demostrar con este trabajo de investigación además de aplicar y fundamentar los conocimientos de ingeniería adquiridos a lo largo de mi carrera; es la construcción de material de referencia para académicos y alumnos.

Este trabajo esta dividido en tres capítulos en los cuales se describen la estructura del microprocesador necesario para entender como funciona el microcontrolador, así como sus características; el diseño implementado en hardware/software y finalmente los resultados.

Para el capitulo uno consideré importante explicar ciertos conceptos, que permitan entender como funciona el microprocesador, tomando en cuenta las diferentes partes de un microcontrolador.

En el capitulo dos, se describen las prácticas propuestas; haciendo hincapié en las ventajas que ofrece el uso de un microcontrolador.

Y finalmente el capitulo tres se destinó a proponer practicas extras y a dos guías rápidas del uso del MPLAB y PROTEUS que se utilizaron para lograr nuestros objetivos de comprobar nuestros dispositivos electrónicos.

Considero que mi principal aportación con este trabajo de tesis, es dar algunas herramientas para la actualización del plan de estudios de la FES Aragón ya que dejo material de referencia con la finalidad de que académicos y alumnos puedan utilizarlo.

OBJETIVOS GENERAL

Presentar una visión general acerca de los microcontroladores PIC’s a través de las prácticas del laboratorio de microprocesadores impartidos en el laboratorio L-3 de la FES ARAGON.

Proporcionar al alumno de ingeniería un manual de programación de microcontroladores PIC’s, escogiendo un microcontrolador de costo económico en el mercado local para la programación; así como guía para el uso de un simulador de los microcontroladores.

OBJETIVOS PARTICULAR

Elaborar un C.D que contenga los formatos y material necesario para resolver las prácticas del laboratorio de microprocesadores.

CAPÍTULO 1

1. ESTRUCTURA DEL MICROCONTROLADOR.

La importancia del los microcontroladores hoy en día son sus extensas áreas de aplicaciones, que están presentes en nuestro trabajo, en nuestra casa y en nuestra vida en general. Se pueden encontrar controlando por ejemplo: el funcionamiento de los ratones y teclados de las computadoras, en los teléfonos, en los hornos de microondas y en los televisores de nuestro hogar, se pueden considerar ilimitadas, por tal motivo y, debido a la necesidad de actualizar el plan de estudios de la universidad, decidí enriquecer las practicas del laboratorio de microprocesadores, para que a su vez académicos y alumnos puedan utilizarlo como material de referencia para nuevos proyectos e investigaciones de mayor calidad.

Con el fin de poder ahondar en materia del tema que investigamos es importante tener presentes ciertos conceptos, que nos permitan entender de manera clara los elementos involucrados y la manera en que se interrelacionan. Por esta razón, emplearemos este primer capítulo para presentar los conceptos y fundamentos más importantes para entender cómo funcionan los microcontroladores, sus características generales.

1.1 Microprocesadores.

Esencialmente, un microprocesador (μP) es un circuito de alta escala de integración, compuesto de muchos circuitos más simples como son los Fip-flops, contadores, registros, decodificadores, comparadores, etcétera; todos ellos en un mismo encapsulado, de modo que el microprocesador puede ser considerado un dispositivo lógico de propósito general o universal. Todos los componentes que llevan a cabo físicamente la lógica y operación del microprocesador se denominan hardware. Además existe una lista de instrucciones –con las que se forman programas- que puede ejecutar; éstas constituyen el lenguaje del microprocesador o software.

Es un circuito integrado programable que contiene todos los componentes de un computador. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta ultima característica es la que le confiere la denominación de controlador incrustado

Los pines o terminales de un microprocesador sacan del encapsulado las líneas de sus buses de direcciones, datos y control, para permitir la conexión con memorias, módulos E/S, reloj y alimentación.

Sistema mínimo

Un microprocesador por sí mismo no es capaz de realizar tarea alguna, es necesario hardware de soporte; los elementos mínimos que requiere son:.

- Una fuente de alimentación
- Un circuito de reloj
- Dispositivos de memoria
- Interfaz o módulo de entrada y salida (E/S)

La implementación de todo este hardware constituye lo que se conoce como sistema mínimo, el siguiente diagrama corresponde a un sistema basado en la estructura de Von Newman. Sus bloques básicos son los siguientes.

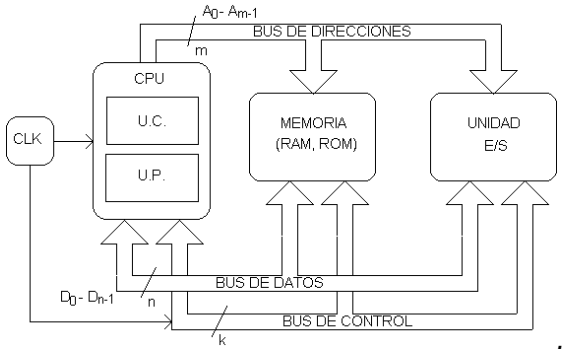


FIG 1.1 ESTRUCTURA DE VON NEWMAN

MICROPROCESADOR O CPU (unidad central de procesamiento), formado por los bloques

principales: Unidad de Control y Unidad de Proceso.

MEMORIA, dispositivos o circuitos donde residen los códigos de las instrucciones del programa y los datos.

MÓDULOS E/S, dispositivos o circuitos encargados de recibir y entregar información entre el **CPU** y la aplicación.
Los tres módulos están conectados entre sí por medio de los ***Buses del sistema*.** Un bus está formado por un conjunto de conductores por los cuales se transmite la información digital en forma de pulsos eléctricos.

1.2 BUSES DEL SITEMA

BUS DE DIRECCIONES: A0-AM-1.- Es el empleado por la CPU para seleccionar la dirección de memoria o el dispositivo de E/S con el cual va a intercambiar información. Es por tanto unidireccional y su tamaño, o número de conductores que lo constituyen, determina la capacidad de direccionamiento de la CPU, que es el máximo número de posiciones de memoria y dispositivos E/S a los que la CPU puede acceder. Para ***m*** líneas la capacidad de direccionamiento será: 2^m localidades.

BUS DE DATOS: D0 – DN-1.- Es el conjunto de conductores a través del cual el µP intercambia información con la unidad de memoria o E/S seleccionado mediante el bus de direcciones. Características:

- Bidireccional: la información puede viajar en los dos sentidos.
- Número de líneas (N): representa la cantidad de bits que se pueden transmitir simultáneamente. Suele denominarse la Palabra del µP.
- Triestado: las líneas del bus de datos deben ser triestado. Las líneas triestado son aquellas capaces de tener tres estados:
    - Estado alto (High, H)

- Estado bajo (Low, L)
- Estado de alta impedancia (High Impedance, Hi-Z)

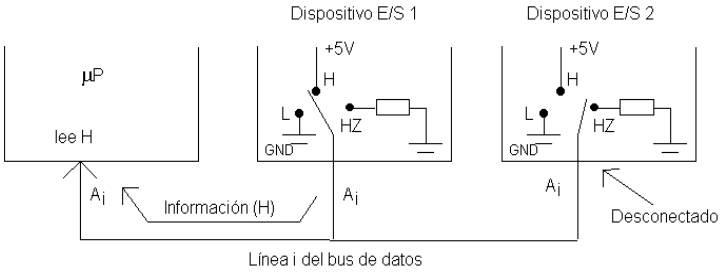


FIG 1.2. BUS DE DATOS

- Mientras el µP se está comunicando con el dispositivo E/S 1, sólo estos dos dispositivos pueden disponer del bus de datos.

- Por lo que el resto de dispositivos conectados físicamente al bus de datos deben permanecer con sus líneas en alta impedancia (Dispositivo E/S 2), para evitar corto-circuitos, alteración de la información, etc.

- La Unidad de Control de la CPU es la que decide qué elemento envía la información y qué elemento la recibe, así como los elementos que deben desconectarse del bus y ponerse en alta impedancia.

1.2.1 BUS DE CONTROL

Está formado por un conjunto de líneas por las que circulan las señales auxiliares de gestión y sincronización del sistema. Las líneas existentes dependen del fabricante del µP y de las funciones que desee implementar. Algunas señales típicas en todos los sistemas son:

- Señal de reloj de sincronización
- Señal de RESET o inicialización
- Señal de lectura/escritura en memoria, etc.

1.3 UNIDAD DE MEMORIA

- Almacenamiento de las instrucciones que constituyen el programa. Las instrucciones se codifican mediante su Código de Operación (OPC), formado por uno o varios Bytes.

- Almacenamiento de los datos y variables del programa. Normalmente agrupados en paquetes de 8 bits (Byte), 16 bits (Word) o 32 bits (Long Word). La velocidad (tiempo de respuesta) de la memoria debe ser compatible con el µP empleado. Normalmente se emplean memorias basadas en semiconductores:

    - RAM, memorias de lectura/escritura para almacenamiento de variables y datos del programa. Son volátiles.

    - ROM (EPROM, EEPROM), memorias de sólo lectura para almacenamiento de programas fijos (aplicaciones, rutinas básicas de sistemas operativos, etc.) y constantes. Son no volátiles.

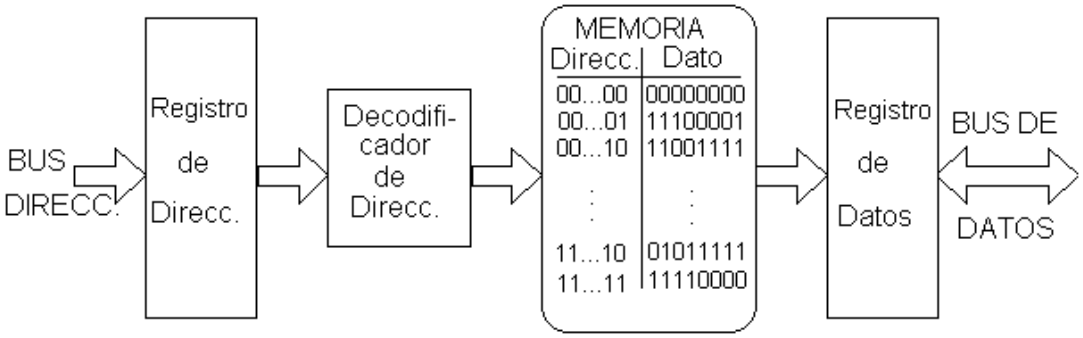


FIG. 1.3 ALMACENAMIENTO DE DATOS

1.4 UNIDAD CENTRAL DE PROCESO (CPU)

Está formada por los siguientes bloques:

- Registros internos
- Unidad de Control
- Unidad de Proceso u Operativa

1.4.1 REGISTROS INTERNOS DE LA CPU

Los registros de la CPU son bloques de biestables que permiten el almacenamiento de los datos básicos con los cuales va a trabajar la CPU durante la ejecución de cada instrucción.

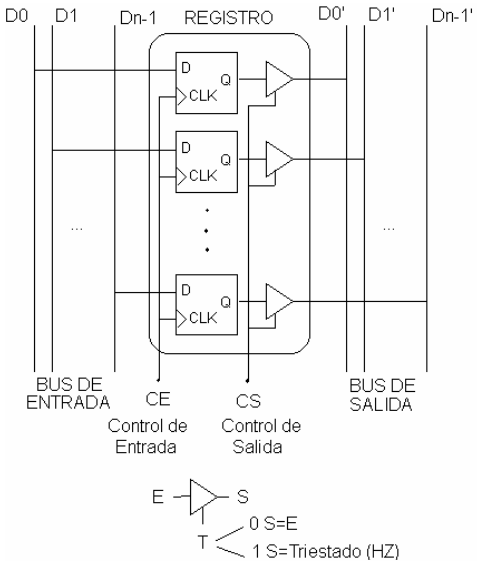


FIG.1.4 REGISTRO INTERNO DE LA CPU

Los registros pueden escribir, leer e intercambiar información entre ellos por medio de los ***Buses Internos de la CPU***. Con estructuras adecuadas se puede intercambiar información entre registros, cargar registros, descargarlos, etc.:

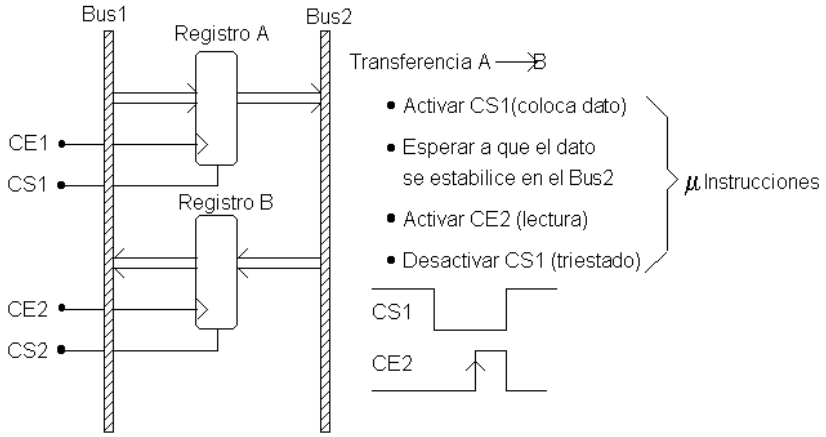


FIG 1.5 BUSES INTERNOS DE LA CPU

Las señales CS1 y CE2 son gestionadas por la unidad de control de la CPU instruida por el código de la instrucción de transferencia A→B. Los datos que están almacenados en los registros de la CPU tienen un **tiempo de acceso muy bajo**, muy inferior del correspondiente a los datos que se encuentran en memoria externa. Existen diferentes tipos de registros que permiten realizar diferentes funciones:

De Desplazamiento, empleando biestables D encadenados. Permiten hacer rotaciones y desplazamientos de bits (x2, %2, E/S serie, etc.)

Contadores, con biestables T encadenados. Permiten medir tiempos, temporizar, etc.

Registros Índices o de Direcciones,(fig 1.5) permiten el acceso estructurado a datos de memoria

Registros de datos, para almacenamiento de variables del programa

Registros especiales, contador de programa (Program Counter, PC), puntero de pila (Stack Pointer, SP)

1.4.2 UNIDAD DE CONTROL

Su misión principal consiste en decodificar (interpretar y ejecutar) las instrucciones que le llegan desde la memoria:

- El Decodificador de Instrucciones selecciona las posiciones que corresponden a esa instrucción en una memoria ROM interna de la CPU. En ella se almacenan las diferentes instrucciones elementales o µinstrucciones que componen esa instrucción.

- Estas µinstrucciones hacen que el Secuenciador active las señales correspondientes para ejecutar la instrucción.

- El Contador de Programa (PC) es un registro que contiene la dirección de memoria donde está la siguiente instrucción del programa a ejecutar. Una de las primeras acciones del secuenciador antes de ejecutar una instrucción es incrementar el PC para que apunte a la instrucción siguiente.

- Existen instrucciones que permiten modificar el PC o incrementarlo en más de una unidad, permitiendo la realización de saltos en la secuencia del programa.

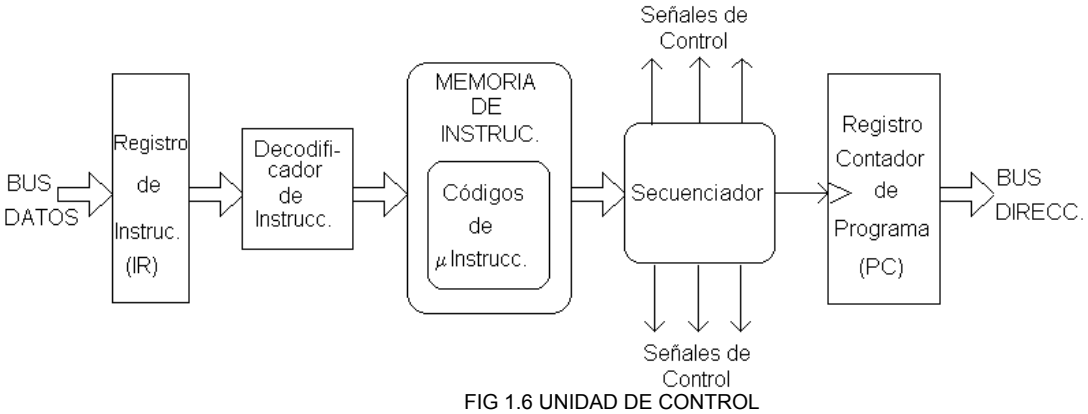


FIG 1.6 UNIDAD DE CONTROL

1.4.3 UNIDAD DE PROCESO

Su bloque principal es la ALU o Unidad Lógico-Aritmética, que permite realizar las operaciones aritméticas y lógicas indicadas por las instrucciones del programa.

- **EL SECUENCIADOR**, instruido por el código de la instrucción en curso, activa las líneas de selección de la ALU para realizar la operación.

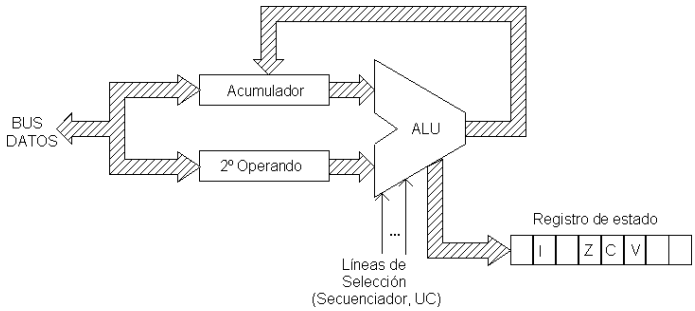


FIG. 1.7 UNIDAD DE PROCESO

Los operandos se suministran por medio de dos registros cargados desde el bus de datos:

- Registro Acumulador, contiene siempre el resultado de la última operación realizada en la ALU.

- Registro 2º Operando, proporciona el 2º operando para realizar la instrucción y viene normalmente suministrado por el código de operación de la instrucción a ejecutar según los diferentes modos de direccionamiento.

Esta forma de trabajar es típica de los micros de 8 bits y permite simplificar las instrucciones ya que cada instrucción sólo tiene que suministrar un operando, el otro se encuentra cargado previamente en el acumulador.

- El Registro de Estado esta formado por bits denominados banderas (flags) que se ponen a 1 ó 0 de acuerdo con el resultado obtenido. Algunos bits típicos son:

  - Z, bit zero; se pone a 1 si el resultado fue cero.
  - C, bit carry; se pone a 1 si hubo acarreo de orden superior
  - V, bit overflow; se pone a 1 si hubo desbordamiento
  - I, bit de interrupción; este bit es independiente del resultado. Escribiendo un uno en él por medio de la instrucción correspondiente, se puede habilitar la interrupción exterior.

1.5 INTERFAZ O MÓDULO DE E/S

Permite la comunicación del sistema µP con el exterior. Los dispositivos de E/S se denominan habitualmente periféricos, ejemplos son: controladores, teclado, pantalla, impresora, unidades de disco, etc. Cualquier periférico necesita un módulo adicional que permite realizar la conexión del mismo con los buses del µP.

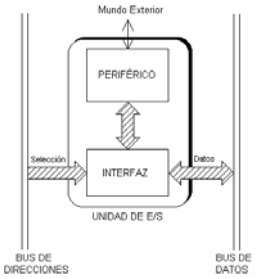


FIG. 1.8 COMUNICACIÓN DEL MICROPROCESADOR AL EXTERIOR A TRAVES DE UNA INTERFAZ

Existen varios métodos para manejar los dispositivos de E/S:

- Mediante instrucciones específicas de E/S, que se emplean en el programa para acceder al periférico.

- Con Acceso Directo a Memoria (DMA). La CPU pone en triestado los buses de

direcciones y de datos. Un dispositivo controlador de DMA toma el control de los buses y pasa los datos directamente entre el dispositivo E/S y la memoria.

- A través de Técnicas de Interrupción. El periférico activa las líneas de interrupción de la CPU, que detienen el programa en ejecución y trasladan el contador de programa a la dirección de inicio de otro programa (la dirección del contador de programa se almacena en la Pila –Stack-) creado especialmente para atender al periférico que solicita la interrupción. Suele denominarse a este programa rutina de servicio de interrupción. La Pila puede estar implementada como registros internos ó ubicada en algún lugar de la memoria de datos; la cantidad de localidades destinada para la pila, determina el número de interrupciones y anidamientos de subrutinas que puede soportar el µP. La dirección donde se encuentran los datos que se almacenan/recuperan de la pila, está determinada por un registro interno conocido como Puntero de Pila –Stack Pointer, (SP-), un método comúnmente empleado para el manejo de la Pila es el LIFO (Last In First Out, último en entrar primero en salir), mediante el cual, cada vez que se almacena una dato en la pila el valor del SP decrece, y cuando se extrae un dato el SP se incrementa.

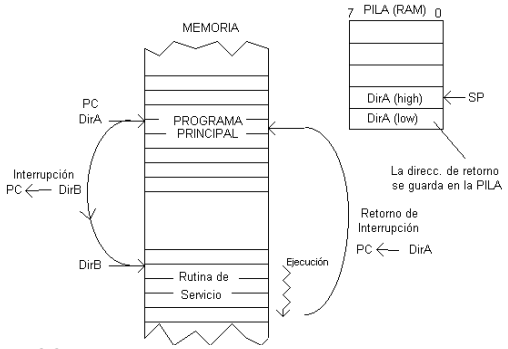


FIG. 1.9 USO DE LA PILA PARA ENTENDER UNA INTERRUPCION

- Mediante el tratamiento de E/S como posiciones de memoria. Permite el empleo de las mismas instrucciones para acceso a memoria y a E/S. Una zona del mapa de memoria es reservada para los dispositivos de E/S. Estas posiciones se llaman Puertos de E/S o registros mapeados en memoria. Escribir o leer en uno de estos puertos equivale a hacerlo en el periférico. El mapa de memoria es un esquema o representación de todas las posiciones de memoria que puede direccionar el µprocesador, mostrando las clases de datos y su uso en el sistema.

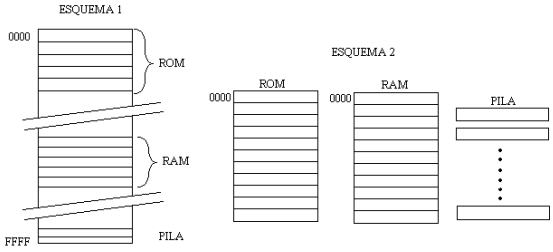


FIG. 1.10 REGISTROS MAPEADOS EN MEMORIA

1.6 DIAGRAMA GENERAL DE UN SISTEMA BASADO EN UN MP DE 8 BITS

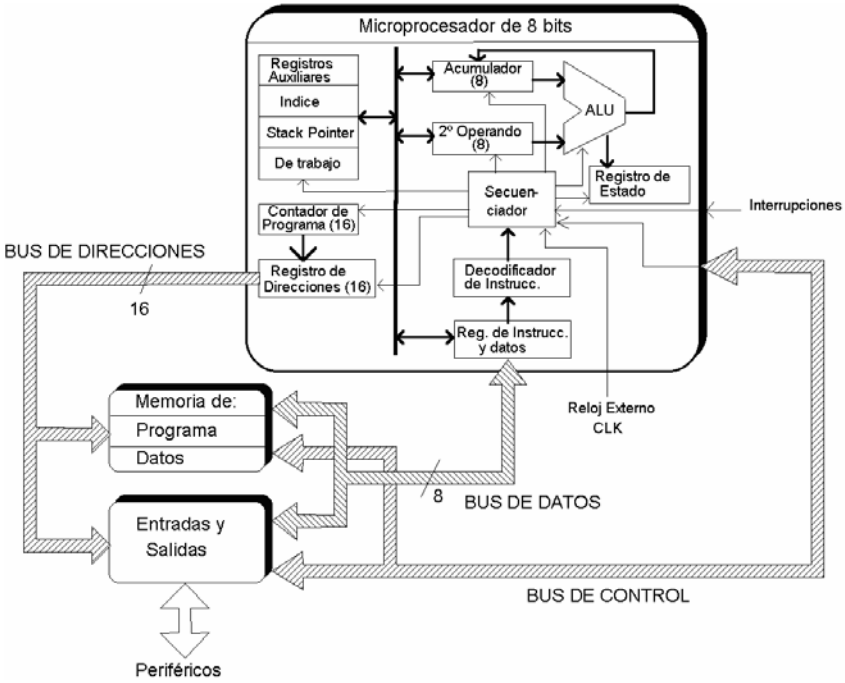


FIG. 1.11 MICROPROCESADOR DE 8 BITS

EJECUCIÓN DE UNA INSTRUCCIÓN.- La ejecución de una instrucción se lleva a cabo en dos fases:

- **ADQUISICIÓN DE LA INSTRUCCIÓN (CICLO FETCH).-** Es común a todas las instrucciones. Se inicia en el contador de programa (PC), que contiene la dirección de memoria donde se encuentra el código binario de la instrucción. Esta dirección se

coloca en el registro de direcciones de la CPU y de ahí a la memoria a través del bus de direcciones. Una vez decodificada la dirección de memoria, el contenido de la localidad se traslada por el bus de datos hacia el registro de Instrucciones de la Unidad de Control. En este momento finaliza la fase de adquisición. Como se muestra en la fig. 1.12.

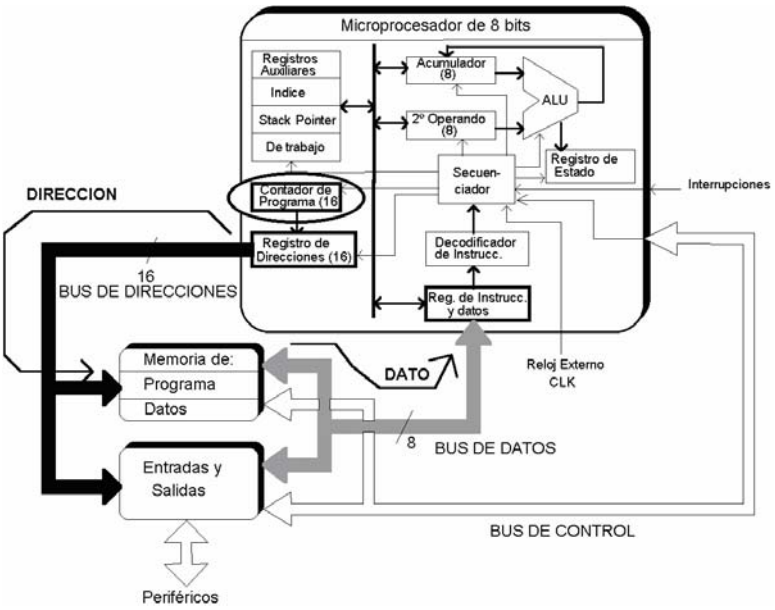


FIG. 1.12 FASE DE ADQUISICION.

Las instrucciones que constituyen el programa se almacenan en memoria en paquetes de 8 bits (Bytes). Normalmente de la forma siguiente:

- Byte 1: es el código de operación (OPC ó OPCODE) que indica la operación de la que se trata y su función. Por ejemplo operación lógica AND.

- Byte 2 y siguientes: normalmente nos dan la información necesaria para acceder al dato sobre el que va a trabajar la instrucción. Pueden ser el propio dato, la dirección de memoria donde se encuentra el dato, etc. las diferentes posibilidades para acceder a ese dato se denominan modos de direccionamiento del µP.

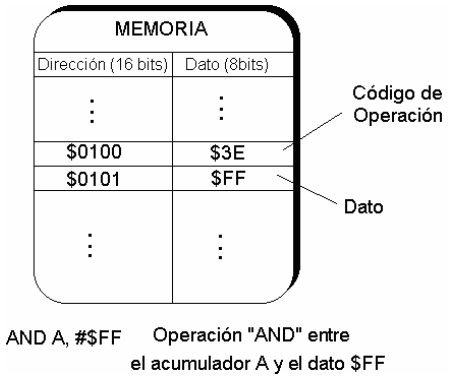


FIG. 1.13 INSTRUCCIÓN ALMACENADA EN MEMORIA

Los OPCs suelen ser del mismo tamaño que el bus de datos del micro. En los micros de 8 bits (6502, Z80, 8088) los OPCs son de 8 bits. En los micros de 16 bits (68000, 8086, Z8000) los OPCs son de 16 bits.

En general, el conjunto de instrucciones del µProcesador se puede dividir en los siguientes bloques funcionales:

- Aritméticas: suma, resta, producto, … etc.
- Lógicas: AND, OR, NOT, EXOR, … etc.
- De transferencia de datos: permiten transferir datos entre registros; entre memoria y los registros de la CPU; entre dos posiciones de memoria, etc.
- De Entrada/Salida: permiten la lectura y escritura en los bloques periféricos de E/S.
- De control del procesador: parada (STOP), no operación (NOP), interrupciones…etc
- De ruptura de secuencia del programa: permiten realizar saltos condicionales o incondicionales dentro del programa:

ESTRUCTURA DEL MICROCONTROLADOR.

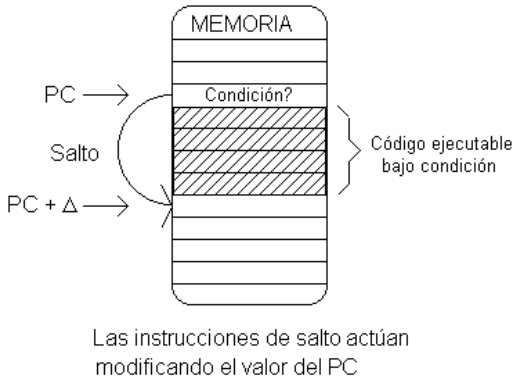


FIG. 1.14 INSTRUCCIÓN DE SALTO CONDICIONAL

1.7 MICROCONTROLADORES

Un microcontrolador es un sistema completo (microprocesador + E/S + memoria + otros periféricos), aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado programable y se destina a gobernar una sola tarea con el programa que reside en su memoria. Sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar.

Si sólo se fabricara un modelo de microcontrolador, éste debería tener muy potenciados todos sus recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. Esta potenciación supondría en muchos casos un gasto innecesario. En la práctica, cada fabricante de microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más potentes. Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S, la cantidad y prestaciones de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar.

1.7.1 APLICACIONES DE LOS MICROCONTROLADORES

Cada vez existen más productos que incorporan un microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir su tamaño y costo, mejorar su fiabilidad y disminuir el consumo energético. Los microcontroladores están siendo empleados en multitud de sistemas presentes en nuestra vida diaria, como pueden ser: juguetes, hornos de microondas,

refrigeradores, televisores, computadoras, impresoras, módems, computadoras de viaje para automóviles, etc. Una aplicación típica podría emplear varios microcontroladores para controlar pequeñas partes del sistema. Estos pequeños controladores podrían comunicarse entre ellos y con un procesador central, probablemente más potente, compartir la información y coordinar sus acciones, como de hecho, ocurre ya habitualmente en cualquier computador.

1.7.2 EL MERCADO DE LOS MICROCONTROLADORES

Aunque en el mercado de la informática la mayor atención la acaparan los desarrollos de los microprocesadores, lo cierto es que se venden cientos de microcontroladores por cada procesador.

Existe una gran diversidad de microcontroladores. Quizá la clasificación más importante sea entre microcontroladores de 4, 8, 16 ó 32 bits. Aunque las prestaciones de los microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los microcontroladores de 8 bits dominan el mercado. La razón de esta tendencia es que los microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace absurdo emplear micros más potentes y consecuentemente más caros.

Uno de los sectores que más se utiliza en el mercado del microcontrolador es el automovilístico. De hecho, algunas de las familias de microcontroladores actuales se desarrollaron pensando en este sector, siendo modificadas posteriormente para adaptarse a sistemas más genéricos. El mercado del automóvil es además uno de los más exigentes: los componentes electrónicos deben operar bajo condiciones extremas de vibraciones, choques, ruido, etc. y seguir siendo fiables. El fallo de cualquier componente en un automóvil puede ser el origen de un accidente.

1.7.3 RECURSOS COMUNES A TODOS LOS MICROCONTROLADORES

Al estar todos los microcontroladores integrados en un chip, su estructura fundamental y sus características básicas son muy parecidas. Todos deben disponer de los bloques esenciales: Procesador, memoria de datos y de instrucciones, líneas de E/S, oscilador de reloj y módulos

controladores de periféricos. Sin embargo, cada fabricante intenta enfatizar los recursos más idóneos para las aplicaciones a las que se destinan preferentemente.

1.7.4 ARQUITECTURA BÁSICA

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de von Neumann, en el momento presente se impone la arquitectura Harvard. La arquitectura de von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control).La arquitectura Harvard dispone de dos memorias independientes, una que contiene sólo instrucciones y otra sólo los datos. Ambas disponen de sus respectivos sistemas de buses de acceso, siendo posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias.

1.7.5 EL PROCESADOR O CPU

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales.

- CISC: Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores de Juego de Instrucciones Complejo). Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución.Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros.

- RISC: Tanto la industria de los computadores comerciales como la de los microcontroladores, están decantándose hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y simple, generalmente se ejecutan en un ciclo. La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del

procesador.

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

1.7.6 MEMORIA

En los microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos. Hay dos peculiaridades que diferencian a los microcontroladores de las PCs: No existen sistemas de almacenamiento masivo como disco duro o disquetes, y como el microcontrolador sólo se destina a una tarea en la memoria ROM, sólo hay que almacenar un único programa de trabajo.

- La RAM.- En estos dispositivos es de poca capacidad, pues sólo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la RAM pues se ejecuta directamente desde la ROM. Los usuarios de computadoras personales están habituados a manejar Megabytes de memoria, pero los diseñadores con microcontroladores trabajan con capacidades de RAM comprendidas entre 20 bytes y 1 Kbytes.

Según el tipo de memoria ROM que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente. Se describen cinco versiones de memoria no volátil que se pueden encontrar en los microcontroladores del mercado.

- ROM con máscara.- Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip. El elevado coste del diseño de la máscara sólo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se

precisan cantidades superiores a varios miles de unidades.

- ROM OTP (de una sola programacion).- El microcontrolador contiene una memoria no volátil de sólo lectura "programable una sola vez" por el usuario, es él quien puede escribir el programa en el chip mediante un grabador controlado por un programa desde una PC. La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas. Tanto en este tipo de memoria como en la EPROM, se suele usar la encriptación mediante fusibles para proteger el código contenido.

- EPROM (memoria leída programable y borrable).- Los microcontroladores que disponen de memoria EPROM pueden borrarse y grabarse muchas veces. La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde una PC. Si posteriormente se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico.

- EEPROM (memoria leida programable y borable eléctricamente).- Se trata de memorias de sólo lectura, programables y borrables eléctricamente. Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de una PC. Es muy cómoda y rápida la operación de grabado y de borrado. No disponen de ventana de cristal en la superficie. Los microcontroladores dotados de memoria EEPROM, una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo. El número de veces que puede grabarse y borrarse una memoria EEPROM es finito, por lo que no es recomendable una reprogramación continua. Son muy idóneos para la enseñanza y la Ingeniería de diseño. Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria EEPROM en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que adecúan el dispositivo a las condiciones del entorno. Este tipo de memoria es relativamente lenta.

- FLASH.- Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y

borrar. Funciona como una ROM y una RAM, pero consume menos y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM. La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado

Las memorias EEPROM y FLASH son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados "en circuito", es decir, sin tener que sacar el circuito integrado de la tarjeta. Así, un dispositivo con este tipo de memoria incorporado al control del motor de un automóvil permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes y otros factores tales como las adaptaciones, la instalación de nuevas piezas, etc. La reprogramación del microcontrolador puede convertirse en una labor rutinaria dentro de la puesta a punto.

1.7.7 PUERTOS DE ENTRADA Y SALIDA

La principal utilidad de las terminales que posee la cápsula que contiene un microcontrolador es soportar las líneas de E/S que comunican al µprocesador con los periféricos exteriores. Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control.

1.7.8 RELOJ PRINCIPAL

Todos los microcontroladores necesitan de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema. Generalmente, el circuito de reloj está incorporado en el microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un oscilador de cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C. Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía; además la frecuencia de reloj tiene un límite máximo permitido.

1.8 RECURSOS ESPECIALES

Cada fabricante ofrece numerosas versiones de una arquitectura básica de microcontrolador. En algunas amplía la capacidad de las memorias, en otras incorpora nuevos recursos, en otras reduce las prestaciones al mínimo para aplicaciones muy simples, etc. La labor del diseñador es encontrar el modelo mínimo que satisfaga todos los requerimientos de su aplicación. De esta forma, reducirá el costo, el hardware y el software.

Los principales recursos específicos que incorporan los microcontroladores son:

- Temporizadores (Timers).
- Perro guardián (Watchdog).
- Protección ante fallo de alimentación (Brownout).
- Estado de reposo o de bajo consumo (Sleep o Idle).
- Conversor A/D.
- Conversor D/A.
- Comparador analógico.
- Modulador de anchura de pulsos (PWM).
- Puertos de E/S digitales.
- Puertos de comunicación.

1.8.1TEMPORIZADORES

Se emplean para controlar periodos (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores). Para la medida de tiempos se carga un registro con el valor adecuado, y a continuación dicho valor se va incrementando o decrementando a la frecuencia de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a cero, momento en el que se produce un aviso. Cuando se desean contar acontecimientos que se suceden por cambios de nivel o flancos en alguna de las terminales del microcontrolador, el mencionado registro se va incrementando o decrementando a la frecuencia de dichos pulsos.

1.8.2 PERRO GUARDIÁN

Cuando una computadora personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continua las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por cero, provoca un reset automáticamente en el sistema. Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, enviará una señal "ladrará" para provocar el reset.

1.8.3 PROTECCIÓN ANTE FALLO DE ALIMENTACIÓN

Se trata de un circuito que reinicia al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo ("brownout"). Mientras el voltaje de alimentación sea inferior al de brownout el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor de voltaje.

1.8.4 ESTADO DE REPOSO Ó DE BAJO CONSUMO

Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los microcontroladores disponen de una instrucción especial (Sleep en algunos modelos), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se "congelan" sus circuitos asociados, quedando el microcontrolador en un profundo "sueño". Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se "despierta" y reanuda su trabajo.

1.8.5 CONVERSOR A/D (CAD) (CONV. ANALOGICO – DIGITAL)

Los microcontroladores que incorporan un CAD pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD varias señales analógicas desde las terminales del circuito integrado.

1.8.6 CONVERSOR D/A (CDA) (CONV. DIGITAL – ANALOGICO)

Transforma los datos digitales obtenidos del procesamiento en su correspondiente señal analógica, y que saca al exterior por una de las terminales de la cápsula. Existen muchos actuadores que trabajan con señales analógicas.

1.8.7 COMPARADOR ANALÓGICO

Algunos modelos de microcontroladores disponen internamente de un Amplificador Operacional, que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por una de las terminales de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra. También hay modelos de microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia que se pueden aplicar en los comparadores.

1.8.8 MODULADOR DE ANCHURA DE PULSOS

Son circuitos que proporcionan en su salida pulsos de anchura variable, que se muestran al exterior a través de las terminales del encapsulado.

1.8.9 PUERTOS DE E/S DIGITALES

Todos los microcontroladores destinan algunas de sus terminales a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos. Las líneas digitales de los Puertos pueden configurarse como Entrada o como Salida cargando un 1

ó un 0 en el bit correspondiente de un registro destinado a su configuración.

1.8.10 PUERTOS DE COMUNICACIÓN

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos; algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

- **UART**, (Universal Synchronous Receiver Transmitter, Transmisor Receptor Asíncrono Universal), adaptador de comunicación serie asíncrona.

- **USART** (Universal Synchronous Asynchronous Receiver Transmitter, Transmisor Receptor Síncrono y Asíncrono Universal), adaptador de comunicación serie síncrona y asíncrona

- **PUERTO** paralelo esclavo, para poder conectarse con los buses de otros microprocesadores.

- **USB** (Universal Serial Bus, Bus serial Universal), es un moderno bus serie para las PC.

- **BUS I2C** (Inter-IC bus, bus de intercambio de circuitos integrados), es una interfaz serie de dos hilos desarrollada por Phillips.

- **CAN** (Controller Area Network, Área de Red Controlada), permite la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE.UU.

1.9 HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES

Uno de los factores que más importancia tiene al momento de seleccionar un microcontrolador entre todos los demás, es el soporte tanto de software como de hardware del que se disponga. Un buen conjunto de herramientas de desarrollo puede ser decisivo en la elección, ya que pueden suponer una ayuda inestimable en el desarrollo del proyecto. Las principales

herramientas de soporte al desarrollo de software para sistemas basados en microcontroladores son:

ENSAMBLADOR. La programación en lenguaje ensamblador puede resultar un tanto ardua para el principiante, pero permite desarrollar programas muy eficientes, ya que otorga al programador el dominio absoluto del sistema. Está aplicación traduce las instrucciones en lenguaje ensamblador -en forma de mnemónicos y operandos- a código maquina. Los fabricantes suelen proporcionar el programa ensamblador de forma gratuita y en cualquier caso siempre se puede encontrar una versión gratuita para los microcontroladores más populares.

COMPILADOR. La programación en un lenguaje de alto nivel (como el C ó el Basic) permite disminuir el tiempo de desarrollo de un producto. No obstante, si no se programa con cuidado, el código resultante puede ser mucho más ineficiente que el programado en ensamblador. Las versiones más potentes suelen ser muy caras, aunque para los microcontroladores más populares pueden encontrarse versiones demo limitadas e incluso compiladores gratuitos.

DEPURADOR. Debido a que los microcontroladores van a controlar dispositivos físicos, los desarrolladores necesitan herramientas que les permitan comprobar el buen funcionamiento del microcontrolador cuando es conectado al resto de circuitos.

SIMULADOR. Son capaces de ejecutar en una PC los programas realizados para el microcontrolador. Los simuladores permiten tener un control absoluto sobre la ejecución de un programa, siendo ideales para la depuración de los mismos. Su gran inconveniente radica en la dificultad para simular la entrada y salida de datos del microcontrolador. Tampoco cuentan con los posibles ruidos en las entradas, pero al menos permiten el paso físico de la implementación de un modo más seguro y menos costoso, puesto que ahorraremos en grabaciones de chips para la prueba in-situ.

SISTEMAS DE DESARROLLO. Se trata de pequeños sistemas con un microcontrolador ya montado, y que suelen conectarse a una PC desde el que se cargan los programas que se ejecutan en el microcontrolador. Las placas suelen incluir visualizadores LCD, teclados, LEDs, fácil acceso a los pines de E/S, etc. El sistema operativo de la placa recibe el nombre de programa monitor. El programa monitor de algunos sistemas de desarrollo, aparte de permitir cargar programas y datos en la memoria del microcontrolador, puede permitir en cualquier momento realizar ejecuciones paso a paso, monitoreo del estado del microcontrolador o

modificar los valores almacenados en los registros o en la memoria.

EMULADORES EN CIRCUITO. Se trata de un instrumento que se coloca entre la PC anfitrión y el zócalo de la tarjeta de circuito impreso donde se alojará el microcontrolador definitivo. El programa es ejecutado desde la PC, pero para la tarjeta de aplicación es como si lo hiciese el mismo microcontrolador que luego irá en el zócalo. Presenta en pantalla toda la información tal como sucederá cuando se coloque la cápsula.

1.10 MICROCONTROLADORES PIC

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

La arquitectura del PIC, que se comercializó en 1975, era sustancialmente la misma que la de los actuales modelos PIC16C5X. En aquel momento se fabricaba con tecnología NMOS y el producto sólo se ofrecía con memoria ROM y con un pequeño pero robusto microcódigo.

La década de los 80 no fue buena para GI, que tuvo que reestructurar sus negocios, concentrando sus actividades en los semiconductores de potencia. La GI Microelectronics Division se convirtió en una empresa subsidiaria, llamada GI Microelectronics Inc. Finalmente, en 1985, la empresa fue vendida a un grupo de inversores de capital de riesgo, los cuales, tras analizar la situación, rebautizaron a la empresa con el nombre de Arizona Microchip Technology, y orientaron su negocio a los PIC, las memorias EPROM paralelo y las EEPROM serie. Se comenzó rediseñando los PIC, que pasaron a fabricarse con tecnología CMOS, surgiendo la familia de gama baja PIC16CSX, considerada como la "clásica".

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

Microchip cuenta con su factoría principal en Chandler, Arizona, en donde se fabrican y prueban los chips con los más avanzados recursos técnicos. En 1993 construyó otra factoría de similares características en Tempe, Arizona. También cuenta con centros de ensamblaje y pruebas en Taiwan y Tailandia. Para tener una idea de su alta producción, hay que tener en cuenta que ha superado el millón de unidades por semana en productos CMOS de la familia PIC16CSX.

1.10.1 CARACTERÍSTICAS

La arquitectura del procesador sigue el modelo Harvard. En esta arquitectura, el CPU se conecta de forma independiente con buses distintos a la memoria de instrucciones y a la de datos.

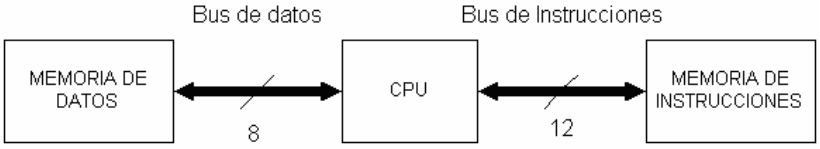


FIG. 1.15 ARQUITECTURA HARVARD

La arquitectura Harvard permite al CPU acceder simultáneamente a las dos memorias. Además, propicia numerosas ventajas al funcionamiento del sistema como se irán describiendo.

Se aplica la técnica de segmentación (“pipe-line”) en la ejecución de las instrucciones. La segmentación permite al procesador realizar al mismo tiempo la ejecución de una 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 cuatro pulsos de reloj).

El formato de todas las instrucciones tiene la misma longitud. Todas las instrucciones de los microcontroladores de la gama baja tienen una longitud de 12 bits. Las de la gama media tienen 14 bits y 16 bits las de la gama alta.

Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, 35 los de la gama media y casi 60 los de la alta.

Todas las instrucciones son ortogonales. Cualquier instrucción puede manejar cualquier elemento de la arquitectura como 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 implementados físicamente como registros. Existe una gran diversidad de modelos de microcontroladores con prestaciones y recursos diferentes.

Herramientas de soporte. 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, simuladores por software, emuladores en tiempo real, ensambladores, compiladores C, intérpretes y compiladores BASIC, etc.

1.10.2 LAS GAMAS DE PIC.

Para resolver aplicaciones sencillas se precisan pocos recursos; en cambio, las aplicaciones grandes los requieren numerosos y potentes. Siguiendo esta filosofía, Microchip construye diversos modelos de microcontroladores orientados a cubrir las necesidades de cada proyecto. Así, hay disponibles microcontroladores sencillos y baratos para atender las aplicaciones simples, y otros complejos y más costosos para las de mucha envergadura.

Con las gamas de PIC se dispone de gran diversidad de modelos y encapsulados, pudiendo seleccionar el que mejor se acople a las necesidades de acuerdo con el tipo y capacidad de las memorias, el número de líneas de E/S y las funciones auxiliares precisas. Sin embargo, todas las versiones están construidas alrededor de una arquitectura común, un repertorio mínimo de instrucciones y un conjunto de opciones muy apreciadas, como el bajo consumo y el amplio margen del voltaje de alimentación. La tabla (1.1) que se presenta a continuación resume las características más importantes de algunos modelos que pueden ser adquiridos en el mercado local:

MODELO	MEMORIA DE PROGRAMA	EEPROM (Bytes)	RAM (Bytes)	ADC	TIMER	PINES I/O	GAMA
PIC12F675	1024x14 Flash	128	64	4/10 bit	3	6	Baja
PIC16C765	8192x14 PROM	0	256	8/8 bit	4	30	Media
PIC16F77	8192X14 Flash	0	368	8/8 bit	4	33	Media
PIC16F84	1024x14 Flash	64	68	0	2	13	Media
PIC16F874	4096x14 Flash	128	192	8/10 bit	4	33	Media
PIC16F876	8192x14 Flash	256	368	5/10 bit	4	22	Media
PIC17C42	2048x16 PROM	0	232	0	5	33	Alta
PIC18F258	16384x16 Flash	256	1536	5/10 bit	4	22	Alta

TABLA 1.1 ALGUNOS MODELOS DE PIC´s

1.10.3 ARQUITECTURA

La repercusión más importante del empleo de la arquitectura Harvard en los microcontroladores PIC se manifiesta en la organización de la memoria del sistema. La memoria de programa es independiente a la memoria de datos, teniendo tamaños y longitudes de palabra diferentes. Este tamaño permite codificar en una palabra el código de operación de la instrucción junto al operando o su dirección.

El tamaño de los buses que direccionan la memoria de datos y la de programa son diferentes. Lo mismo pasa con el bus que transfiere las instrucciones y el que lo hace con los datos. La total independencia entre las dos memorias permite realizar accesos simultáneos. Puesto que los datos y operandos que manejan las instrucciones son de 8 bits, la longitud de palabra de la memoria de datos tiene ese tamaño. La capacidad de la SRAM varía según el modelo.

La memoria de programa siempre está direccionada desde el Contador de Programa (PC), mientras que la memoria de datos puede direccionarse directamente desde parte del

código de operación de la instrucción o indirectamente.

La memoria de datos se organiza en "bancos", pudiendo existir varios en los modelos de mayor capacidad. La memoria de datos funciona de forma similar al "banco de registros" de un procesador, por lo cual sus posiciones implementan registros de propósito específico y propósito general. Las primeras posiciones de los bancos contienen registros específicos. Es necesario seleccionar con cual banco vamos a trabajar mediante los bits 6 y 5 del registro FSR.

La pila consta de registros que funcionan en forma de memoria LIFO (último en entrar, primero en salir). El contenido del PC se carga en el primer nivel de la Pila con la instrucción CALL y sucede lo contrario con la instrucción RETLW. La profundidad de la pila varía en cada modelo; el PIC16F876 tiene ocho registros de 13 bit.

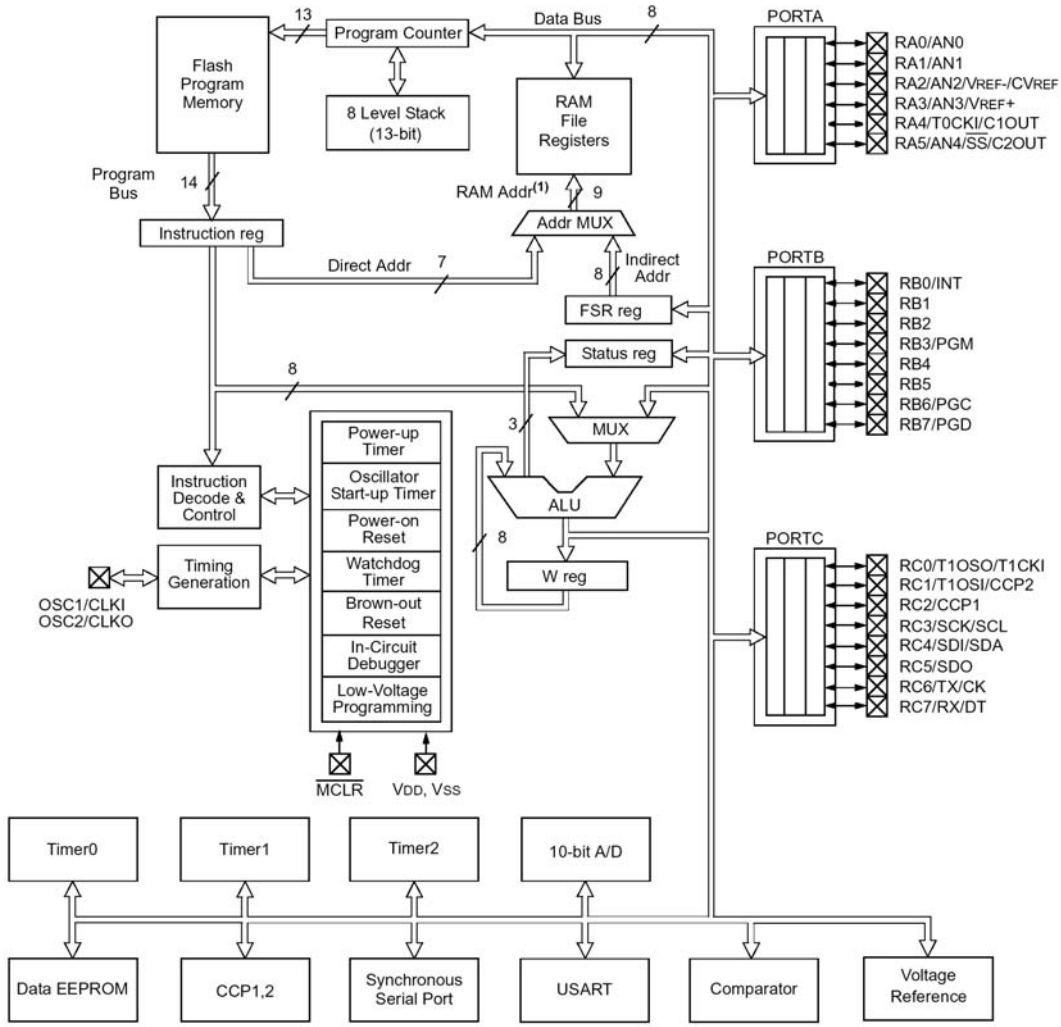


FIG. 1.16 Diagrama a bloques del PIC16F876

CAPÍTULO 2

2. PRACTICAS PROPUESTAS.

Con base en el Plan de Calidad de los laboratorios de la división de ciencias físico-matemáticas y de las ingenierías de la Facultad de Estudios Superiores Aragón, que se anexa en esta tesis en el apartado A-1. Se propone en este capitulo una serie de practicas para la materia de microprocesadores de acuerdo al plan de estudios aprobado en junio 28 de 1991[1]. Y analizando el temario de la materia que se muestra en el anexo A-2 y con el visto bueno del jefe de carrera se diseñaron las siguientes prácticas:

PRÁCTICA 0: EJERCICIOS DE PROGRAMACION EN ENSAMBLADOR. Tomando en cuenta los antecedentes del capitulo uno, se abarcan los temas I, II y IV del temario de la materia y consideramos que el alumno deberá enfocarse a un solo microcontrolador de la gama media, sugiriendo el 16f628 de microchip teniendo como punto de referencia la enciclopedia wikipedia de la Web que publica los PIC mas comúnmente utilizados su juego de instrucciones y su entorno de programación[2].

PRÁCTICA 1: IMPLEMENTAR UN GRABADOR DE UN PIC´s. Para poder grabar el PIC es necesario implementar un grabador de bajo costo que sea accesible a los alumnos por lo que se propone dicha práctica.

PRÁCTICA 2: PROGRAMACION DE UN PIC. Debido a que el tema III del temario habla sobre el tipo de memorias, consideramos muy oportuno que el alumno aprenda a grabar la memoria interna del chip.

PRÁCTICA 3: PROGRAMACION DE UN DISPLAY. En el tema V, se inicia la

[1] http://www.dgae.unam.mx/planes/aragon/Ingmec-ele_aragon.pdf

programación del microprocesador y con las bases adquiridas en las prácticas anteriores, el alumno esta preparado para la realización de programas que involucren entrada y salida a través de los puertos del PIC, por lo que proponemos un contador que se visualiza en el display.

PRÁCTICA 4: PROGRAMACION DE UN LCD. Para el tema VI se requiere realizar una interfase con un dispositivo interno, por lo que proponemos programar un LCD, además de considerar el tema anterior relacionado con las funciones aritméticas.

PRÁCTICA 5: PROGRAMACION DE UN TECLADO MATRICIAL. Debido a que en los sistemas digitales es necesario un dispositivo que interactué con el microcontrolador se realizo esta práctica para proporcionarle más herramientas al alumno.

PRÁCTICA 6: PROGRAMACION DE MOTORES A PASOS. Para esta práctica se hace un enfoque del tema VIII sustituyendo el puerto paralelo de la PC, al puerto paralelo del microcontrolador, programando un motor a pasos.

[2] http://es.wikipedia.org/wiki/Microcontrolador_PIC

2.1 PRÁCTICA 0: EJERCICIOS DE PROGRAMACION EN ENSAMBLADOR.

2.1.1 Objetivo.

Describir las 35 instrucciones correspondientes a la gama media de los PICs; esta gama es la más abundante, y que la mayoría de las aplicaciones pueden resolverse con uno de sus diferentes modelos.

2.1.2 Fundamentos teóricos básicos

La sintaxis de cada instrucción está en lenguaje ensamblador. Una línea típica en lenguaje ensamblador presenta la siguiente forma.

[Etiqueta] Mnemónico Operando(s) ;Comentario.

La **etiqueta** es opcional y sirve para indicarle al ensamblador la ubicación destino de una instrucción de salto o brinco a subrutina. Los **Mnemónicos** son las instrucciones en lenguaje ensamblador que puede reconocer el microprocesador. Dependiendo de la instrucción, puede haber uno o dos **operandos**, incluso pueden no existir; su contenido consiste en un número, una variable o una dirección; normalmente el resultado de una operación lógica, aritmética o de carga es almacenado en el primer operando.

NOMENCLATURA DEL CONJUNTO DE INSTRUCCIONES

Campo	Descripción
f	Dirección de registros en los bancos de memoria (0x00 a 0x7F)
W	Registro de trabajo (acumulador)
b	Posición de un bit en un registro (0 a 7)
k	Dato inmediato de 8 bit; También puede constar de hasta 11bit en las instrucciones de salto que cargan al PC
x	Valor indeterminado (puede ser un 0 o un 1). Para mantener la compatibilidad con las herramientas software de Microchip conviene hacer x = 0
d	Bit que selecciona el destino. Si d=0 el destino es W, y si d=1 el destino es f
dest	Destino (W o f)
label	Nombre de etiqueta
TOS	Cima de la pila
PC	Contador de programa
PCLATH	Parte alta del PC
GIE	Bit de habilitación global de interrupciones
WDT	Temporizador del watchdog
TO	Bit "time-out" del registro de estado
PD	Bit "power-down" del registro de estado
[ ]	Opcional
( )	Contenido

←	Asignado a
< >	Campo de bits de un registro
∈	Pertenece al conjunto de
Itálicas	Términos definidos por el usuario

TABLA P.0.1

2.1.2 RESUMEN DEL CONJUNTO DE INSTRUCCIONES

Mnemónico	**Operandos**	**Descripción**
ADDWF	f, d	Suma a W y f
ANDWF	f, d	AND lógico entre W y f
CLRF	f	Poner a cero f
CLRW	-	Poner a cero W
COMF	f, d	Complemento de f
DECF	f, d	Decrementar f
DECFSZ	f, d	Decrementar f, saltar si es cero
INCF	f, d	Incrementar f
INCFSZ	f, d	Incrementar f, saltar si es cero
IORWF	f, d	OR lógico entre W y f
MOVF	f, d	Mueve f
MOVWF	f	Mueve W hacia f
NOP	-	No operación
RLF	f, d	Rotar f a la izquierda con acarreo
RRF	f, d	Rorar f a la derecha con acarreo
SUBWF	f, d	Resta W de f
SWAPF	f, d	Intercambia nibbles de f
XORWF	f, d	XOR entre W y f
BCF	f, b	Poner a cero un bit de f
BSF	f, b	Poner a uno un bit de f
BTFSC	f, b	Probar un bit de f, brincar si es cero

BTFSS	f, b	Probar un bit de f, brincar si es uno
ADDLW	k	Suma una literal a W
ANDLW	k	AND lógico entre W y una literal
CALL	k	Llamada a subrutina
CLRWDT	-	Limpiar el registro del "watchdog"
GOTO	k	Salta a una dirección
IORLW	k	OR Lógico entre W y una literal
MOVLW	k	Mueve una literal hacia W
RETFIE	-	Regreso de una interrupción
RETLW	k	Regreso con una literal en W
RETURN	-	Regreso de subrutina
SLEEP	-	Modo de reposo
SUBLW	k	Resta a W una literal
XORLW	k	XOR entre W y una literal

TABLA P0.2 CONJUNTOS DE INSTRUCCIONES

ADDLW Suma un literal

Sintaxis: [label] ADDLW k
Operandos: $0 \le k \le 255$
Operación: (W) + (k) → (W)
Banderas afectadas: C, DC, Z
Código OP: 11 111x kkkk kkkk

Descripción: Suma el contenido del registro W y k, guardando el resultado en W.

Ejemplo: ADDLW 0xC2

Antes: W = 0x17
Después: W = 0xD9

ADDWF W + F

Sintaxis: [label] ADDWF f,d
Operandos: $d \in [0,1]$; $0 \le f \le 127$
Operación: (W) + (f) → (dest)
Banderas afectadas: C, DC, Z
Código OP: 00 0111 dfff ffff

Descripción: Suma el contenido del registro W y el registro f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: ADDWF REG,0

Antes: W = 0x17
Después: W = 0xD9

ANDLW W AND literal

Sintaxis: [label] ANDLW k
Operandos: $0 \le k \le 255$
Operación: (W) AND (k)⇒ (W)
Banderas afectadas: Z
Código OP: 11 1001 kkkk kkkk

Descripción: Realiza la operación lógica AND entre el contenido del registro W y k, guardando el resultado en W.

Ejemplo: ADDLW 0xC2

Antes: W = 0x17
Después: W = 0xD9

ANDWF W AND F

Sintaxis: [label] ANDWF f,d
Operandos: $d \in [0,1]$; $0 \le f \le 127$
Operación: (W) AND (f) → (dest)
Banderas afectadas: Z
Código OP: 00 0101 dfff ffff

Descripción: Realiza la operación lógica AND entre los registros W y f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: ANDWF REG,0

Antes: W = 0x17
Después: W = 0x17

BCF Borra un bit

Sintaxis: [label] BCF f,b
Operandos: $0 \le f \le 127$, $0 \le b \le 7$
Operación: 0 → (f<b>)
Banderas afectadas: Ninguna
Código OP: 01 00bb bfff ffff

Descripción: Borra el bit b del registro f

Ejemplo: BCF REG, 7

Antes: REG = 0xC7
Después:

BSF Activa un bit

Sintaxis: [label] BSF f,b
Operandos: $0 \le f \le 127$; $0 \le b \le 7$
Operación: 1 → (f<b>)
Banderas afectadas: Ninguna
Código OP: 01 01bb bfff ffff

Descripción: Activa el bit b del registro f

Ejemplo: BSF REG, 7

Antes: REG = 0x0A
Después:

BTFSC Test de bit y salto

Sintaxis: [label] BTFSC f,d
Operandos: $d \in [0,1]$; $0 \le f \le 127$
Operación: Salta si (f<b>) = 0
Banderas afectadas: Ninguna
Código OP: 01 10bb bfff ffff

Descripción: Si el bit b del registro f es 0, se salta una instrucción y se continúa con la ejecución. En caso de salto, ocupará dos ciclos de reloj.

Ejemplo:
```
    BTFSC REG, 6
    GOTO NO_ES_0
SI_ES_0
    …
NO_ES_0   instrucción
```

BTFSS Test de bit y salto

Sintaxis: [label] BTFSS f,d
Operandos: $d \in [0,1]$; $0 \le f \le 127$
Operación: Salto si (f<b>) = 1
Banderas afectadas: Ninguna
Código OP: 01 11bb bfff ffff

Descripción: Si el bit b del registro f es 1, se salta una instrucción y se continúa con la ejecución. En caso de salto, ocupará dos ciclos de reloj.

Ejemplo:
```
        GOTO NO_ES_1
SI_ES_1   instrucción
NO_ES_1   instrucción
```

CALL Salto a subrutina

Sintaxis: [label] CALL k
Operandos: $0 \le k \le 2047$
Operación: PC → Pila; k → PC
Banderas afectadas: Ninguna
Código OP: 10 0kkk kkkk kkkk

Descripción: Salto a una subrutina. La parte baja de k se carga en PCL, y la alta en PCLATCH. Ocupa 2 ciclos de reloj.

Ejemplo:
```
    Destino …
```

Antes: PC = ORIGEN
Después:

CLRF Borra un registro

Sintaxis: [label] CLRF f
Operandos: 0 ≤ f ≤ 127
Operación: 0x00 → (f), 1 → Z
Banderas afectadas: Z
Código OP: 00 0001 1fff ffff

Descripción: El registro f se carga con 0x00. La bandera Z se activa.

Ejemplo: CLRF REG

Antes: REG = 0x5A
Después:

CLRW Borra el registro W

Sintaxis: [label] CLRW
Operandos: Ninguno
Operación: 0x00 → W, 1 → Z
Banderas afectadas: Z
Código OP: 00 0001 0xxx xxxx

Descripción: El registro de trabajo W se carga con 0x00. La bandera Z se activa.

Ejemplo: CLRW

Antes: W = 0x5A
Después:

CLRWDT Borra el WDT

Sintaxis: [label] CLRWDT
Operandos: Ninguno
Operación: 0x00 → WDT, 1 → /TO

Banderas afectadas: /TO, /PD
Código OP: 00 0000 0110 0100
Descripción: Esta instrucción borra tanto el WDT como su preescaler. Los bits /TO y /PD del registro de estado se ponen a 1.

Ejemplo: CLRWDT

Después:
Preescales WDT = 0,
/TO = 1, /PD = 1

COMF Complemento de f

Sintaxis: [label] COMF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (/f), 1 → (dest)
Banderas afectadas: Z
Código OP: 00 1001 dfff ffff

Descripción: El registro f es complementado. La bandera Z se activa si el resultado es 0. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: COMF REG,0

Antes: REG = 0x13
Después:

DECF Decremento de f

Sintaxis: [label] DECF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (f) – 1 → (dest)
Banderas afectadas: Z
Código OP: 00 0011 dfff ffff

Descripción: Decrementa en 1 el contenido de f. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: DECF CONT,1

Antes: CONT = 0x01, Z = 0
Después:

DECFSZ Decremento y salto

Sintaxis: [label] DECFSZ f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (f) -1 → d; Salto si R=0
Banderas afectadas: Ninguna
Código OP: 00 1011 dfff ffff

Descripción: Decrementa el contenido del registro f. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f. Si el resultado es 0 salta la siguiente instrucción, en cuyo caso consume 2 ciclos.

Ejemplo:

SI_ES_0
…

GOTO

Sintaxis: [label] GOTO k
Operandos: 0 ≤ k ≤ 2047
Operación: k → PC <8:0>
Banderas afectadas: Ninguna
Código OP: 10 1kkk kkkk kkkk

Descripción: Se trata de un salto incondicional. La parte baja de k se carga en PCL, y la alta en PCLATCH. Ocupa 2 ciclos de reloj.

Ejemplo: Origen GOTO
Destino …

Antes: PC = ORIGEN
Después:

INCF Decremento de f

Sintaxis: [label] INCF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (f) + 1 → (dest)
Banderas afectadas: Z
Código OP: 00 1010 dfff ffff

Descripción: Incrementa en 1 el contenido de f. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: INCF CONT, 1

Antes: CONT = 0xFF, Z = 0
Después:

INCFSZ Incremento y salto

Sintaxis:[label] INCFSZ f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (f) -1 → d; Salto si R=0
Banderas afectadas: Ninguna
Código OP: 00 1111 dfff ffff

Descripción: Incrementa el contenido del registro f. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f. Si la resta es 0 salta la siguiente instrucción, en cuyo caso necesita 2 ciclos.

Ejemplo: INCFSC REG,0
GOTO NO_ES_0
SI_ES_0 *instrucción*
…
NO_ES_0 *salta instrucción anterior*

IORLW W OR literal

Sintaxis: [label] IORLW k
Operandos: 0 ≤ k ≤ 255
Operación: (W) OR (k) → (W)
Banderas afectadas: Z
Código OP: 11 1000 kkkk kkkk

Descripción: Se realiza la operación lógica OR entre el contenido del registro W y k, guardando el resultado en W.

Ejemplo: IORLW 0x35

Antes W = 0x9A
Después:

IORWF

Sintaxis: [label] IORWF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (W) OR (f) → (dest)
Banderas afectadas: Z
Código OP: 00 0100 dfff ffff

Descripción: Realiza la operación lógica OR entre los registros W y f. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: IORWF REG,0

Antes: W = 0x91, REG = 0x13
Después:

MOVLW Cargar literal en W

Sintaxis: [label] MOVLW f
Operandos: 0 ≤ f ≤ 255
Operación: (k) → (W)
Banderas afectadas: Ninguna
Código OP: 11 00xx kkkk kkkk

Descripción: El literal k pasa al registro W.

Ejemplo: MOVLW 0x5A

Después: REG = 0x4F, W = 0x5A

MOVF Mover a f

Sintaxis: [label] MOVF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (f) → (dest)
Banderas afectadas: Z
Código OP: 00 1000 dfff ffff

Descripción: El contenido del registro f se mueve al destino d. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f. Permite verificar el registro, puesto que afecta a Z.

Ejemplo: MOVF REG,0

Después:

MOVWF Mover hacia f

Sintaxis: [label] MOVWF f
Operandos: 0 ≤ f ≤ 127
Operación: W → (f)
Banderas afectadas: Ninguna
Código OP: 00 0000 1fff ffff

Descripción: El contenido del registro W pasa el registro f.

Ejemplo: MOVWF REG,0

Antes: REG = 0xFF, W = 0x4F
Después:

NOP No operación

Sintaxis: [label] NOP
Operandos: Ninguno
Operación: No operación
Banderas afectadas: Ninguna
Código OP: 00 0000 0xx0 0000

Descripción: No realiza operación alguna. En realidad consume un ciclo de instrucción sin hacer nada.

Ejemplo: CLRWDT

Después:
Preescales WDT = 0,
/TO = 1, /PD = 1

RETFIE Retorno de interrupción

Sintaxis: [label] RETFIE
Operandos: Ninguno
Operación: 1 → GIE; TOS → PC
Banderas afectadas: Ninguna
Código OP: 00 0000 0000 1001

Descripción: El PC se carga con el contenido de la cima de la pila (TOS): dirección de retorno. Consume 2 ciclos. Las interrupciones vuelven a ser habilitadas.

Ejemplo: RETFIE

Después:
GIE = 1

RETLW Retorno, carga W

Sintaxis: [label] RETLW k
Operandos: 0 ≤ k ≤ 255
Operación: (k) → (W); TOS → PC
Banderas afectadas: Ninguna
Código OP: 11 01xx kkkk kkkk

Descripción: El registro W se carga con la constante k. El PC se carga con el contenido de la cima de la pila (TOS): dirección de retorno. Consume 2 ciclos.

Ejemplo: RETLW 0x37

Después:
W = 0x37

RETURN Retorno de subrutina

Sintaxis: [label] RETURN
Operandos: Ninguno
Operación: TOS → PC
Banderas afectadas: Ninguna
Código OP: 00 0000 0000 1000

Descripción: El PC se carga con el contenido de la cima de la pila (TOS): dirección de retorno. Consume 2 ciclos.

Ejemplo: RETURN

Después:

RLF Rota f a la izquierda

Sintaxis: [label] RLF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: Rotación a la izquierda
Banderas afectadas: C
Código OP: 00 1101 dfff ffff

Descripción: El contenido de f se rota a la izquierda. El bit de menos peso de f pasa al carry (C), y el carry se coloca en el de mayor peso. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: RRF REG,0

Antes: REG = 1110 0110, C = 0
Después:
W = 1100 1100, C = 1

RRF Rota f a la derecha

Sintaxis: [label] RRF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: Rotación a la derecha
Banderas afectadas: C
Código OP: 00 1100 dfff ffff

Descripción: El contenido de f se rota a la derecha. El bit de menos peso de f pasa al carry (C), y el carry se coloca en el de mayor peso. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: RRF REG,0

Antes: REG = 1110 0110, C = 1
Después:
W = 01110 0011, C = 0

SLEEP Modo bajo consumo

Sintaxis: [label] SLEEP
Operandos: Ninguno
Operación: 0x00 → WDT, 1 → /TO
0 → WDT Preescaler, 0 → / PD
Banderas afectadas: / PD, / TO
Código OP: 00 0000 0110 0011

Descripción: El bit de energía se pone a 0 y el de descanso a 1. El WDT y su preescaler se borran. El oscilador se para. Saldrá de este estado con un RESET o con una interrupción que afecte a este modo.

Ejemplo: SLEEP

Después:
/TO = 1, /PD = 1

SUBLW Resta Literal - W

Sintaxis: [label] SUBLW k
Operandos: 0 ≤ k ≤ 255
Operación: (k) - (W) → (W)
Banderas afectadas: Z, C, DC
Código OP: 11 110x kkkk kkkk
Descripción: Mediante el método del complemento a dos el contenido de W es restado al literal. El resultado se almacena en W.

Ejemplos: SUBLW 0x02

Antes: W=1, C=? Después: W=1 C=1
Antes: W=2, C=? Después: W=0 C=1
Antes: W=3, C=? Después: W=FF C=0 (Resultado negativo)

SUBWF Resta f – W

Sintaxis: [label] SUBWF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: (f) – (W) → (dest)
Banderas afectadas: C, DC, Z
Código OP: 00 0010 dfff ffff
Descripción: Mediante el método del complemento a dos el contenido de W es restado al de f. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplos: SUBWF REG,1
Antes: REG=0x03, W=0x02, C=?
Después: REG=0x01, W=0x4F, C=1
Antes: REG=0x02, W=0x02, C = ?
Después: REG=0x00, W=0x02, C=1
Antes: REG=0x01, W=0x02, C=?
Después: REG=0xFF, W=0x02, C=0 (Resultado negativo)

SWAPF Intercambio de f

Sintaxis: [label] SWAPF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (f <3: 0>) ⇔ (f <7:4>)
Banderas afectadas: Ninguna
Código OP: 00 1110 dfff ffff

Descripción: Los 4 bits de más peso y los 4 de menos son intercambiados. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: SWAPF REG,0

Antes: REG = 0xA5
Después:

XORLW W OR literal

Sintaxis: [label] XORLW k
Operandos: 0 ≤ k ≤ 255
Operación: (W) XOR (k) → (W)
Banderas afectadas: Z
Código OP: 11 1010 kkkk kkkk

Descripción: Se realiza la operación lógica XOR entre el contenido del registro W y k, guardando el resultado en W.

Ejemplo: XORLW 0xAF

Antes: W = 0xB5
Después:

XORWF W AND F

Sintaxis: [label] XORWF f,d
Operandos: d ∈ [0,1]; 0 ≤ f ≤ 127
Operación: (W) XOR (f) → (dest)
Banderas afectadas: Z
Código OP: 00 0110 dfff ffff

Descripción: Realiza la operación lógica XOR entre los registros W y f. Si d es 0 el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: XORWF REG,0

Antes: W = 0xB5, REG = 0xAF
Después:

2.1.4 HABILIDADES

Identificar los diferentes tipos de mnemonicos

A) Identificar mnemonicos suma

B) Identificar mnemonicos resta

C) Identificar mnemonicos movimientos

** nota cualquier duda preguntar al instructor*

2.2 PRÁCTICA 1: IMPLEMENTAR UN GRABADOR DE UN PIC´s

2.2.1 Objetivo

Proporcionar al alumno de ingeniería el diseño para construir un programador de microcontroladores PIC´s que tengan un costo económico, una elaboración sencilla y que sus componentes sean accesibles en el mercado local.

Fundamentos teóricos básicos

Respecto a la herramienta hardware, una indispensable es el grabador, encargado de escribir el programa en la memoria del microcontrolador. Existen grabadores muy completos, capaces de trabajar con muchos modelos de diferentes familias, pero su elevado precio los aleja de los usuarios personales. Para estos últimos existen bastantes versiones de sencillos grabadores, específicos para ciertos modelos de microcontroladores, que gobernados desde un computador personal se ofrecen por un precio ligeramente superior al de un libro.

2.2.3 Esquema eléctrico

Aquí tenemos el diagrama esquemático, que nos muestra como se debe implementar el alambrado del circuito.

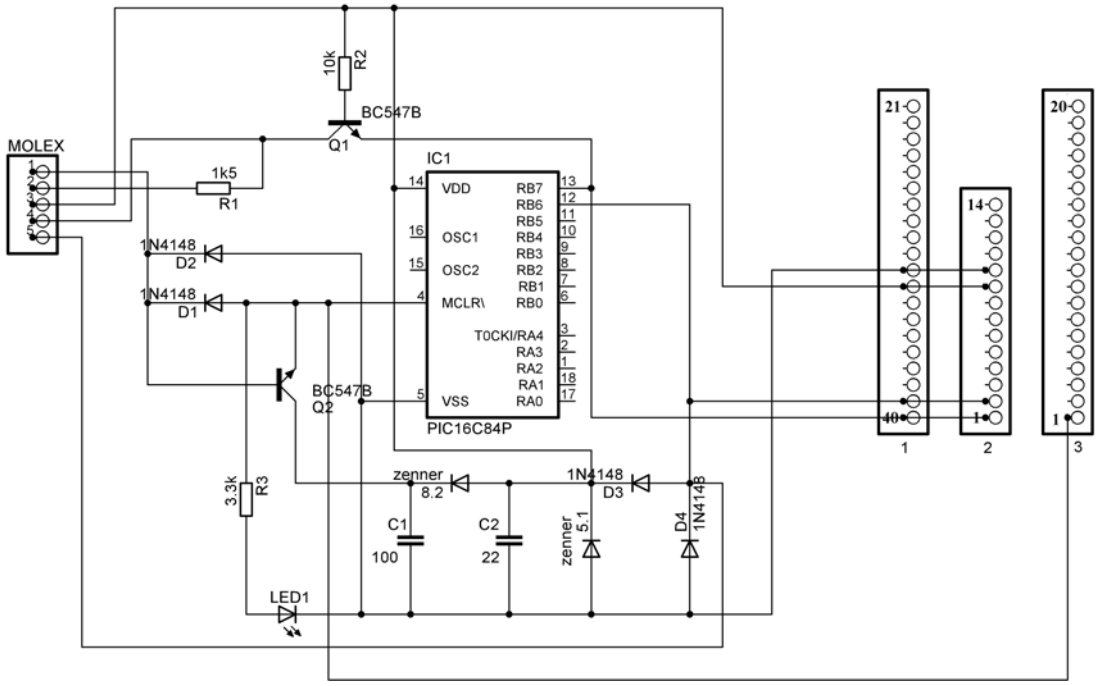


FIG. P1.1 DIAGRAMA ESQUEMATICO DEL GRABADOR DE PIC´s

2.2.4 Material necesario

1 base de 18 pines
2 tiras de 40 pines torneados
1 tira de 20 pines torneados
1 metro cable telefónico
1 conector DB9 hembra
1 conector molex paso 100 de 5 pines completo (hembra, macho en ángulo y zapatas)
2 transistores BC337 o BC547B (T1 y T2)
Resistencias de: 1.5KΩ (R1), 10KΩ (R2) y 3.3KΩ (R3)
1 LED
1 capacitor de 22µF/16V (C2) y otro de 100µF/16V (C1)
4 diodos 1N4148 (D1, D2, D3 y D4)

1 diodo zener de 8.2V a 1W y otro de 5.1V a 1W

2.2.5 Desarrollo de la práctica

PIC's SOPORTADOS

Los chips soportados son todos aquellos que cumplan con la asignación de pines mostrada en la siguiente tabla:

SEÑAL	PIN_BASE 18	PIN_BASE 28	PIN_BASE 40
VDD	14	20	32
VSS	5	19	31
MCLR	4	1	1
RB6	12	27	39
RB7	13	28	40

EJEMPLOS	16F84	16F876	16F77

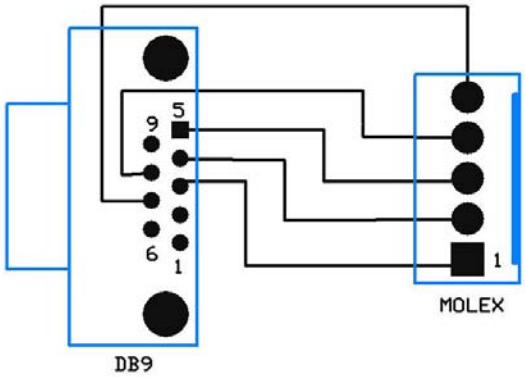
TABLA P1.1 ALGUNOS MICROCONTROLADORES SOPORTADOS PARA EL GRABADOR

Los PIC's de 18 pines deben colocarse en la base de 18 pines*

Los PIC's de 28 pines deben colocarse entre las tiras de pines 1 y 2, colocando el pin menos significativo cerca del número 2*

Los PIC's de 40 pines deben colocarse entre las tiras de pines 1 y 3, colocando el pin menos significativo cerca del número 3*

Para la interfaz necesitamos el conector DB9 hembra, el conector hembra del molex y el cable plano; que se deben conectar como muestra la siguiente figura P1.2 y de acuerdo a la tabla:



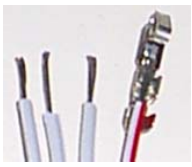
ASIGNACIÓN DE PINES	
MOLEX	**DB9**
1	3
2	4
3	5
4	8
5	7

FIGURA. P1.2 CONECTOR MOLEX

Para colocar las zapatas o terminales del conector molex hembra, necesitamos unas pinzas de punta y unas pinzas para pelar cables o un cutter. Los pasos a seguir son:

Como todos los hilos del cable plano están unidos, debemos tomar el extremo que vamos a utilizar y separar cada uno de los cables –con 3cm. es suficiente-.

Pelamos el extremo de cada cable aproximadamente 3mm.



Colocamos el extremo con la banda roja a nuestra derecha, y ponemos la zapata tal como lo ilustra la imagen.

Con las pinzas de punta doblamos las agarraderas inferiores, para que sujeten al cable en una zona donde aun tiene aislante, tal como lo ilustra la imagen.

Ahora doblamos las agarraderas superiores para que sujeten al cable en la sección que está pelada. No necesitamos –y no debemos- poner soldadura.

Por último, introducimos las zapatas en el conector molex hembra.

SOFTWARE

Después de armar el programador descomprimimos el archivo IC-PROG.ZIP en una carpeta con el mismo nombre, y ejecutamos ICPROG.EXE, elegimos el programador JDM Programmer (lo demás permanece sin cambios). Presionamos el botón OK.

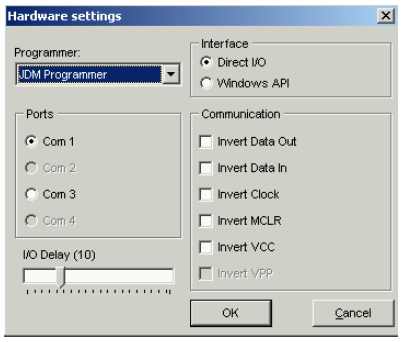


FIG. P1.3 CONFIGURACION DE HARDWARE EN ICPROG

Para usuarios de Windows 95,98 y ME:

Saltamos a la sección de ELECCIÓN DEL DISPOSITIVO.

Para usuarios de Windows XP:

Es necesario usar el controlador ICPROG.SYS, el cual debe estar en el mismo subdirectorio donde se aloja el programa ICPROG.EXE. Para instalarlo:

Colocamos el puntero del ratón sobre el ICPROG.EXE, presionamos el botón secundario, y seleccionamos Propiedades del menú emergente.

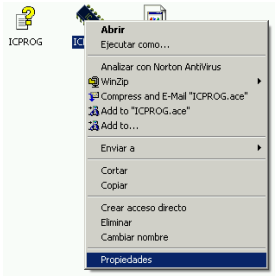


FIG. P1.4 MENU EMERGENTE EN 1CPROG

Elegimos la pestaña Compatibilidad, y en Modo de compatibilidad seleccionamos Windows 2000, presionamos Aplicar y después Aceptar.

FIG.5 MODO DE COMPATIBILIDAD

Ahora ejecutamos ICPROG.EXE y seleccionamos Options del botón Settings del menú.

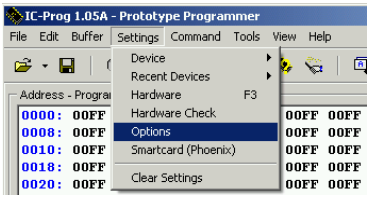


FIG. P1.6 CONFIGURACION

Elegimos la pestaña de Misc, y activamos la opción Enable NT/2000/XP Driver, y

presionamos el botón OK.

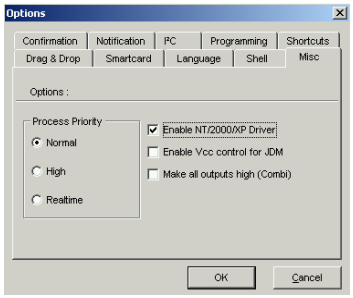


FIG. P1.7 OPCIONES

A continuación presionamos Yes para reiniciar el programa.

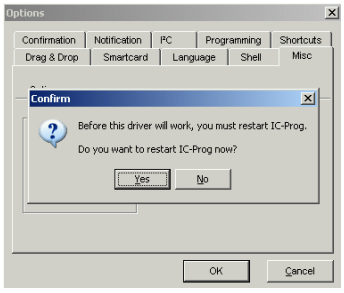


FIG. P1.8 REINICIAR EL PROGRAMA

Nuevamente presionamos Yes para confirmar la instalación.

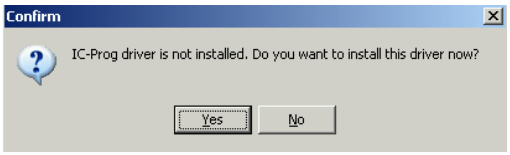


FIG P1.9 CONFIRMAR LA INSTALACION

ELECCIÓN DEL DISPOSITIVO

En el campo de selección de dispositivos escogemos el que vamos a utilizar, por ejemplo: PIC 16F84A.

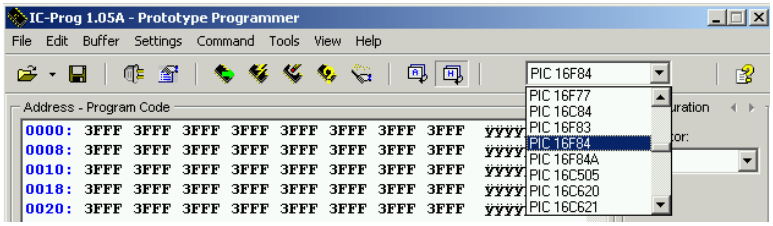


FIG. P1.10 SELECCIÓN DEL PIC

Con los iconos de acceso rápido podemos escoger entre: Leer  , Programar  , Borrar  o Verificar  el contenido de la memoria del microcontrolador.

Para cargar un programa (en formato HEX) en la opción de Archivo seleccionamos Abrir archivo; buscamos su ubicación, y elegimos el programa.

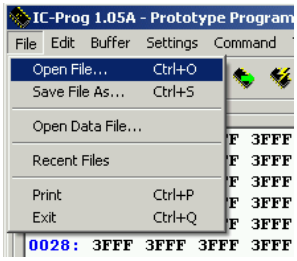


FIG. P1.11 ABRIR ARCHIVO EN FORMATO. HEX

Por último presionamos el icono de programar  .

REFERENCIAS

El programador se basa en la nota de aplicación “In-Circuit Serial Programming™ (ICSP™) Guide” de Microchip (http://www.microchip.com), y en el circuito JDM Programmer

El software IC-PROG, el controlador para Windows XP y el circuito JDM Programmer fueron descargados de la página http://www.ic-prog.com

2.2.6 HABILIDADES

Eje. Verificar el grabador funcione, leyendo un pic pregrabado.

** nota cualquier duda preguntar al instructor*

2.3 PRÁCTICA 2: PROGRAMACION DE UN PIC

2.3.1 Objetivos

Realizar la grabación de un microcontrolador (PIC). Para ello se propone un sencillo ejemplo consiste en el encendido de un led en determinado tiempo.

2.3.2 Fundamentos teóricos básicos

¿Que es un microcontrolador?

Es un circuito integrado programable que contiene todos los componentes de un computador. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna.

El microcontrolador es un computador dedicado. En su memoria solo reside un programa destinado a gobernar una aplicación determinada, sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar.

2.3.3 Esquema eléctrico

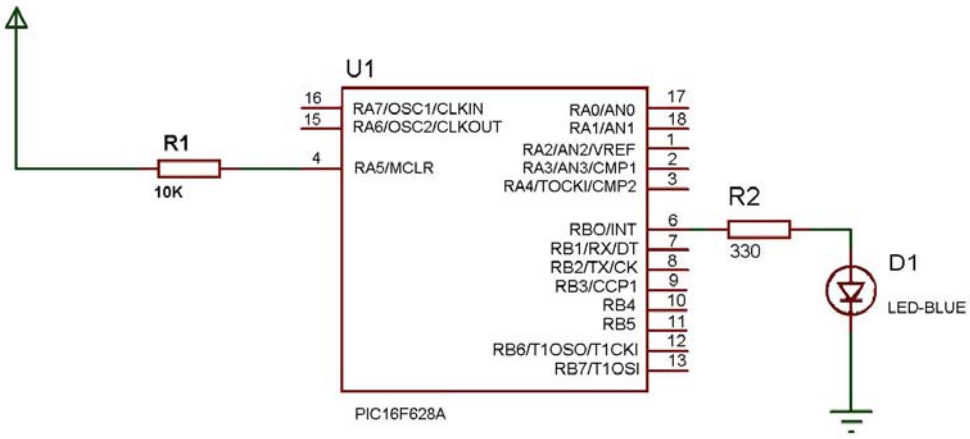


FIG. P2.1 CIRCUITO PARA LA PRACTICA 2

2.3.4 Material necesario

1 led
1 microcontrolador pic 16f628
1 protoboard
1 metro cable telefonico
1 grabador de PIC´s

- ✓ Para la Polarizacion del cto de la FIG.P2.1 pin 5 y 14 son (vss y vdd) respectivamente masa y alimentación.
- ✓ El oscilador ya se encuentra integrado dentro del microcontrolador 16F628

2.3.5 Desarrollo de la práctica

1.- Introducir el siguiente código el microcontrolador 16F628

```
        list p=16f628a
        include p16f628a.inc

        __CONFIG  _CP_OFF  &  _WDT_OFF  &  _BODEN_OFF  &  _PWRTE_ON  &
_INTOSC_OSC_NOCLKOUT & _MCLRE_OFF & _LVP_OFF

ContadorA     EQU   0x20
ContadorB     EQU   0x21
ContadorC     EQU   0x22

;configuracion de puertos

        CLRF PORTA              ;Inicializando el puerto A
        CLRF PORTB              ;Inicializando el puerto B
        MOVLW 0x07          ;Apagando comparadores
        MOVWF CMCON
        BCF STATUS, RP1
        BSF STATUS, RP0         ;Seleccionar Banco de memoria 1
        MOVLW 0xFF              ;direccion de datos puerto A
        MOVWF TRISA
        MOVLW 0xFE              ;direccion de datos puerto B
        MOVWF TRISB
        BCF STATUS, RP0         ; volver al banco de memoria 0


;programa
```

```
CicloPrincipal

        BSF PORTB,0
        CALL Retardo
        BCF PORTB,0
        CALL Retardo

        goto CicloPrincipal

Retardo

;PIC Time Delay = 1,0000020 s with Osc = 4 MHz

        MOVLW                   D'6'
        MOVWF                   ContadorC
        MOVLW                   D'24'
        MOVWF                   ContadorB
        MOVLW                   D'168'
        MOVWF                   ContadorA
CicloRetardo                    DECFSZ          ContadorA,1
        GOTO                    CicloRetardo
        DECFSZ                  ContadorB,1
        GOTO                    CicloRetardo
        DECFSZ                  ContadorC,1
        GOTO                    CicloRetardo
        RETURN

        end
```

2.- Armar el circuito de la figura P2.1 con el microcontrolador grabado.

Nota Se explican los procedimientos que se requieren para simular los microcontroladores en el capitulo tres de esta tesis en las guías rápidas de MPLA y Proteus.

2.3.6 HABILIDADES

Ejemplo.- Modificar el programa para que el led se encienda y apague en la mitad del tiempo

** nota cualquier duda preguntar al instructor*

2.4 PRÁCTICA 3: PROGRAMACION DE UN DISPLAY

2.4.1 Objetivo

Verificar la programación de un display a través de un microcontrolador PIC

2.4.2 Fundamentos teóricos básicos

EL DISPLAY DE 7 SEGMENTOS

Una de las aplicaciones más populares de los LED's es la de señalización. Quizás la mas utilizada sea la de 7 LED's colocadas en forma de ocho tal y como se indica en la figura P3.1 Aunque externamente su forma difiere considerablemente de un diodo LED típico, internamente están constituidos por una serie de diodos LED con unas determinadas conexiones internas. En la figura 9 se indica el esquema eléctrico de las conexiones del interior de un indicador luminoso de 7 segmentos.

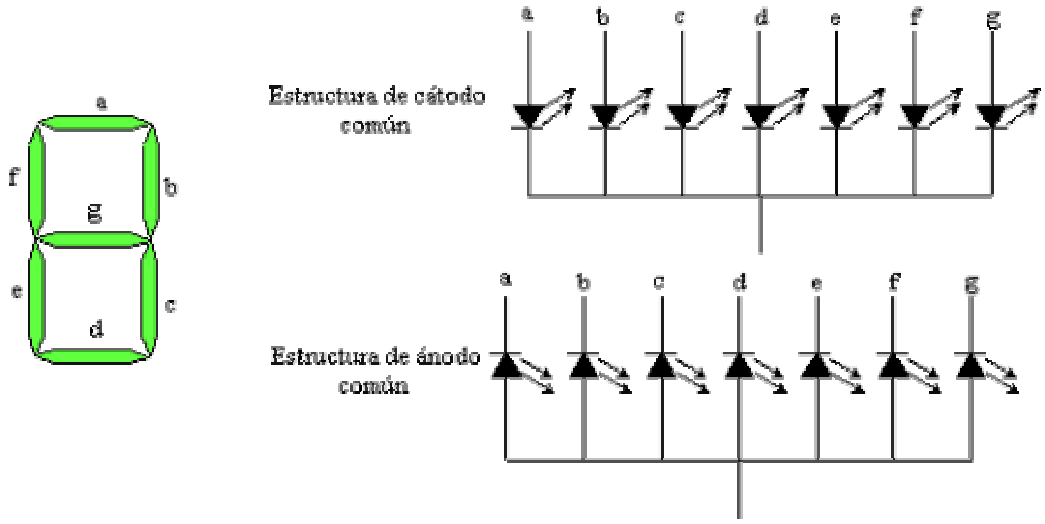


FIG. P3.1 DISPLAY DE 7 SEGMENTO

Display de 7 segmentos. arriba aparecen las dos posibles formas de construir el circuito

La figura se muestra un indicador de siete segmentos. Contiene siete LED rectangulares (a - g), en el que cada uno recibe el nombre de segmento porque forma parte del símbolo que esta mostrando. Con un indicador de siete segmentos se pueden formar los dígitos del 0 al 9, también las letras a, c, e y f y las letras minúsculas b y d. Los entrenadores de microprocesadores usan a menudo indicadores de siete segmentos para mostrar todos los

dígitos del 0 al 9 mas a, b, d, d, e y f .

Polarizando los diferentes diodos, se iluminaran los segmentos correspondientes. De esta manera podemos señalizar todos los números en base 10. Por ejemplo, si queremos representar el número de 1 en el display deberemos mandar señal a los diodos b y b, y los otros diodos deben de tener tensión cero. Esto lo podemos escribir así 0110000(0). El primer digito representa al diodo a, el segundo al b, el tercero al c, y así sucesivamente. Un cero representa que no polarizamos el diodo, es decir no le aplicamos tensión. Un uno representa que el diodo esta polarizado, y por lo tanto, emite luz.

2.4.3 Esquema eléctrico

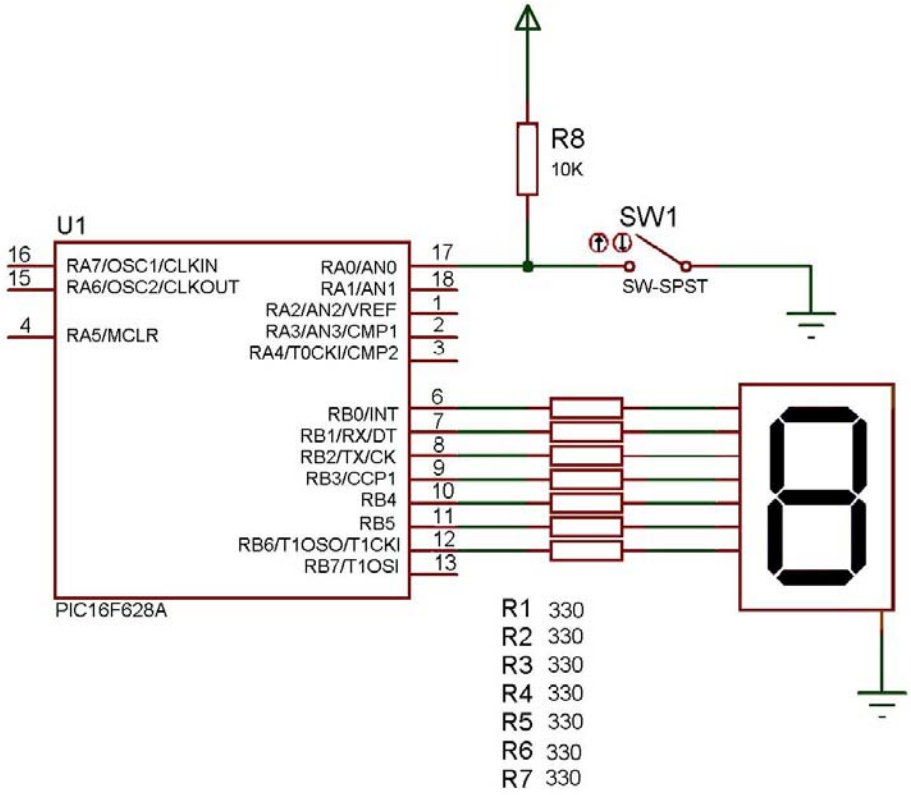


FIG. P3.2 CIRCUITO PARA LA PRACTICA 3

2.4.4 Material necesario

1 display
8 resistencias de 330 ohms
1 protoboard
1 metro cable telefónico
1 grabador de PIC´s
1 microcontrolador PIC 16f628

2.4.5 Desarrollo de la práctica

1.- Grabar el microcontrolador 16f628 con el siguiente programa

```
        list p=16f628a
        include p16f628a.inc

        __CONFIG  _CP_OFF  &  _WDT_OFF  &  _BODEN_OFF  &  _PWRTE_ON  &
_INTOSC_OSC_NOCLKOUT & _MCLRE_OFF & _LVP_OFF



ContadorA     EQU   0x20
ContadorB     EQU   0x21
ContadorC     EQU   0x22
Boton         EQU 0x23
Numero              EQU 0x24


;configuracion de puertos

        CLRF PORTA              ;Inicializando el puerto A
        CLRF PORTB              ;Inicializando el puerto B
```

```
        MOVLW 0x07          ;Apagando comparadores
        MOVWF CMCON
        BCF STATUS, RP1
        BSF STATUS, RP0             ;Seleccionar Banco de memoria 1
        MOVLW 0xFF                  ;direccion de datos puerto A
        MOVWF TRISA
        MOVLW 0x00          ;direccion de datos puerto B
        MOVWF TRISB
        BCF STATUS, RP0             ; volver al banco de memoria 0

        MOVLW d'1'                  ;Inicializar la variable numero (apuntador de tabla)
        MOVWF Numero




;programa

CicloPrincipal


        MOVF Numero,w                   ;sacar valor de tabla
        call Tabla
        MOVWF PORTB
        call Retardo
        BTFSS PORTA,0
        goto decrementa

incrementa:
        INCF Numero,F
        MOVLW d'9'
        SUBWF                           Numero,w
        MOVLW d'1'

        BTFSC STATUS,Z
```

```
        MOVWF Numero
        goto CicloPrincipal



decrementa:
        DECFSZ Numero,F
        goto CicloPrincipal
        MOVLW d'8'
        MOVWF Numero
        goto CicloPrincipal
Retardo

;PIC Time Delay = 1,0000020 s with Osc = 4 MHz

        MOVLW   D'6'
        MOVWF   ContadorC
        MOVLW   D'24'
        MOVWF   ContadorB
        MOVLW   D'168'
        MOVWF   ContadorA
CicloRetardo    DECFSZ  ContadorA,1
        GOTO CicloRetardo
        DECFSZ  ContadorB,1
        GOTO CicloRetardo
        DECFSZ  ContadorC,1
        GOTO CicloRetardo
        RETURN



Tabla:
        ADDWF PCL,F
        NOP
        RETLW B'00111111'  ;0
        RETLW B'00000110'  ;1
        RETLW B'01011011'  ;2
```

```
        RETLW B'01001111'  ;3
        RETLW B'01100110'  ;4
        RETLW B'01101101'  ;5
        RETLW B'01111101'  ;6
        RETLW B'00000111'  ;7
        RETLW B'01111111'  ;8
        RETLW B'01100111'  ;9



        end
```

2.- Armar el circuito de la figura P3.2 con el microcontrolador grabado.

Nota se explican los procedimientos que se requieren para simular los microcontroladores en el capitulo tres de esta tesis en las guías rápidas de MPLA y Proteus.

2.4.6 HABILIDADES

Ejemplo.- Modificar el programa introducir su propio numero de cuenta y se vea proyectado en el display.

** nota cualquier duda preguntar al instructor*

2.5 PRÁCTICA 4: PROGRAMACION DE UN LCD

2.5.1 Objetivo

Verificar la programación de un lcd a través de un microcontrolador PIC

2.5.2 Fundamentos teóricos básicos

LCD (*Liquid Crystal Display*) son las siglas en inglés de **Pantalla de Cristal Líquido**, dispositivo inventado por Jack Janning, quien fue empleado de NCR.
Se trata de un sistema eléctrico de presentación de datos formado por 2 capas conductoras transparentes y en medio un material especial cristalino (cristal líquido) que tienen la capacidad de orientar la luz a su paso.

Cuando la corriente circula entre los electrodos transparentes con la forma a representar (por ejemplo, un segmento de un número) el material cristalino se reorienta alterando su transparencia.

El material base de un LCD lo constituye el cristal líquido, el cual exhibe un comportamiento similar al de los líquidos y unas propiedades físicas anisotrópicas similares a las de los sólidos cristalinos. Las moléculas de cristal líquido poseen una forma alargada y son más o menos paralelas entre sí en la fase cristalina. Según la disposición molecular y su ordenamiento, se clasifican en tres tipos: nemáticos, esméticos y colestéricos. La mayoría de cristales responden con facilidad a los campos eléctricos, exhibiendo distintas propiedades ópticas en presencia o ausencia del campo. El tipo más común de visualizador LCD es, con mucho, el denominado nemático de torsión, término que indica que sus moléculas en su estado desactivado presentan una disposición en espiral. La polarización o no de la luz que circula por el interior de la estructura, mediante la aplicación o no de un campo eléctrico exterior, permite la activación de una serie de segmentos transparentes, los cuales rodean al cristal líquido. Según sus características ópticas, pueden también clasificarse como: reflectivos, transmisivos y transreflectivos.

Las pantallas LCD se encuentran en multitud de dispositivos industriales y de consumo:

máquinas expendedoras, electrodomésticos, equipos de telecomunicaciones, computadoras, etc. Todos estos dispositivos utilizan pantallas fabricadas por terceros de una manera más o menos estandarizada. Cada LCD se compone de una pequeña placa integrada que consta de:

- La propia pantalla LCD.
- Un microchip controlador.
- Una pequeña memoria que contiene una tabla de caracteres.
- Un interfaz de contactos eléctricos, para conexión externa.
- Opcionalmente, una luz trasera para iluminar la pantalla.

El controlador simplifica el uso del LCD proporcionando una serie de funciones básicas que se invocan mediante el interfaz eléctrico, destacando:

- La escritura de caracteres en la pantalla.
- El posicionado de un cursor parpadeante, si se desea.
- El desplazamiento horizontal de los caracteres de la pantalla (*scrolling*).
- Etc.

La memoria implementa un mapa de bits para cada carácter de un juego de caracteres, es decir, cada octeto de esta memoria describe los puntitos o pixels que deben iluminarse para representar un carácter en la pantalla. Generalmente, se pueden definir caracteres a medida modificando el contenido de esta memoria. Así, es posible mostrar símbolos que no están originalmente contemplados en el juego de caracteres.

El interfaz de contactos eléctricos suele ser de tipo paralelo, donde varias señales eléctricas simultáneas indican la función que debe ejecutar el controlador junto con sus parámetros. Por tanto, se requiere cierta sincronización entre estas señales eléctricas.
La luz trasera facilita la lectura de la pantalla LCD en cualquier condición de iluminación ambiental.

Existen dos tipos de pantallas LCD en el mercado: pantallas de texto y pantallas gráficas.

FUNCIONAMIENTO

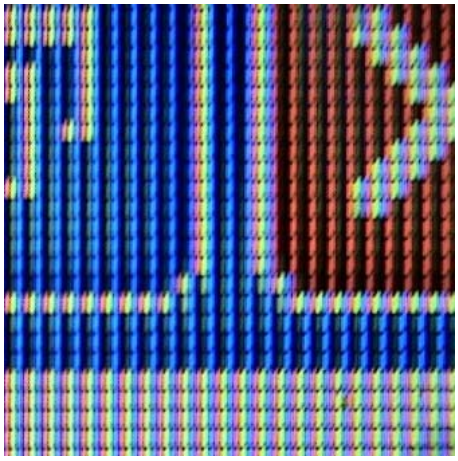


FIG. P4.1 LCD

Imagen donde se pueden observar los pixeles

El funcionamiento de estas pantallas se fundamenta en sustancias que comparten las propiedades de sólidos y líquidos a la vez. Cuando un rayo de luz atraviesa una partícula de estas sustancias tiene necesariamente que seguir el espacio vacío que hay entre sus moléculas como lo haría atravesar un cristal sólido pero a cada una de estas partículas se le puede aplicar una corriente eléctrica que cambie su polarización dejando pasar a la luz o no.

Una pantalla LCD esta formada por 2 filtros polarizados colocados perpendicularmente de manera que al aplicar una corriente eléctrica al segundo de ellos dejaremos pasar o no la luz que ha atravesado el primero de ellos. Para conseguir el color es necesario aplicar tres filtros más para cada uno de los colores básicos rojo, verde y azul y para la reproducción de varias tonalidades de color se deben aplicar diferentes niveles de brillo intermedios entre luz y no luz lo, cual consigue con variaciones en el voltaje que se aplicaba los filtros.

Los LCD de texto son los más baratos y simples de utilizar. Solamente permiten visualizar mensajes cortos de texto. Existen algunos modelos estandarizados en la industria, en función de su tamaño medido en número de líneas y columnas de texto. Existen modelos de una, dos y cuatro filas únicamente. El número de columnas típico es de ocho, dieciséis, veinte y cuarenta caracteres.

El controlador **Hitachi HD44780** se ha convertido en un estándar de industria cuyas especificaciones funcionales son imitadas por la mayoría de los fabricantes. Este controlador cuenta con los siguientes interfaces eléctricos:

- **D0-D7**: Ocho señales eléctricas que componen un bus de datos.

- **R/W**: Una señal que indica si se desea leer o escribir en la pantalla (generalmente solamente se escribe).

- **RS**: Una señal que indica si los datos presentes en D0-D7 corresponden bien a una instrucción, bien a sus parámetros.

- **E**: Una señal para activar o desactivar la pantalla.

- **V0**: Señal eléctrica para determinar el contraste de la pantalla. Generalmente en el rango de cero a cinco voltios. Cuando el voltaje es de cero voltios se obtienen los puntos más oscuros.

- **Vss y Vdd**: Señales de alimentación. Generalmente a cinco voltios. La señal Vss sirve para encender la luz trasera de la pantalla en algunos modelos y Vdd es el voltaje interno.

Estas señales son fácilmente controladas desde un ordenador a través de un interfaz paralelo, típicamente a través del interfaz IEEE 1284, también conocido como "Centronics". El mismo que se utiliza para conectar impresoras.

2.5.3 Esquema eléctrico

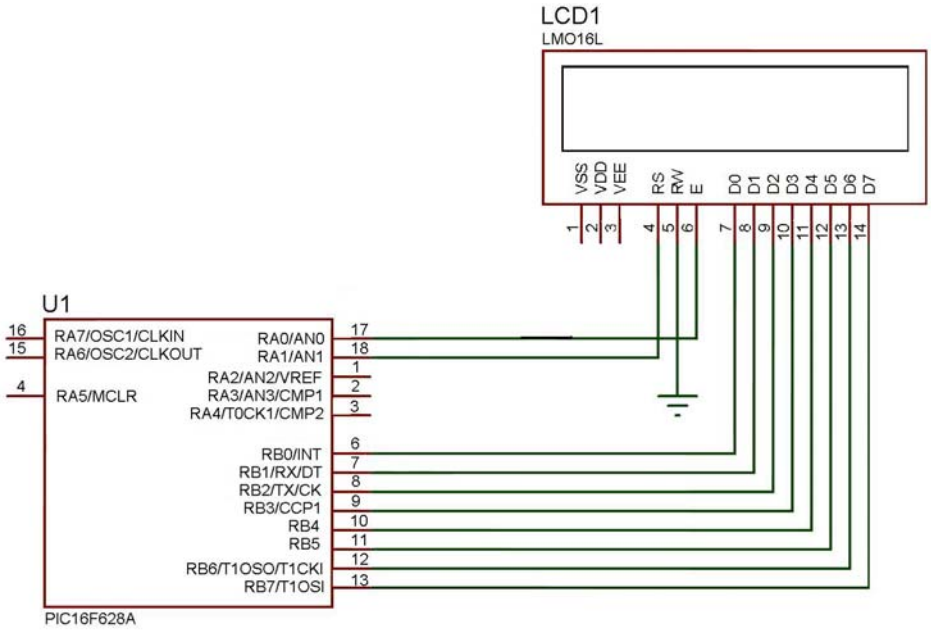


FIG P4.2 CIRCUITO PARA LA PRACTICA 4

2.5.4 Material necesario

1 lcd

1 protoboard

1 metro cable telefónico

1 grabador de PIC´s

1 microcontrolador PIC 16f628

2.5.5 Desarrollo de la práctica

1 Grabar el microcontrolador 16f628 con el siguiente programa.

```
      list p=16f628a
      include p16f628a.inc

      __CONFIG  _CP_OFF  &  _WDT_OFF  &  _BODEN_OFF  &  _PWRTE_ON  &
_INTOSC_OSC_NOCLKOUT & _MCLRE_OFF & _LVP_OFF

#define LcdEnable                PORTA,0
#define LcdRs                                                    PORTA,1
#define LcdData                                                    PORTB

ContadorA                          EQU                                0x20
ContadorB                          EQU                                0x21
ContadorC                          EQU                                0x22
Boton                                                            EQU 0x23
Numero                                                           EQU 0x24


;configuracion de puertos

      CLRF PORTA                                                  ;Inicializando el puerto A
      CLRF PORTB                                                  ;Inicializando el puerto B
      MOVLW 0x07                                                 ;Apagando comparadores
      MOVWF CMCON
      BCF STATUS, RP1
      BSF STATUS, RP0 ;Seleccionar Banco de memoria 1
      MOVLW 0xFC                                                ;direccion de datos puerto A
      MOVWF TRISA
      MOVLW 0x00                                                ;direccion de datos puerto B
      MOVWF TRISB
      BCF STATUS, RP0          ; volver al banco de memoria 0
      CLRF Numero
      BCF                              LcdEnable
      BCF                                LcdRs


;inicializar el LCD

      MOVLW 0x38
      CALL LcdCaracter

      MOVLW 0x0C
      CALL LcdCaracter

      MOVLW 0x82
      CALL LcdCaracter
```

```
        BSF LcdRs

        MOVLW 'H'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter
        MOVLW 'l'
        CALL LcdCaracter
        MOVLW 'a'
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        MOVLW 'M'
        CALL LcdCaracter
        MOVLW 'u'
        CALL LcdCaracter
        MOVLW 'n'
        CALL LcdCaracter
        MOVLW 'd'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter

        BCF LcdRs
        MOVLW 0xC0
        CALL LcdCaracter
        BSF LcdRs

        MOVLW 'C'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter
        MOVLW 'n'
        CALL LcdCaracter
        MOVLW 't'
        CALL LcdCaracter
        MOVLW 'a'
        CALL LcdCaracter
        MOVLW 'n'
        CALL LcdCaracter
        MOVLW 'd'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter



CicloPrincipal:

        BCF LcdRs
        MOVLW 0xCA
        CALL LcdCaracter
```

```
        BSF LcdRs

        MOVLW 0x30
        IORWF Numero,w
        CALL LcdCaracter
        INCF Numero
        MOVLW d'10'
        SUBWF Numero,w
        BTFSC STATUS,Z
        CLRF Numero

        CALL Retardo

        GOTO CicloPrincipal
        SLEEP


LcdCaracter:
        MOVWF LcdData
        MOVLW                   D'255'
        MOVWF                   ContadorA
LcdRetardo                                              DECFSZ
        GOTO                    LcdRetardo

        BSF LcdEnable

        MOVLW                   D'255'
        MOVWF                   ContadorA
LcdRetardo1                                             DECFSZ
        GOTO                    LcdRetardo1

        BCF LcdEnable

        RETURN

Retardo:

;PIC Time Delay = 1,0000020 s with Osc = 4 MHz

        MOVLW                   D'6'
        MOVWF                   ContadorC
        MOVLW                   D'24'
        MOVWF                   ContadorB
        MOVLW                   D'168'
        MOVWF                   ContadorA
CicloRetardo                                            DECFSZ
        GOTO                    CicloRetardo
        DECFSZ                  ContadorB,1
        GOTO                    CicloRetardo
        DECFSZ                  ContadorC,1
        GOTO                    CicloRetardo
        RETURN
```

```
end
```

2.- Armar el circuito de la FIG. P4.2 con el microcontrolador ya grabado

Nota Se explican los procedimientos que se requieren para simular los microcontroladores en el capitulo tres de esta tesis en las guías rápidas de MPLA y Proteus.

2.5.6 HABILIDADES

Ejemplo.- Modificar el programa introducir su propio comentario y se vea proyectado en el LCD.

** nota cualquier duda preguntar al instructor*

2.6 PRÁCTICA 5: PROGRAMACION DE UN TECLADO MATRICIAL

2.6.1 Objetivos

Verificar la programación de un teclado matricial a través de un microcontrolador PIC

2.6.2 Fundamentos teóricos básicos

Teclado matricial de 4 x 3 teclas tipo teléfono que resulta especialmente útil para conectarlo al módulo display lcd con conexión serie + I2C S310118, ya que el conector coincide directamente con las conexiones de dicho módulo. El teclado resulta una opción sencilla y barata para introducir datos, claves de acceso, instrucciones, etc. en cualquier sistema microcontrolador.

2.2.3 Esquema eléctrico

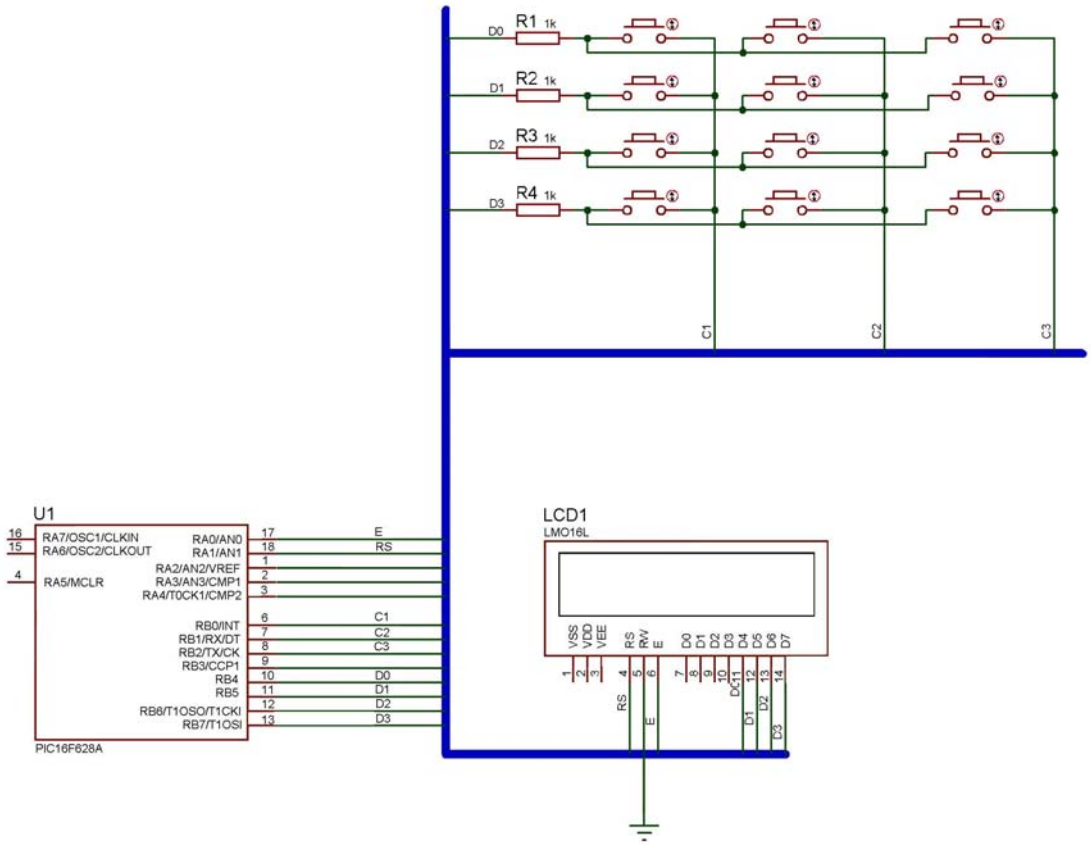


FIG. P5.1 CIRCUITO PARA LA PRACTICA 5

2.2.4 Material necesario

1 lcd

1 protoboard

1 metro cable telefónico

1 grabador de PIC´s

1 microcontrolador PIC 16f628

4 resistencias de 330 ohms

12 push botón

2.2.5 Desarrollo de la práctica

1.- Grabar el microcontrolador 16f628 con el siguiente código

```
        list p=16f628a
        include p16f628a.inc

        __CONFIG  _CP_OFF  &  _WDT_OFF  &  _BODEN_OFF  &  _PWRTE_ON  &
_INTOSC_OSC_NOCLKOUT & _MCLRE_OFF & _LVP_OFF

#define LcdEnable     PORTA,0
#define LcdRs         PORTA,1
#define LcdData               PORTB

ContadorA     EQU   0x20
ContadorB     EQU   0x21
ContadorC     EQU   0x22
LcdTemp               EQU 0x23


;configuracion de puertos

        CLRF PORTA                  ;Inicializando el puerto A
        CLRF PORTB          ;Inicializando el puerto B
        MOVLW 0x07          ;Apagando comparadores
        MOVWF CMCON
        BCF STATUS, RP1
        BSF STATUS, RP0 ;Seleccionar Banco de memoria 1
        MOVLW 0xFC                  ;direccion de datos puerto A
        MOVWF TRISA
```

```
MOVLW 0x0F                  ;direccion de datos puerto B
MOVWF TRISB
BCF STATUS, RP0    ; volver al banco de memoria 0
BCF LcdRs
BCF LcdEnable

MOVLW 0x02
CALL LcdCaracter
MOVLW 0x28
CALL LcdCaracter
MOVLW 0x0C
CALL LcdCaracter

BSF LcdRs
MOVLW 'P'
CALL LcdCaracter
MOVLW 'r'
CALL LcdCaracter
MOVLW 'e'
CALL LcdCaracter
MOVLW 's'
CALL LcdCaracter
MOVLW 'i'
CALL LcdCaracter
MOVLW 'o'
CALL LcdCaracter
MOVLW 'n'
CALL LcdCaracter
MOVLW 'a'
CALL LcdCaracter
MOVLW ' '
CALL LcdCaracter
MOVLW 't'
CALL LcdCaracter
MOVLW 'e'
```

```
        CALL LcdCaracter
        MOVLW 'c'
        CALL LcdCaracter
        MOVLW 'l'
        CALL LcdCaracter
        MOVLW 'a'
        CALL LcdCaracter
        MOVLW 's'
        CALL LcdCaracter

CicloPrincipal
        BCF LcdRs
        MOVLW 0xC8
        CALL LcdCaracter
        BSF LcdRs
        CALL Teclado
        CALL LcdCaracter
        GOTO CicloPrincipal



sleep

;_____________________________________________________
Teclado

        MOVLW B'00010000'
        MOVWF PORTB
                BTFSC       PORTB,0
                RETLW       '1'
                BTFSC       PORTB,1
                RETLW       '2'
                BTFSC       PORTB,2
                RETLW       '3'
```

```
        MOVLW B'00100000'
        MOVWF PORTB
                BTFSC       PORTB,0
                RETLW       '4'
                BTFSC       PORTB,1
                RETLW       '5'
                BTFSC       PORTB,2
                RETLW       '6'

        MOVLW B'01000000'
        MOVWF PORTB
                BTFSC       PORTB,0
                RETLW       '7'
                BTFSC       PORTB,1
                RETLW       '8'
                BTFSC       PORTB,2
                RETLW       '9'

        MOVLW B'10000000'
        MOVWF PORTB
                BTFSC       PORTB,0
                RETLW       '*'
                BTFSC       PORTB,1
                RETLW       '0'
                BTFSC       PORTB,2
                RETLW       '#'
                RETLW   ' '


;_____________________________________________________
LcdCaracter:

        MOVWF LcdTemp
        MOVLW 0xF0
        ANDWF LcdTemp,w
```

```
        MOVWF LcdData
        MOVLW   D'255'
        MOVWF   ContadorA
LcdRetardo      DECFSZ  ContadorA,1
        GOTO LcdRetardo

        BSF LcdEnable

        MOVLW   D'255'
        MOVWF   ContadorA
LcdRetardo1     DECFSZ  ContadorA,1
        GOTO LcdRetardo1

                BCF LcdEnable

        SWAPF LcdTemp,f
        MOVLW 0xF0
        ANDWF LcdTemp,w
        MOVWF LcdData
        MOVLW   D'255'
        MOVWF   ContadorA
LcdRetardo2     DECFSZ  ContadorA,1
        GOTO LcdRetardo2

        BSF LcdEnable

        MOVLW   D'255'
        MOVWF   ContadorA
LcdRetardo3     DECFSZ  ContadorA,1
        GOTO LcdRetardo3

        BCF LcdEnable

        RETURN
;_____________________________________________________________
```

```
;PIC Time Delay = 1,0000020 s with Osc = 4 MHz

        MOVLW       D'6'
        MOVWF       ContadorC
        MOVLW       D'24'
        MOVWF       ContadorB
        MOVLW       D'168'
        MOVWF       ContadorA
CicloRetardo  DECFSZ      ContadorA,1
        GOTO CicloRetardo
        DECFSZ      ContadorB,1
        GOTO CicloRetardo
        DECFSZ      ContadorC,1
        GOTO CicloRetardo
        RETURN
        end
```

2.-Armar el circuito DE LA FIG. P5.1 con el micrograbado

Nota se explican los procedimientos que se requieren para simular los microcontroladores en el capitulo tres de esta tesis en las guías rápidas de MPLA y Proteus.

2.5.6 HABILIDADES

Ejemplo.- Modificar el programa introducir nuevos comandos en el teclado matricial.

** nota cualquier duda preguntar al instructor*

2.7 PRÁCTICA 6: PROGRAMACION DE MOTORES A PASOS

2.7.1 Objetivos

Controlar los pulsos electrónicos del microcontrolador y con ello gobernar el motor a pasos de sus dos sentidos del reloj.

2.7.2 Fundamentos teóricos básicos

En numerosas ocasiones es necesario convertir la energía eléctrica en energía mecánica, esto se puede lograr, por ejemplo, usando los motores de corriente continua. Pero cuando lo deseado es posicionamiento con un elevado grado de exactitud y/o una muy buena regulación de la velocidad, se puede contar con una gran solución: utilizar un motor paso a paso.

La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°, es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360°.

PARÁMETROS DE LOS MOTOREA PASO A PASO

Desde el punto de vista mecánico y eléctrico, es conveniente conocer el significado de algunas de las principales características y parámetros que se definen sobre un motor paso a paso:

- Par dinámico de trabajo (Working Torque): Depende de sus características dinámicas y es el momento máximo que el motor es capaz de desarrollar sin perder paso, es decir, sin dejar de responder a algún impulso de excitación del estator y dependiendo, evidentemente, de la carga.

Generalmente se ofrecen, por parte del fabrican, curvas denominadas de arranque sin error (pull-in) y que relaciona el par en función el número de pasos
Hay que tener en cuenta que, cuando la velocidad de giro del motor aumenta, se produce un aumento de la f.c.e.m. en él generada y, por tanto, una disminución de la corriente absorbida por los bobinados del estator, como consecuencia de todo ello, disminuye el par motor.

- Par de mantenimiento (Holding Torque): Es el par requerido para desviar, en régimen de excitación, un paso el rotor cuando la posición anterior es estable; es mayor que el par dinámico y actúa como freno para mantener el rotor en una posición estable dada

- Para de detención (Detention Torque): Es una par de freno que siendo propio de los motores de imán permanente, es debida a la acción del rotor cuando los devanados del estator están desactivados.

- Angulo de paso (Step angle): Se define como el avance angular que se produce en el motor por cada impulso de excitación. Se mide en grados, siendo los pasos estándar más importantes los siguientes:

Grados por impulso de excitación	**Nº de pasos por vuelta**
0,72º	500
1,8º	200
3,75º	96
7,5º	48
15º	24

FIGP6.1

2.7.3 Esquema eléctrico

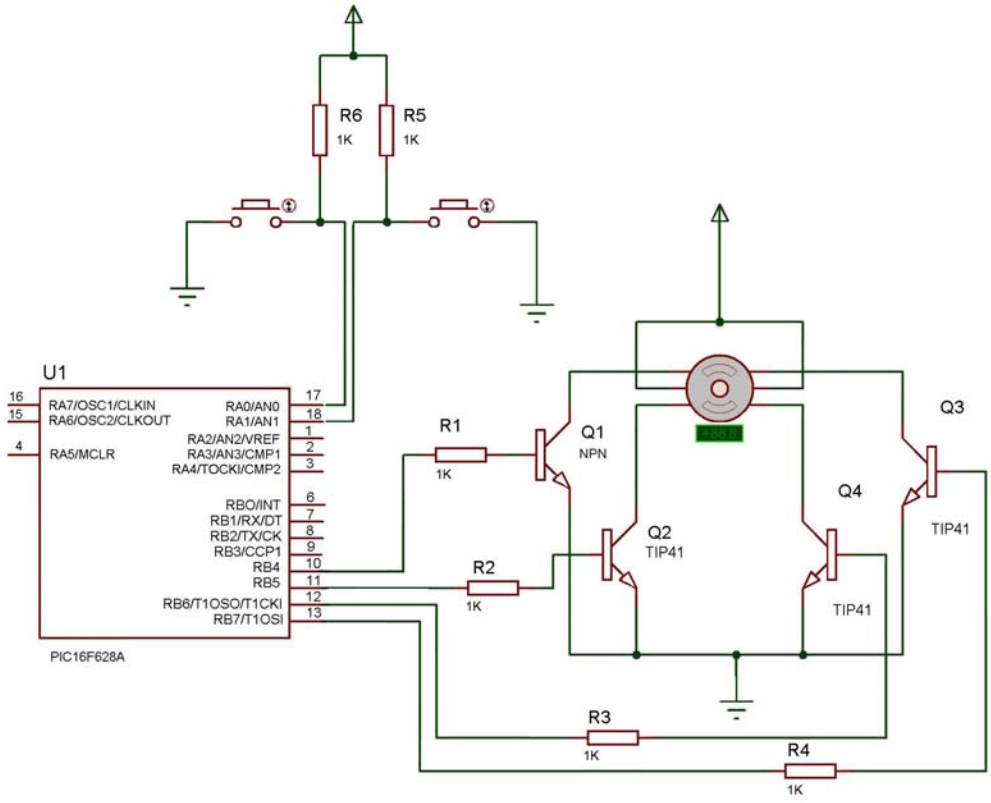


FIG.P6.2 CIRCUITO DE LA PRACTICA 6

2.7.4 Material necesario

1 protoboard
1 metro cable telefónico
1 grabador de PIC´s
1 microcontrolador PIC 16f628
6 resistencias de 1 kohms
2 push botón
4 transistores tip41
1 motor a pasos

2.2.5 DESARROLLO DE LA PRÁCTICA

1.- Grabar el microcontrolador con el siguiente programa

```
        list p=16f628a
        include p16f628a.inc

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
_INTOSC_OSC_NOCLKOUT & _MCLRE_OFF & _LVP_OFF


ContadorA     EQU   0x20
ContadorB     EQU   0x21
ContadorC     EQU   0x22
Paso          EQU 0x23

;configuracion de puertos

        CLRF PORTA                  ;Inicializando el puerto A

        CLRF PORTB          ;Inicializando el puerto B
        MOVLW 0x07          ;Apagando comparadores
        MOVWF CMCON
        BCF STATUS, RP1
        BSF STATUS, RP0 ;Seleccionar Banco de memoria 1
        MOVLW 0xFF                  ;direccion de datos puerto A
        MOVWF TRISA
        MOVLW 0x0F                  ;direccion de datos puerto B
        MOVWF TRISB
        BCF STATUS, RP0    ; volver al banco de memoria 0
```

```
        MOVLW 0x01
        MOVWF Paso

CicloPrincipal

        MOVF Paso,w
        call TablaMotor
        MOVWF PORTB
        call Retardo

        BTFSS PORTA,1
        GOTO Reloj
        BTFSS PORTA,0
        GOTO ContraReloj
        GOTO CicloPrincipal

Reloj:
        INCF Paso,F
        MOVLW d'9'
        SUBWF Paso,w
        MOVLW d'1'
        BTFSC STATUS,Z
        MOVWF Paso
        GOTO CicloPrincipal


ContraReloj:
        DECFSZ Paso,F
        goto CicloPrincipal
        MOVLW d'8'
        MOVWF Paso
        goto CicloPrincipal
```

```
        sleep


;_____________________________________________________

TablaMotor
        ADDWF       PCL,F
        NOP
        RETLW B'10000000'
        RETLW B'11000000'
        RETLW B'01000000'
        RETLW B'01100000'
        RETLW B'00100000'
        RETLW B'00110000'
        RETLW B'00010000'
        RETLW B'10010000'
;_____________________________________________________


;PIC Time Delay = 0,2000030 s with Osc = 4 MHz

Retardo
        MOVLW       D'1'            ;2
        MOVWF       ContadorC
        MOVLW       D'127'          ;5
        MOVWF       ContadorB
        MOVLW       D'185'
        MOVWF       ContadorA
CicloRetardo  DECFSZ       ContadorA,1
        GOTO CicloRetardo
        DECFSZ      ContadorB,1
        GOTO CicloRetardo
```

```
        DECFSZ      ContadorC,1
        GOTO CicloRetardo
        RETURN

        end
```

2.- Armar el circuito de la figura p62.2 con el microcontrolador grabado

Nota Se explican los procedimientos que se requieren para simular los microcontroladores en el capitulo tres de esta tesis en las guías rápidas de MPLA y Proteus.

2.7.6 HABILIDADES

Ejemplo.- Modificar el programa y aumentar la velocidad de los pulsos eléctricos

** nota cualquier duda preguntar al instructor*

CAPÍTULO 3

3. PROPUESTA DE PRÁCTICAS EXTRAS.

La necesidad de simular los microcontroladores, su rendimiento ante otros dispositivos electrónicos y su adaptación entre ellos, llevo a este capitulo; a parte de considerar el tema IX de la materia de microprocesadores que tiene como objetivo integrar los conocimientos adquiridos para realizar aplicaciones en el campo de la ingeniería.

Por lo que, se propone la interfase entre el microcontrolador, un motor de pasos y un LCD, esta aplicación tiene la finalidad de proporcionar un control más exacto para un sistema electromecánico que se puede adaptar a varias aplicaciones.

3.1 PRÁCTICA 7: PROGRAMACION DE MOTORES A PASOS CON UN LCD.

3.1.1 Objetivo.

Verificar los sentidos de los motores a pasos a través de un LCD.

3.1.2 FUNDAMENTOS TEORICOS BASICOS.

IDENTIFICANDO LOS CABLES EN MOTORES P-P BIPOLARES:

Para el caso de motores paso a paso bipolares (generalmente de 4 cables de salida), la identificación es más sencilla. Simplemente tomando un tester en modo ohmetro (para

medir resistencias), podemos hallar los pares de cables que corresponden a cada bobina, debido a que entre ellos deberá haber continuidad (en realidad una resistencia muy baja). Luego solo deberemos averiguar la polaridad de la misma, la cual se obtiene fácilmente probando. Es decir, si conectado de una manera no funciona, simplemente damos vuelta los cables de una de las bobinas y entonces ya debería funcionar correctamente. Si el sentido de giro es inverso a lo esperado, simplemente se deben invertir las conexiones de ambas bobinas y el H-Bridge.

PARA RECORDAR

- Un motor de paso con 5 cables es casi seguro de 4 fases y unipolar.
- Un motor de paso con 6 cables también puede ser de 4 fases y unipolar, pero con 2 cables comunes para alimentación. pueden ser del mismo color.
- Un motor de pasos con solo 4 cables es comúnmente bipolar.

3.1.3 ESQUEMA ELECTRICO.

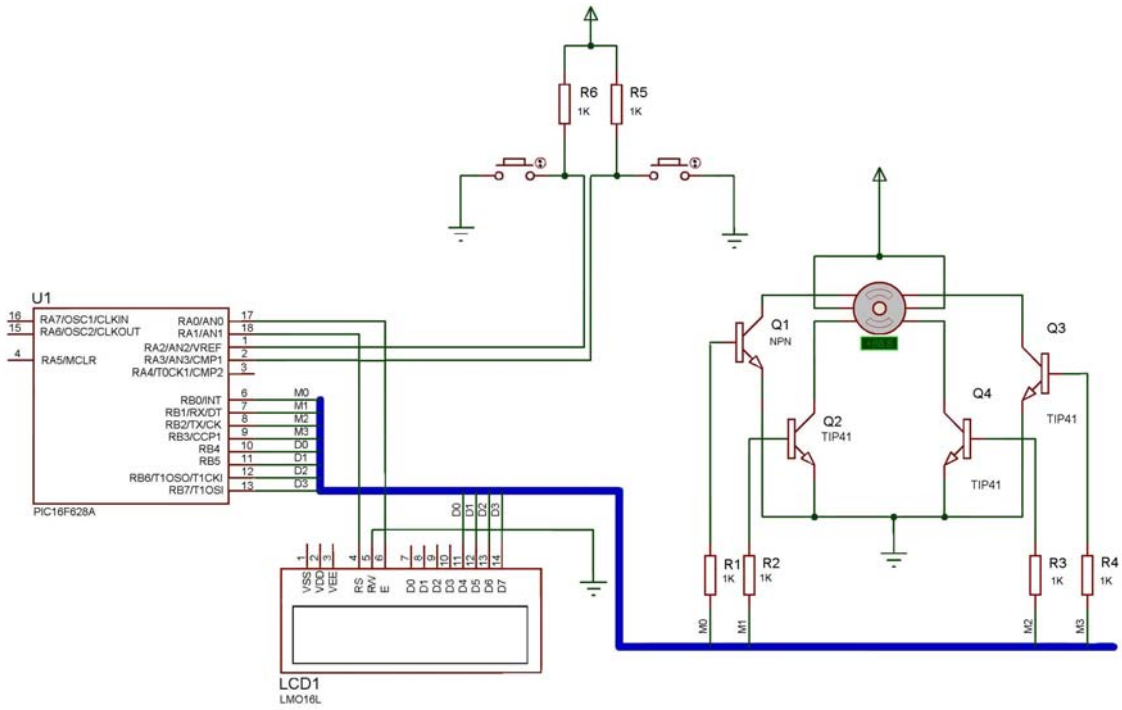


FIG P8.1 circuito de la practica8

3.1.4 Material necesario

1 lcd
1 protoboard
1 metro cable telefónico
1 grabador de PIC´s
1 microcontrolador PIC 16f628
6 resistencias de 1 kohms
2 push botón
4 transistores tip41
1 motor a pasos

3.1.5 DESARROLLO DE LA PRÁCTICA

1.- grabar el microcontrolador con el siguiente programa

```
        list p=16f628a
        include p16f628a.inc

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _INTOSC_OSC_NOCLKOUT & _MCLRE_OFF & _LVP_OFF

#define LcdEnable     PORTA,0
#define LcdRs         PORTA,1
#define LcdData               PORTB

ContadorA     EQU   0x20
ContadorB     EQU   0x21
ContadorC     EQU   0x22
LcdTemp               EQU 0x23
```

```
Paso            EQU 0x24

;configuracion de puertos

        CLRF PORTA              ;Inicializando el puerto A
        CLRF PORTB          ;Inicializando el puerto B
        MOVLW 0x07          ;Apagando comparadores
        MOVWF CMCON
        BCF STATUS, RP1
        BSF STATUS, RP0 ;Seleccionar Banco de memoria 1
        MOVLW 0xFC              ;direccion de datos puerto A
        MOVWF TRISA
        MOVLW 0x00          ;direccion de datos puerto B
        MOVWF TRISB
        BCF STATUS, RP0    ; volver al banco de memoria 0
        BCF LcdRs
        BCF LcdEnable

        MOVLW 0x02
        CALL LcdCaracter
        MOVLW 0x28
        CALL LcdCaracter
        MOVLW 0x0C
        CALL LcdCaracter

        MOVLW 0x01
        MOVWF Paso

        BSF LcdRs
        CALL MensajeSentido


CicloPrincipal

        MOVF Paso,w
```

```
        call TablaMotor
        MOVWF PORTB
        call Retardo

        BTFSS PORTA,2
        GOTO Reloj
        BTFSS PORTA,3
        GOTO ContraReloj
        CALL MensajeParado
        goto CicloPrincipal

Reloj:
        CALL MensajeReloj
        INCF Paso,F
        MOVLW d'9'
        SUBWF Paso,w
        MOVLW d'1'
        BTFSC STATUS,Z
        MOVWF Paso
        goto CicloPrincipal


ContraReloj:
        CALL MensajeContraReloj
        DECFSZ Paso,F
        goto CicloPrincipal
        MOVLW d'8'
        MOVWF Paso
        goto CicloPrincipal
```

```
        sleep

;_____________________________________________________________
MensajeSentido

        MOVLW 'S'
        CALL LcdCaracter
        MOVLW 'e'
        CALL LcdCaracter
        MOVLW 'n'
        CALL LcdCaracter
        MOVLW 't'
        CALL LcdCaracter
        MOVLW 'i'
        CALL LcdCaracter
        MOVLW 'd'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter
        RETURN

MensajeReloj
        BCF LcdRs
        MOVLW 0xC0
        CALL LcdCaracter
        BSF LcdRs

        MOVLW ' '
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        MOVLW 'r'
        CALL LcdCaracter
```

```
        MOVLW 'e'
        CALL LcdCaracter
        MOVLW 'l'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter
        MOVLW 'j'
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        MOVLW ' '
        RETURN

MensajeContraReloj
        BCF LcdRs
        MOVLW 0xC0
        CALL LcdCaracter
        BSF LcdRs

        MOVLW 'C'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter
        MOVLW 'n'
        CALL LcdCaracter
        MOVLW 't'
        CALL LcdCaracter
        MOVLW 'r'
        CALL LcdCaracter
        MOVLW 'a'
        CALL LcdCaracter
        MOVLW 'r'
        CALL LcdCaracter
```

```
        MOVLW 'e'
        CALL LcdCaracter
        MOVLW 'l'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter
        MOVLW 'j'
        CALL LcdCaracter
        RETURN

MensajeParado
        BCF LcdRs
        MOVLW 0xC0
        CALL LcdCaracter
        BSF LcdRs

        MOVLW ' '
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        MOVLW 'P'
        CALL LcdCaracter
        MOVLW 'a'
        CALL LcdCaracter
        MOVLW 'r'
        CALL LcdCaracter
        MOVLW 'a'
        CALL LcdCaracter
        MOVLW 'd'
        CALL LcdCaracter
        MOVLW 'o'
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        MOVLW ' '
```

```
        CALL LcdCaracter
        MOVLW ' '
        CALL LcdCaracter
        RETURN


;_____________________________________________________

TablaMotor
        ADDWF       PCL,F
        NOP
        RETLW B'00001000'
        RETLW B'00001100'
        RETLW B'00000100'
        RETLW B'00000110'
        RETLW B'00000010'
        RETLW B'00000011'
        RETLW B'00000001'
        RETLW B'00001001'
;_____________________________________________________
LcdCaracter:

        MOVWF LcdTemp
        MOVLW 0xF0
        ANDWF LcdTemp,w
        MOVWF LcdData
        MOVLW       D'255'
        MOVWF       ContadorA
LcdRetardo          DECFSZ      ContadorA,1
        GOTO LcdRetardo

        BSF LcdEnable

        MOVLW       D'255'
        MOVWF       ContadorA
```

```
LcdRetardo1         DECFSZ      ContadorA,1
        GOTO LcdRetardo1

                BCF LcdEnable

        SWAPF LcdTemp,f
        MOVLW 0xF0
        ANDWF LcdTemp,w
        MOVWF LcdData
        MOVLW       D'255'
        MOVWF       ContadorA
LcdRetardo2         DECFSZ      ContadorA,1
        GOTO LcdRetardo2

        BSF LcdEnable

        MOVLW       D'255'
        MOVWF       ContadorA
LcdRetardo3         DECFSZ      ContadorA,1
        GOTO LcdRetardo3

        BCF LcdEnable

        RETURN
;_____________________________________________________________________
;PIC Time Delay = 0,2000030 s with Osc = 4 MHz

Retardo
        MOVLW       D'1'
        MOVWF       ContadorC
        MOVLW       D'127'      ;5
        MOVWF       ContadorB
        MOVLW       D'185'
        MOVWF       ContadorA
CicloRetardo  DECFSZ      ContadorA,1
```

```
        GOTO CicloRetardo
        DECFSZ      ContadorB,1
        GOTO CicloRetardo
        DECFSZ      ContadorC,1
        GOTO CicloRetardo
        RETURN

        end
```

2.- armar el circuito FIG P8.1 con el microcontrolador grabado

Nota se explican los procedimientos que se requieren para simular los microcontroladores al final de este capitulo en las guías rápidas de MPLA y Proteus.

2.8.6 HABILIDADES

Ejemplo.- modificar el programa e inverir los sentidos del motor con respectos a los push boton

- *nota cualquier duda preguntar al instructor*

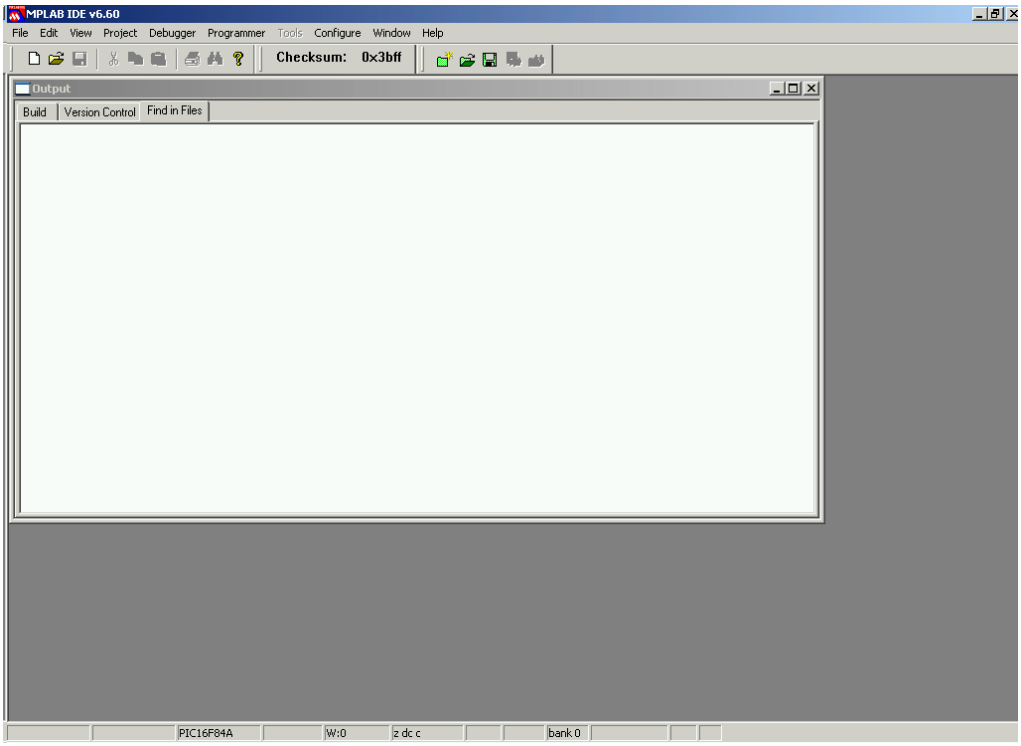
3.2 GUIA RAPIDA PARA EL USO DEL ENTORNO MPLAB IDE VERSION 6.6

3.2.1 INTRODUCCION

El MPLAB es un entorno de desarrollo integrado que le permite escribir y codificar los microcontroladores PIC de Microchip para ejecutarlos. El MPLAB incluye un editor de texto, funciones para el manejo de proyectos, un simulador interno y una variedad de herramientas que lo ayudarán a mantener y ejecutar su aplicación. También provee una interfase de usuario para todos los productos con lenguaje Microchip, programadores de dispositivos, sistemas emuladores y herramientas de tercer orden.

3.2.2 CREAR UN NUEVO PROYECTO SIMPLE

La instalación del entorno no requiere de procedimientos especiales, sólo hay que seguir algunas instrucciones del asistente de instalación. Ejecutamos al entorno y lo primero que observaremos es la siguiente pantalla:



Para comenzar a trabajar, en la barra de menú presionamos: **Proyect > Proyect Wizard…**, en la pantalla de bienvenida presionamos **Siguiente**.

Paso 1: elegimos en el menú desplegable el modelo del PIC con que vamos a trabajar – para el

ejemplo será el PIC16F84A- y presionamos **Siguiente**.

Paso 2: Escogemos el ensamblador; donde dice **Active Toolsuite** seleccionamos **Microchip MPSAM Toolsuite**; Si en la parte de **Toolsuite Contents** alguna de las aplicaciones está tachada, debemos dar un clic sobre ella, y en la parte que dice **Location** presionar **Browse…** para buscar su ruta de instalación. Por ejemplo, si está tachada la aplicación **MPLIB librarian (mplib.exe)** es muy probable que la ubicación de la aplicación este en la ruta *C:\Archivos de programa\Microchip\MPASM Suite*, en está ruta veremos al archivo ejecutable **MPLIB**, damos doble clic. Si ya no hay aplicaciones tachadas, presionamos **Siguiente**.

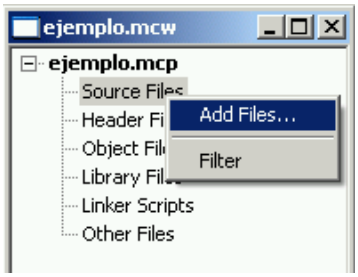
Paso 3: Donde dice Proyect Name escribimos un nombre para el proyecto – yo lo nombrare *ejemplo*-; luego hay que seleccionar el lugar donde se guardará el proyecto presionando **Browse…** y cuando elijamos la ruta para guardar al proyecto presionamos **Select**; por último presionamos **Siguiente**.

Paso 4: Este paso sirve para agregar archivos existentes a nuestro proyecto; no agregamos archivos, por tanto, sólo presionamos Siguiente.

La siguiente ventana nos muestra un resumen de los parámetros que elegimos para nuestro proyecto; si no deseamos realizar cambios presionamos **Finalizar**.

Ahora en la barra de herramientas presionamos el icono  **New File**, con lo cual aparece una ventana en la que vamos a capturar el código de nuestro programa en lenguaje ensamblador. Colocamos la ventana en el lugar de nuestra preferencia y procedemos a capturar nuestro código, posteriormente lo guardamos presionando el icono  **Save**, ahora escogemos el nombre y la ruta para que guarde el código -con extensión asm- y presionamos **Guardar**.

El siguiente paso es integrar a nuestro proyecto el ó los códigos que deseamos ensamblar. En la ventana del proyecto –ubicada a la izquierda- posicionamos el puntero del ratón sobre la rama que dice **Source Files** y damos clic con el botón secundario; ahora del



menú emergente presionamos **Add Files…**, buscamos el archivo con extensión asm que acabamos de guardar y presionamos **Abrir**.

Para ensamblar nuestro código presionamos el icono **Make**; aparecerá la ventana **Output**, mostrando algunos mensajes, si al final del mensaje dice "BUILD SUCEDED: *fecha*", nuestro código –al menos sintácticamente- está correcto; pero si dice "BUILD FAILED: *fecha*", nuestro código tiene errores de sintaxis, para corregirlos posicionamos el puntero sobre el lugar donde dice "Error[xx]: *ruta y posible causa del error*" y damos doble clic, con esto se abre la ventana de edición, y una flecha verde indica la línea donde está el problema. Después presionamos el icono **Save** para guardar los cambios y ensamblamos nuevamente; seguimos así, hasta que ya no existan errores.
Antes de simular, vamos a revisar la sintaxis del ensamblador, y después analizamos un código de ejemplo.

El ensamblador exige una tabulación mínima.
La definición de variables pueden escribirse en la primera columna
Las etiquetas van en la primera columna
Las directivas y mnemónicos **deben ir** a partir de la segunda columna.

Las cifras se expresan de acuerdo a la siguiente tabla, y como ejemplo se utiliza el número 12 decimal para todas las bases:

BASE	**REPRESENTACIÓN**
Decimal	d'12'
Hexadecimal	0x0c ; h'0c' ; 0ch
Binario	b'00001100'

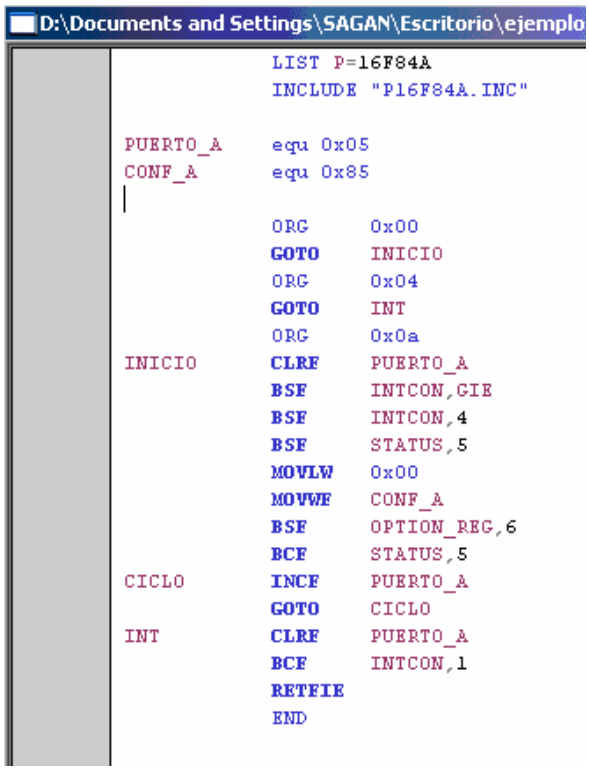
El uso de mayúsculas y minúsculas obedece a una serie de reglas de estilo, que aunque no son obligatorias, facilitan la lectura del código.

Directivas del compilador en mayúsculas
Nombres de variables en minúsculas
Definición de registros en mayúsculas
Mnemónicos en mayúsculas

Todo lo que vaya después del símbolo **punto y coma** es considerado un comentario.

3.2.3 CODIGO EJEMPLO

El código ejemplo es un contador binario, la salida es por el puerto A (de 5-bits) y reinicia su cuenta al desbordarse o con una interrupción externa (PB0, pin6 del chip).



LIST P=16F84A
INCLUYE “P16F84A.INC”

La primera línea es una directiva que le indica al ensamblador el modelo de PIC con que debe trabajar. La segunda línea es una directiva que “llama” a la biblioteca que contiene la definición de los registros y bits más importantes, para que en lugar de escribir su posición en

memoria escribamos su nombre. Por ejemplo, para referirnos al registro de estado, podemos escribir su localidad (0x03) o su nombre (STATUS). Para observar todas las definiciones podemos buscar en la ruta de instalación del entorno **MPLAB** (la ruta por defecto es C:\Archivos de programa\MPLAB IDE\MCHIP_Tools) el archivo **.inc** que llamamos a través de la directiva **INCLUDE**; para abrirlo utilizamos cualquier editor de textos como el block de notas o wordpad.

A la derecha de ésta página, podemos observar una tabla con los nombres y direcciones de registros de propósito especifico para el PIC16F84.

PUERTO_A EQU 0x05
CONF_A EQU 0x85

Con estas dos líneas se está haciendo una definición de registros; para el ensamblador tendrán una mayor prioridad con respecto a los que son llamados con la directiva INCLUDE.

ORG 0x00
GOTO INICIO

La directiva **ORG** *localidad de memoria de programa*, indica al ensamblador donde debe situar el siguiente fragmento de código. Para el ejemplo, la instrucción **GOTO** se escribe en la localidad **0x00** de la memoria de programa. El salto incondicional de la instrucción **GOTO**

se dirige hacia la instrucción donde se encuentra la etiqueta **INICIO**.

```
ORG 0x04
GOTO INT
```

El programa del ejemplo utiliza interrupciones externas; cada vez que se produzca una interrupción, el contador de programa apuntará hacia el vector de interrupciones (en los PIC es la localidad **0x04**). Es indispensable escribir a partir de esta localidad la rutina de servicio de interrupción ó un salto hacia la dirección donde se ubica la rutina de interrupción; está segunda opción es la empleada en el ejemplo.

```
ORG 0xa
INICIO CLRF        PUERTO_A
```

A partir de la localidad **0x0a** se escribirá el código subsiguiente. **INICIO** es una etiqueta que usa el ensamblador para indicar el destino de un salto (no genera código ejecutable). Con la instrucción **CLRF PUERTO_A** estamos poniendo en ceros el contenido del registro asociado a **PUERTO_A**.

```
BSF    INTCON,GIE
BSF    INTCON,4
```

Con la primera instrucción habilitamos las interrupciones globales, esto es, poniendo a uno el bit 7 del registro **INTCON**; en lugar de poner la posición del bit (7) pusimos el nombre del bit, lo cual es válido, tal como lo demuestra la siguiente instrucción, que sirve para habilitar las interrupciones externas con el pin **RBO/INT**.

```
BSF    STATUS,5
```

Las configuraciones realizadas hasta el momento han sido en registros mapeados en el **Banco 0** de la memoria de datos; como también necesitamos configurar algunos registros mapeados en el **Banco 1** debemos cambiarnos a dicho bloque de memoria, esto es posible poniendo a uno el bit 5 del registro **STATUS**.

MOLW 0x00
MOVWF CONF_A

La primera instrucción carga al registro W con **ceros**. La segunda instrucción copia el contenido de W en el registro **CONF_A** (originalmente este registro era llamado **TRISA**, pero nosotros lo redefinimos), este registro especifica si los pines del puerto son entradas o salidas; con todos en **cero** el puerto sólo será de salida, con **unos** el puerto es de entrada; también se pueden hacer combinaciones como lo muestra la tabla:

TRISA	ASIGNACIÓN	ESTADO
RA0	1	Entrada
RA1	0	Salida
RA2	1	Entrada
RA3	0	Salida
RA4	1	Entrada

BSF OPTION_REG,6

Ponemos a uno el bit **INTEDG** del registro **OPTION** para que las interrupciones externas se activen con el flanco de subida.

BCF STATUS,5

Los próximos registros en memoria que vamos a utilizar están en el **Banco 0**; nos pasamos a él, poniendo a **cero** el bit 5 del registro **STATUS**.

CICLO INCF PUERTO_A

Tenemos una etiqueta que nos ayudará para crear un ciclo. Con la instrucción **INCF PUERTO_A**. estamos incrementando el contenido del registro que hemos nombrado **PUERTO_A**, el cual originalmente se llamaba **PORTA**. Si el dispositivo está configurado como salida, en los pines del puerto se muestra el contenido del registro **PUERTO_A**.

GOTO CICLO

Lo dicho, tenemos un brinco incondicional hacia la etiqueta CICLO. Hay varias formas de romper el ciclo: quitando la polarización, reseteando al circuito, pero la que nos interesa es generando una interrupción.

INT CLRF PUERTO_A

Cuando se genera una interrupción, el vector de interrupciones externas direcciona al contador de programa hacia la localidad 0x04 de la memoria de programa y de ahí hacia la etiqueta **INT**, lugar donde reside nuestra rutina de servicio de interrupción. La primera acción que se lleva a cabo es poner a **ceros** al registra **PUERTO_A**, reiniciando así, la cuenta.

BCF INTCON,1

Una vez concluida nuestra rutina de interrupción limpiamos la bandera de interrupción ocurrida, poniendo a **cero** el bit 1 del registro **INTCON**.

RETFIE

Con está instrucción terminamos la interrupción; el contador de programa regresa a la instrucción siguiente que tenia antes de que ocurriera la interrupción.

END

La instrucción **END** no genera código, sirve para indicarle al ensamblador que el programa ha finalizado.

3.3 GUIA RAPIDA PARA EL USO DE PROTEUS

3.3.1 INTRODUCCION

El software de diseño y simulación Proteus VSM es una herramienta útil para estudiantes y profesionales que desean acelerar y mejorar sus habilidades para del desarrollo de aplicaciones analógicas y digitales.

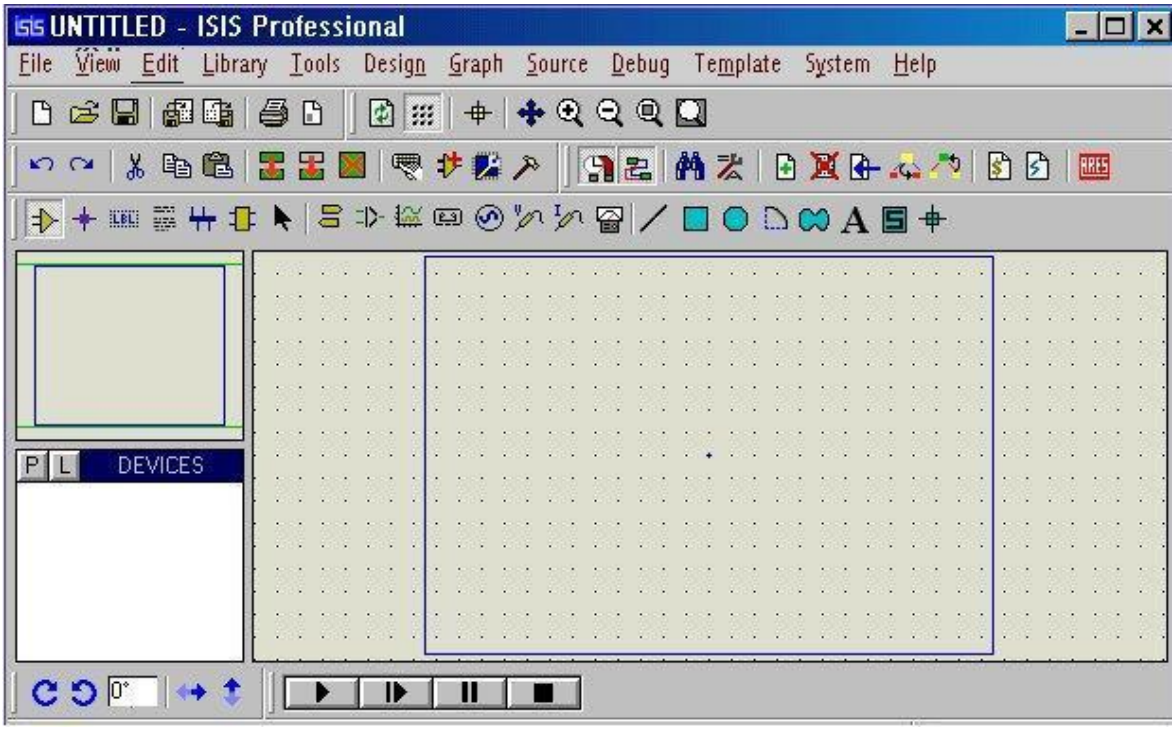
Este permite el diseño de circuitos empleando un entorno gráfico en el cual es posible colocar los símbolos representativos de los componentes y realizar la simulación de su funcionamiento sin el riesgo de ocasionar daños a los circuitos.

La simulación puede incluir instrumentos de medición y la inclusión de gráficas que representan las señales obtenidas en la simulación.

Lo que más interés ha despertado es la capacidad de simular adecuadamente el funcionamiento de los microcontroladores más populares (PICS, ATMEL-AVR, MOTOROLA, 8051, etc

3.3.2 Procedimiento de Arranque del programa:

1.- Inicio -> Programas -> Proteus 6 Professional -> ISIS 6 Professional.

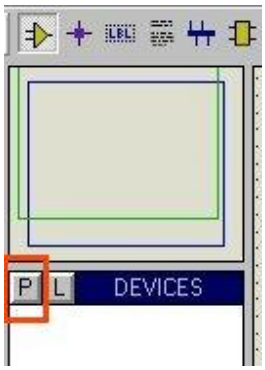


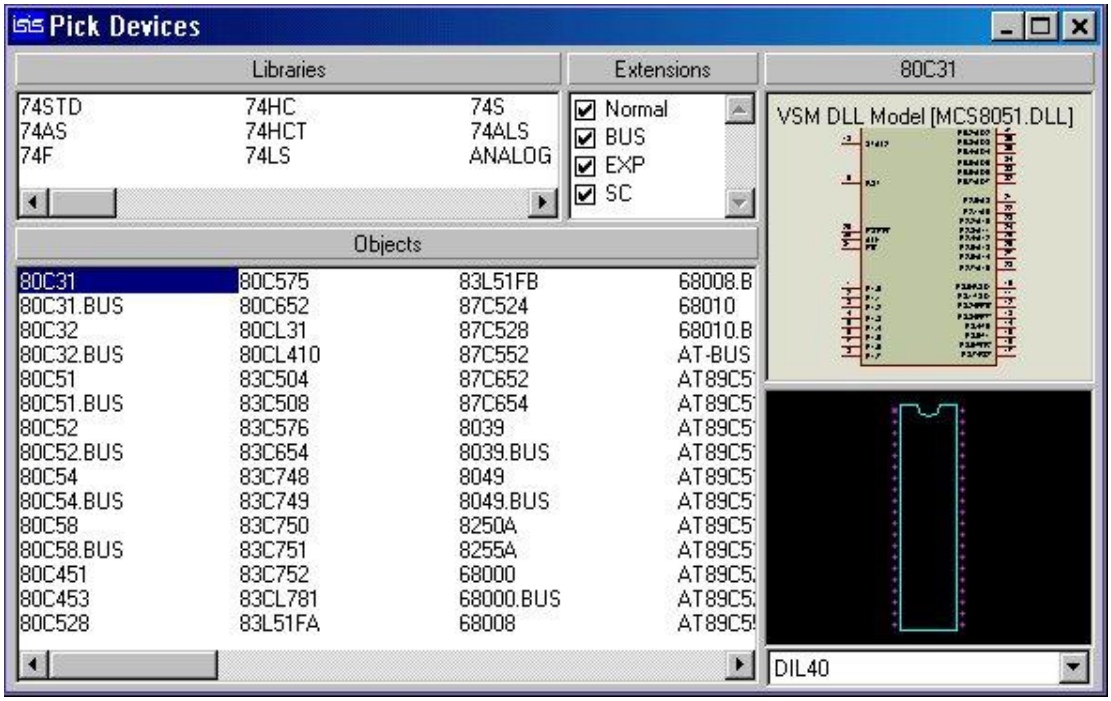
2.- La forma corta es dar doble click en el icono del programa ubicado en el escritorio.

3.3.3 Desarrollo de Circuito Básico # 1

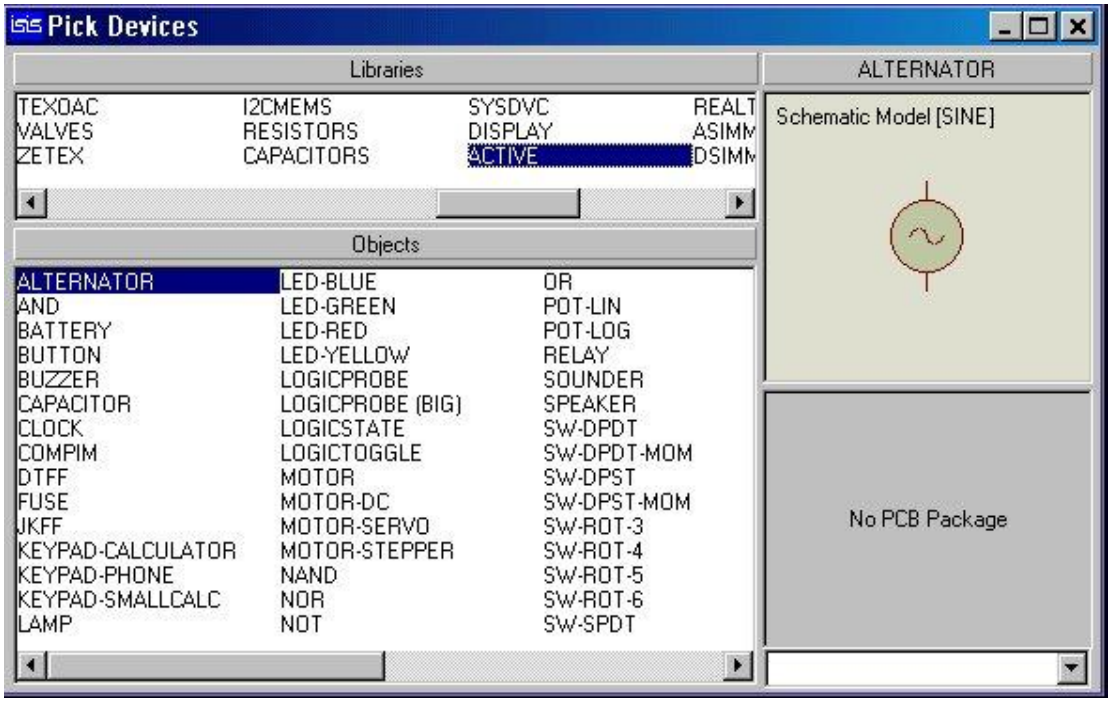
Alimentación de un foco de corriente alterna.

1.- Dar un click en el botón **Pick Devices** localizado en la parte izquierda de la pantalla debajo de la pantalla de exploración del diagrama para abrir la forma del mismo nombre.

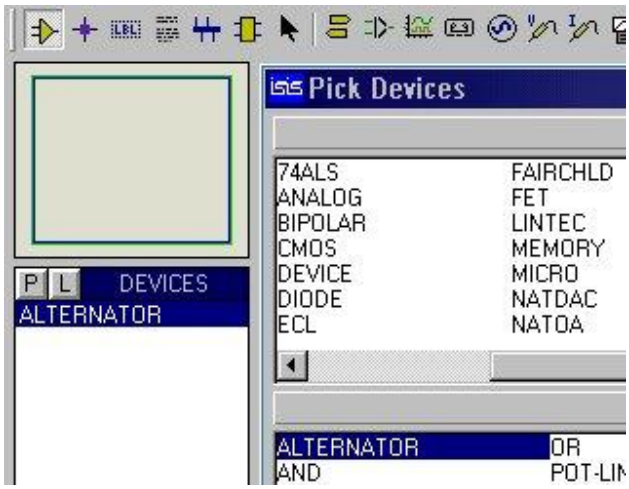




2.- En la ventana **Libraries** (Parte superior izquierda) buscar la librería **ACTIVE**, y dar un click sobre ella.



3.- En la ventana **Objects** elegir el componente **ALTERNATOR** dando doble click sobre el nombre.

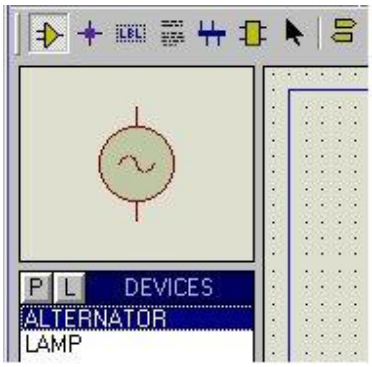


Se puede observar que en la ventana **DEVICES** aparece el nombre del componente elegido. Si es el único componente que se va a elegir se puede cerrar la forma Pick Devices, pero si es necesario más de uno, se puede continuar eligiendo los componentes necesarios para nuestro diseño.

4.- En la misma librería **ACTIVE** dar doble click sobre el componente **LAMP**.

5.- Cerrar la Forma **Pick Devices** en el botón estándar. (La cruz en la esquina superior derecha)

6.- Dar un click en la palabra **ALTERNATOR** de la ventana **DEVICES** y observar que aparece el componente en la pantalla de exploración del circuito.



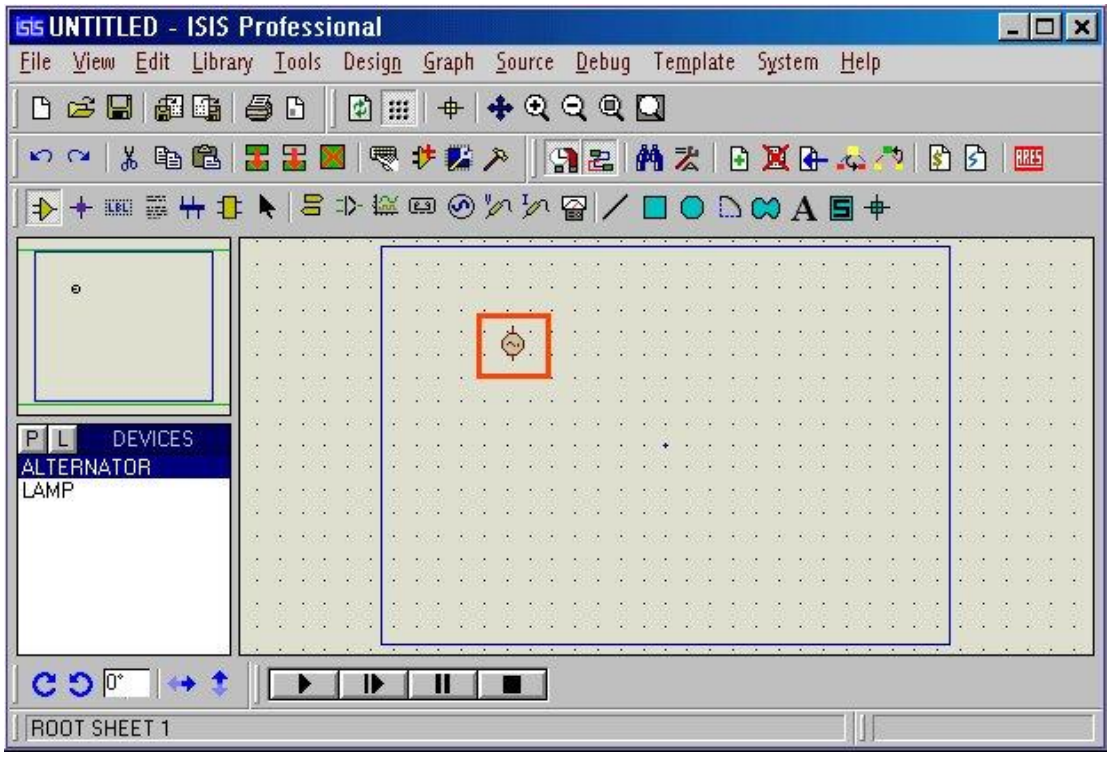
7.- Explorar las funciones de orientación del componente, parte inferior izquierda de la pantalla.

8.- Comenzando por la izquierda presionar cada uno de los botones de orientación.

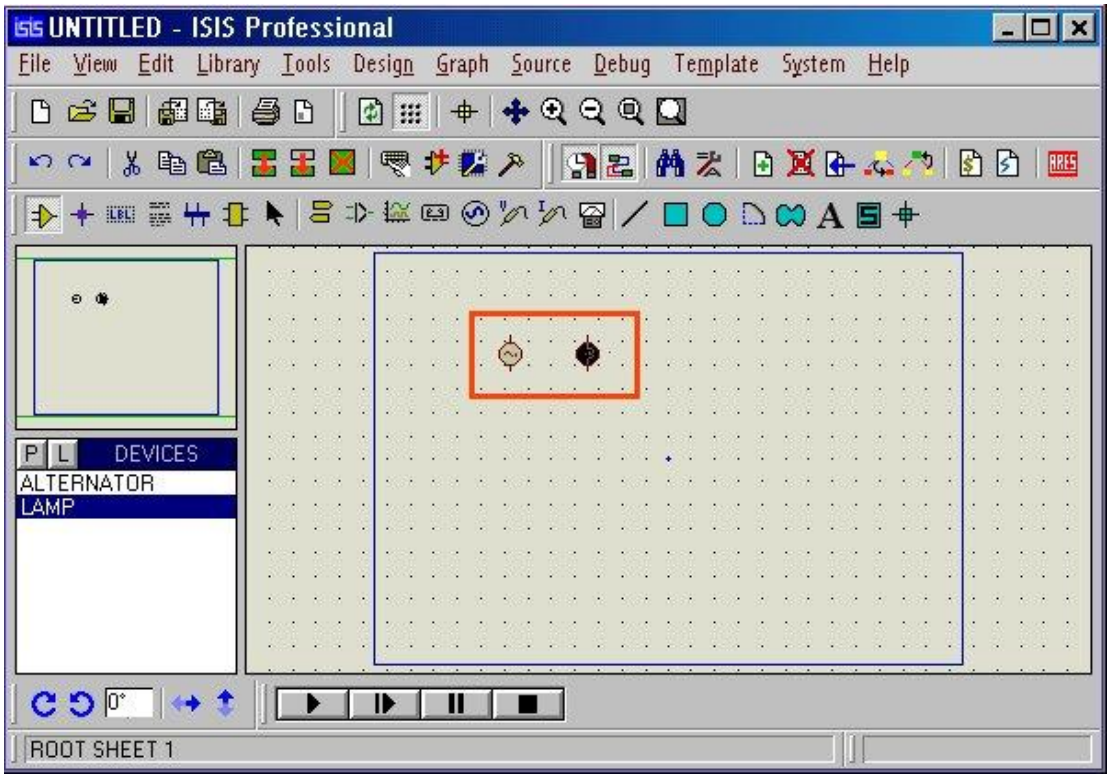
9.- En el cuadro de texto se puede introducir un ángulo pero sólo acepta valores de (0º, +/-90º, +/-180º, +/-270º), por lo que es mejor manejar la orientación por medio de los botones. Este mismo cuadro de texto muestra el ángulo actual obtenido al presionar los botones.

10.- Dejar el componente en la posición inicial.

11.- Con el componente seleccionado dar un click en el área de trabajo, con lo que se logra colocar el componente en el área de trabajo.



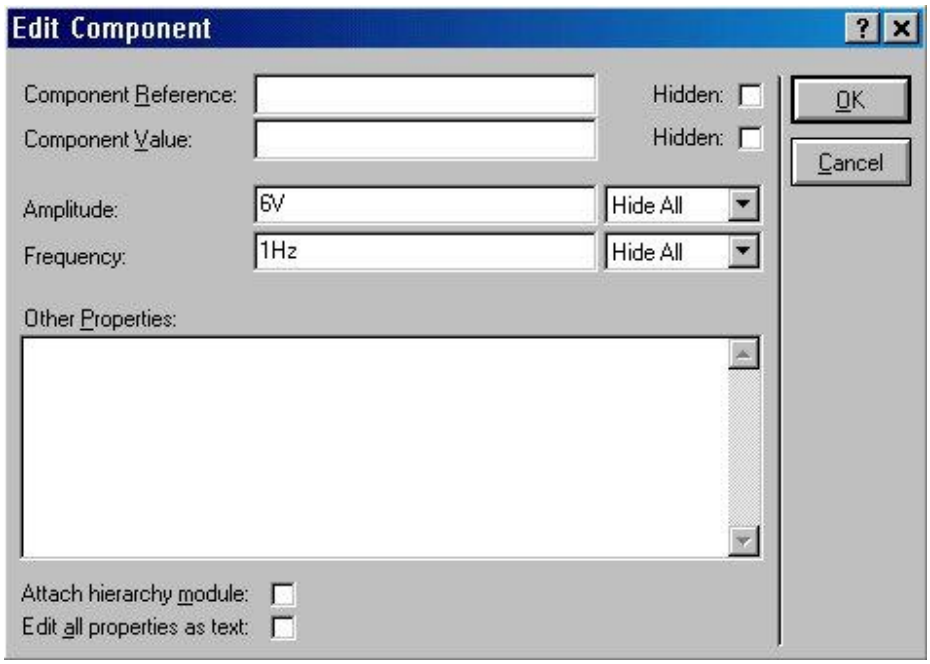
12.- Repetir el procedimiento anterior con el componente LAMP.



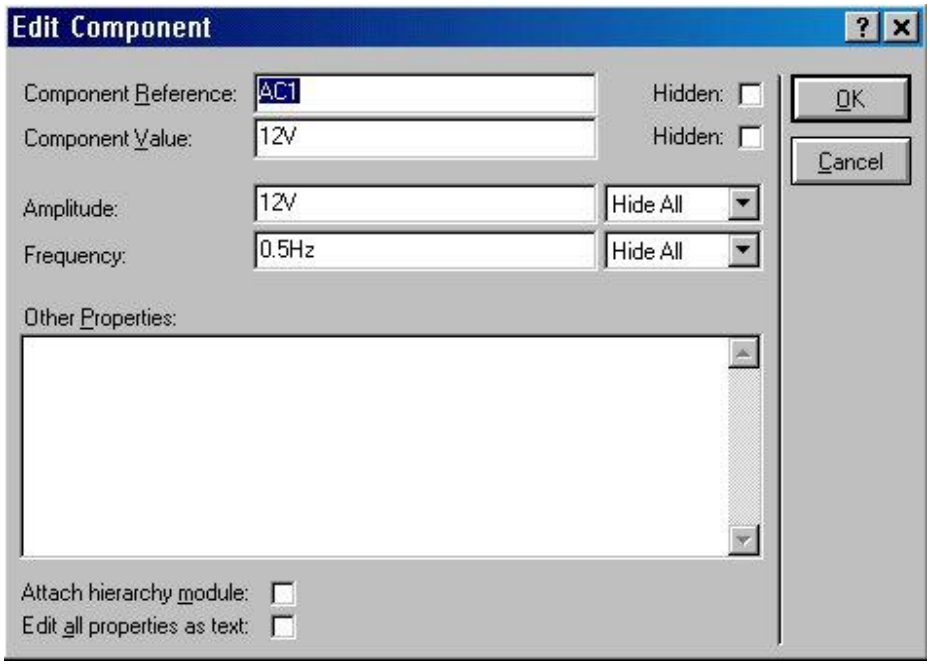
13.- Configurar los componentes de la siguiente manera.

a.- Dar un click con el botón derecho sobre el componente **ALTERNATOR**. Notar que su contorno cambia a rojo.

b.- Dar un click ahora con el botón izquierdo para abrir la forma **Edit Component**.

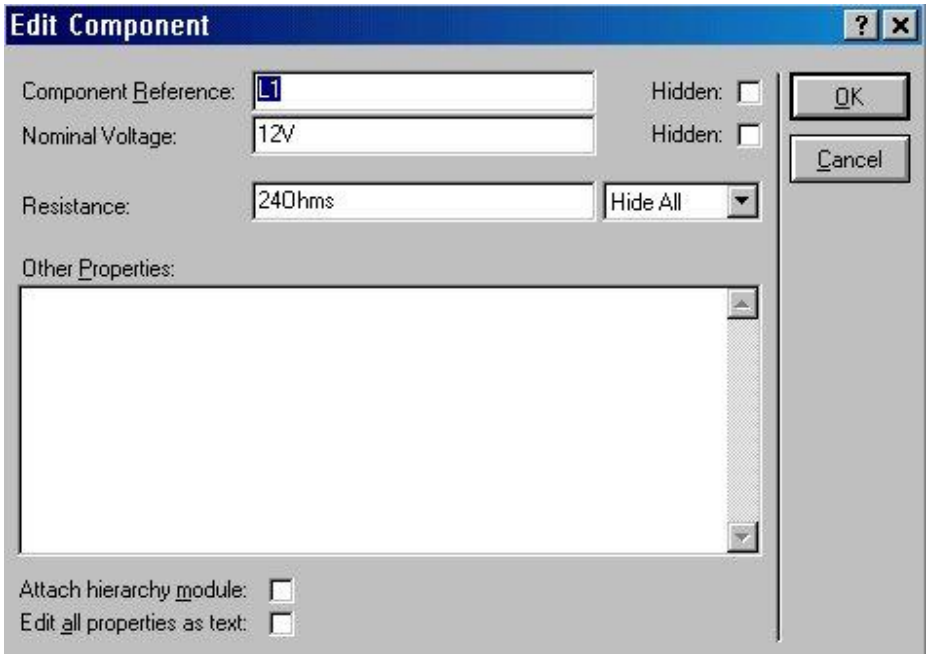


c.- Dar un nombre al componente en el campo Component Reference (AC1), Poner el valor del componente en el Component Value (12V), Modificar el valor de la amplitud a (12V) y la frecuencia a 0.5Hz.



d.- Presionar el botón OK.

e.- Verificar los valores del componente **LAMP** y si el valor del voltaje corresponde con el del **ALTERNATOR**, no es necesario realizar ninguna modificación. Presionar OK.



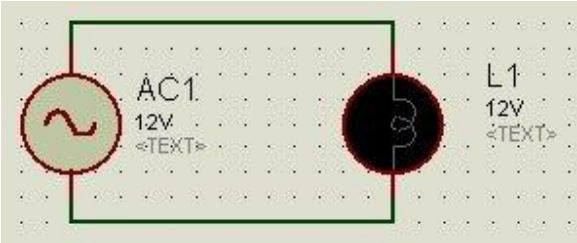
14.- Realizar la conexión de los componentes de la siguiente forma:
a.- Colocar el puntero del mouse en el extremo superior del ALTERNATOR. Aparece una cruz en el extremo de la flecha.

b.- Dar un click para habilitar la conexión por medio de cable.

c.- Desplazar el mouse (desaparece la cruz) hasta el extremo superior del componente LAMP y lograr que vuelva a aparecer la cruz en el extremo de la flecha.

d.- Dar otro click para realizar la conexión.

e.- Repetir los pasos anteriores para la pare inferior de los componentes.
Resultado:



Este es el procedimiento estándar para conectar cualquier componente con el que se trabaje en el programa.

15.- Probar el funcionamiento del circuito presionando el botón play que se encuentra en la parte inferior de la pantalla.

16.- Para acercar el circuito y poder observar mejor la simulación se puede recurrir a los controles de zoom.

Comenzando de izquierda a derecha tenemos:

a.- Re-centrar la pantalla.

b.- Incrementar el acercamiento.

c.- Decrementar el acercamiento.

d.- Ver la hoja completa.

e.- Ver una área seleccionada

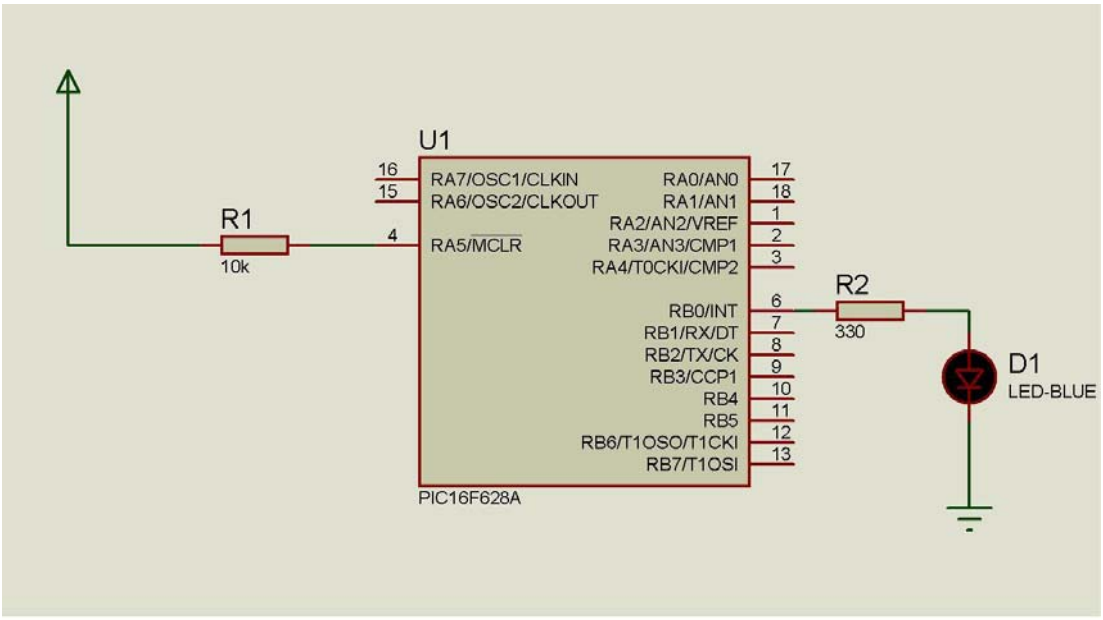
Usar la herramienta para Ver una área seleccionada dando un clic

Usando el cursor modificado presionar el botón izquierdo en el extremo superior izquierdo del circuito armado y sin soltar el botón formar un rectángulo que contenga todo el circuito, por último soltar el botón.

Este procedimiento se puede usar para acercar partes de un circuito de mayor tamaño.

3.3.4 Simulación de un microcontrolador (PIC´s):

I.- Crear el siguiente circuito.



DEVICE (RES)
MICRO (PIC16F628)
ACTIVE (LED-RED)

II.- Guardar el circuito en una carpeta con el nombre Blink y nombrar al archivo Blink.

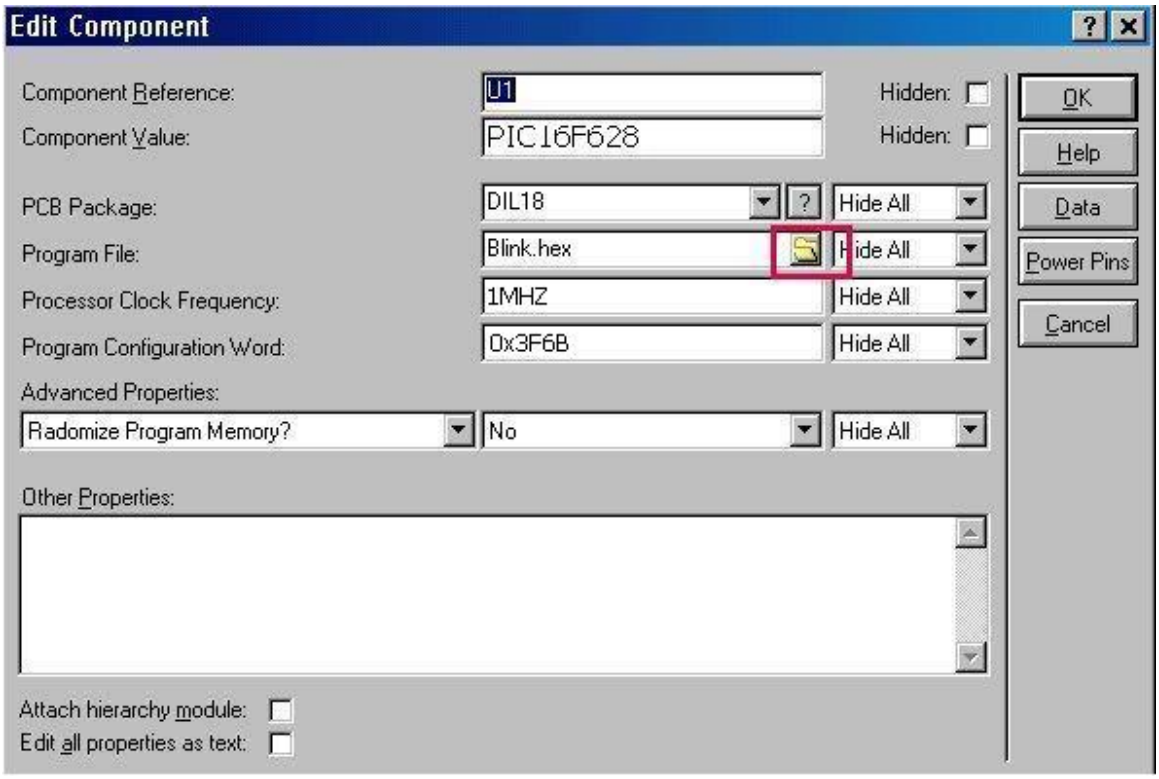
III.- Copiar el siguiente texto al NOTEPAD de Windows y guardarlo en un archivo de texto con el nombre Blink.bas dentro de la carpeta Blink.

```
list p=16f628a
include p16f628a.inc
```

```
        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
_INTOSC_OSC_NOCLKOUT & _MCLRE_OFF & _LVP_OFF

ContadorA     EQU   0x20
ContadorB     EQU   0x21
ContadorC     EQU   0x22

;configuracion de puertos

        CLRF PORTA              ;Inicializando el puerto A
        CLRF PORTB          ;Inicializando el puerto B
        MOVLW 0x07          ;Apagando comparadores
        MOVWF CMCON
        BCF STATUS, RP1
        BSF STATUS, RP0 ;Seleccionar Banco de memoria 1
        MOVLW 0xFF              ;direccion de datos puerto A
        MOVWF TRISA
        MOVLW 0xFE              ;direccion de datos puerto B
        MOVWF TRISB
        BCF STATUS, RP0    ; volver al banco de memoria 0



;programa

CicloPrincipal

        BSF PORTB,0
        CALL Retardo
        BCF PORTB,0
        CALL Retardo

        goto CicloPrincipal

Retardo
```

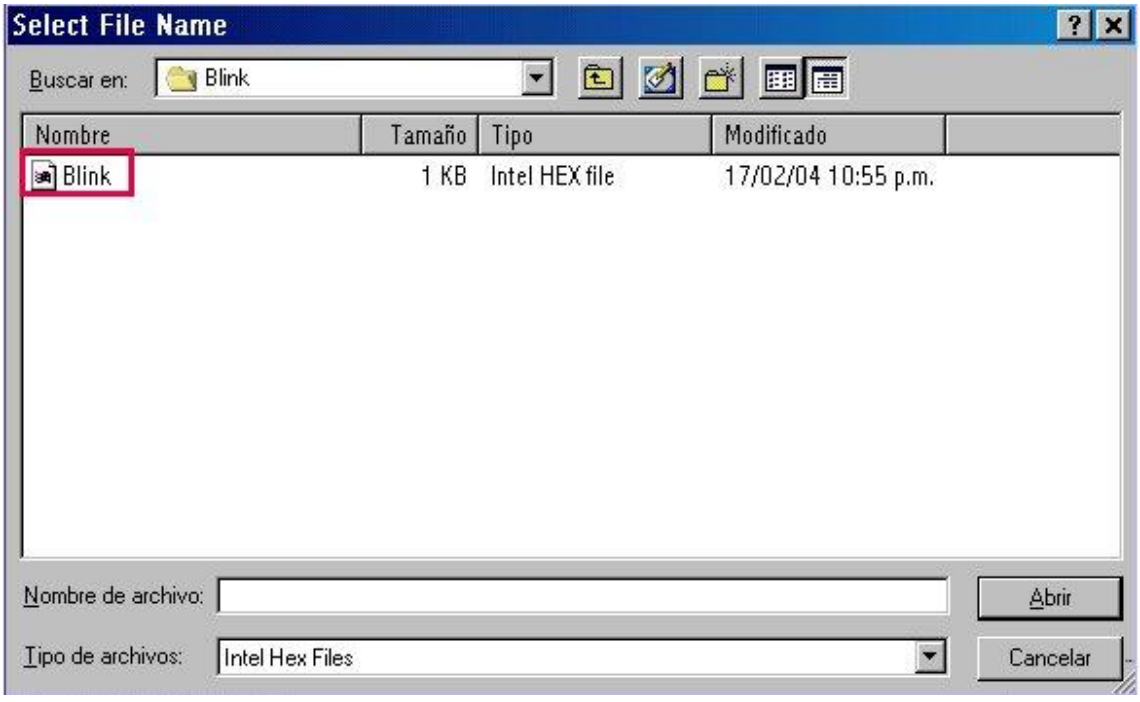
```
;PIC Time Delay = 1,0000020 s with Osc = 4 MHz

        MOVLW   D'6'
        MOVWF   ContadorC
        MOVLW   D'24'
        MOVWF   ContadorB
        MOVLW   D'168'
        MOVWF   ContadorA
CicloRetardo  DECFSZ    ContadorA,1
        GOTO CicloRetardo
        DECFSZ  ContadorB,1
        GOTO CicloRetardo
        DECFSZ  ContadorC,1
        GOTO CicloRetardo
        RETURN

        end
```

IV.- convertir el archivo en blink.hex posteriormente explicado con ayuda del mplab.



V.- Dar un click con el botón izquierdo en el icono que muestra una carpeta abierta (El que marca el rectángulo rojo).

VI.- Seleccionar el archivo Blink.hex.



VII.- Ajustar la frecuencia del reloj a 4MHz en el campo Processor Clock Frecuency.

VIII.- Presionar OK.

IX.- Ejecutar la simulación.

CONCLUSIONES

- De acuerdo a lo que se pudo observa, el microcontrolador 16f628 pudo realizar perfectamente las prácticas propuestas del laboratorio de microprocesadores, sin necesidad de emplear algun otro microcontrolador para poder gobernar dispositivos de salida como por ejemplo LCD, Motores a paso o Teclados Matriciales.

- Existen una gran variedad de microcontroladores en el mercado y no se puede decir cual es el mejor o el peor, Simplemente se tiene que buscar el mas adecuado para el proyecto que se este realizando.

- Se escogió el microcontrolador 16f628 porque tiene un precio accesible en nuestro mercado local, en su estructura interna tiene un oscilador integrado y es también un microcontrolador de la gama media de la familia de microchip que consta de 33 instrucciones en la memoria del programa y esto facilita nuestra programación.

- El microcontrolador mas famoso de microchip es el 16f84 de la gama media, pero ya es un microcontrolador obsoleto ya que cuenta con solo dos bancos de memoria y actualmente microchip los hace con cuatro bancos que ayudan entre otras cosas facilitar la programación y a reconocer a otros microcontroladores para diseños mas complejos.

- La empresa microchip que sus microcontroladores son conocidos como PIC´s, es la empresa que mas vende en el mundo microcontroladores, sus mas cercanos competidores en la venta a nivel mundial son las AVRS y lo microcontroladores de motorota

Bibliografía

Microcontroladores PIC – Aprenda una profesión.
Ing. HoracioD. Vallejo
Quark S.R.l
2002

MICROCONTROLADORES PIC – Diseño practico de aplicaciones
Jose M. Angulo Usategui
Ignacio Angulo Martinez
Mc Graw Hill (200#)

Circuitos electrónicos y sus aplicaciones.
Grob Bernard
Ed. McGraw Hill.
1986 M

Electrónica fundamental para científicos.
Brophy, James J.
Ed. Reverté, S. A.
1974

PIC16C77X, PIC16F7X, PIC16F8X, PIC16F87XA, PIC18FXX8
Data Sheets
Microchip Technology Inc.

PICmicro™ Mid-Range MCU Family
Reference Manual

Desarrollo de un caso practico
“programadores cpld´s y microcontroladores”
Ismael Díaz Rangel

ENLACES A PÁGINAS DE INTERÉS

Agilent Technologies	www.agilent.com
Microchip Technology Inc.	www.microchip.com
Atmel Corp.	www.atmel.com
FES Aragón	http://informatica.aragon.unam.mx/
Maxim integrated Products	www.maxim-ic.com
UNAM	www.unam.mx

APENDICE A

1. PLAN DE CALIDAD.

ANEXO CUATRO: Estructura de formato para elaborar otro tipo de documentos

Estos se desarrollarán en función de las necesidades específicas de la información a documentar, estos pueden ser catálogos, especificaciones, organigramas, perfiles de puestos, etc, y debido a que en la mayoría de las ocasiones ya están definidos por la organización, sólo se definirán los contenidos mínimos que deberán cumplir estos documentos.

1.- Debe desarrollarse dentro del formato F1, que se muestra en el Anexo CINCO.

2.- El Documento debe estar contenido en una carpeta identificada como Planes, programas e instructivos o el nombre correspondiente al tipo de documento, la cual funcionará como documento original, las copias podrán ser distribuidas en engargolados, carpetas , según convenga los intereses del laboratorio e identificadas con la leyenda COPIA CONTROLADA y su respectivo número de copia, ver anexo OCHO.
La distribución en la red, no tiene asignado un número de copia, por lo que llevará la leyenda COPIA NO CONTROLADA

3.- La carpeta debe contener un Índice General de los documentos que incluye, este índice tendrá un formato semejante al F1, solo que no llevará revisión, paginación ya que solo funcionará como portada.

4.- Además de esta portada, cada documento debe tener una portada desarrollada en el Formato Universal F1, con el índice o contenido del mismo, ésta será la página principal y contendrá por lo tanto las firmas de, elaboración, revisión y autorización.

5- El desarrollo de estos documentos utilizará una metodología libre, ya que como se explicó al inicio de este anexo, algunas de las actividades o contenidos de los mismos ya vienen definidos por la organización o el fabricante.

6- La persona que elabore un documento debe estar autorizada por el Jefe del Laboratorio.

7- La fecha de emisión del procedimiento debe ser igual o posterior a la fecha correspondiente de revisión del historial de cambios.

8- Todos los documentos serán revisados por el jefe del laboratorio o la persona designada por él, cuando menos una vez al año para asegurar continuamente la congruencia entre lo que se dice y lo que se hace dentro del laboratorio. Si en la revisión anual no hay ningún cambio, solo se actualiza la portada del documento y el historial de cambios, estas serán sustituidas también en las correspondientes copias bajo resguardo de otras áreas o laboratorios.

9.- Destruir todas las copias obsoletas (si es que existen) y sellar con la leyenda de documento obsoleto, las versiones anteriores si se conserva alguna como referencia histórica.

ANEXO CINCO: Formato Universal para todo tipo de documento (Formato F1)

<table>
<tr><td colspan="6">UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</td></tr>
<tr><td></td><td colspan="5">FACULTAD DE ESTUDIOS SUPERIORES
“ARAGÓN”</td></tr>
<tr><td></td><td colspan="5">TÍTULO DEL DOCUMENTO:</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="6">DIVISIÓN DE CIENCIAS FÍSICO-MATEMÁTICAS Y DE LAS INGENIERÍAS</td></tr>
<tr><td>Nombre y Firma elaboró</td><td>Nombre y Firma elaboró</td><td>Nombre y Firma elaboró</td><td>Nombre y Firma revisó</td><td colspan="2">Fecha de emisión

Rev #</td></tr>
<tr><td colspan="6">ÉSTA INFORMACIÓN ES DE USO EXCLUSIVO DE LOS LABORATORIOS DE LA DIVISIÓN DE CIENCIAS FISICO-MATEMÁTICAS Y DE LAS INGENIERIAS</td></tr>
</table>

El Formato F1 debe contener los siguientes datos:

En la parte superior:

1. Logotipo y Nombre de la institución
2. Título del documento

Al calce, esta información sólo debe aparecer en la primer hoja del documento.

1. Nombre y firma de quien(es) elabora(n)
2. Nombre y firma de quien revisa
3. Fecha de Emisión
4. Revisión
5. Una leyenda indicando : La información contenida en el documento es de uso exclusivo del laboratorio.

La paginación estará ubicada en parte inferior derecha de los documentos.

[1] PE1: Proceso Externo 1 "Adquisiciones, almacén e inventarios"

2. PLAN DE ESTUDIOS.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ENEP – ARAGON

Programa de Asignatura

Carrera:	INGENIERO MECANICO – ELECTRICISTA
Area de Conocimiento:	ELECTRONICA

DIVISION DE ESTUDIOS PROFESIONALES

Fecha de aprobación del:
* Consejo Técnico de la ENEP – Aragón JUNIO 28 DE 1991
* Consejo Universitario

Programa de la asignatura:	**MICROPROCESADORES**

Clave:	0586		Núm. De créditos:	10
Duración del curso:	Semanas:	16	Asignatura:	Obligatoria
	Horas:	96	Semestre:	Octavo
	Teoría:	4.0		
	Práctica:	2.0		

OBJETIVO DEL CURSO

Comprender la estructura y funcionamiento de los microprocesadores para aplicarlos en la solución de problemas de ingenieria.

TEMAS

Núm.	Nombre	Horas	
		Teoría	Práctica
I	INTRODUCCION A LOS MICROPROCESADORES Y MICROCOMPUTADORES	6.0	2.0
II	ESTRUCTURAS DE UN MICROCOMPUTADOR	10.0	6.0
III	ESTRUCTURA DE LOS ELEMENTOS DE MEMORIA	8.0	4.0
IV	INTRODUCCION A LA PROGRAMACION DE MICROPROCESADORES	6.0	4.0
V	PROGRAMACION DEL MICROPROCESADOR	10.0	6.0
VI	INTERFASE DE MICROCOMPUTADORAS CON DISPOSITIVOS EXTERNOS	6.0	2.0
VII	COMUNICACION SERIE ASINCRONA Y SINCRONA	8.0	4.0
VIII	COMUNICACION PARALELO	6.0	2.0
IX	APLICACIONES DE LOS MICROPROCESADORES	4.0	2.0
		64.0	32.0
HORAS TOTALES:		96.0	

Asignatura: MICROPROCESADORES

ANTECEDENTES, OBJETIVOS Y CONTENIDO DE LOS TEMAS

TEMA I INTRODUCCION A LOS MICROPROCESADORES Y RMICROCOMPUTADORES"

Antecedentes: Diseño Lógico

Objetivo: Dar a conocer los elementos constitutivos de un microcomputador.

Contenido:

I.1 Elementos de un microcomputador.

- El microprocesador, las memorias y las interfases.
- Comunicación entre los módulos.

TEMA II	"ESTRUCTURA DE UN MICROCOMPUTADOR"
Antecedentes:	Diseño Lógico.
Objetivo:	Conocer el funcionamiento lógico y físico de un microcomputador.
Contenido:	
II.1	Las señales.
II.2	Los elementos internos.
II.3	El funcionamiento interno: los diagramas de estado.
II.4	Estudio detallado de un microprocesador de 8 bits.

TEMA III	"ESTRUCTURA DE LOS ELEMENTOS DE MEMORIA"
Antecedentes:	Contenidos en la misma.
Objetivo:	Conocer el manejo y la tecnología de las memorias empleadas en los sistemas microcomputarizados.
Contenido:	
III.1	Estado actual de la tecnología.
III.2	Estructura de los sistemas de memoria.
III.3	Estudio detallado de varios casos (RAM, ROM)

TEMA IV	"INTRODUCCION A LA PROGRAMACION DE LOS MICROPROCESADORES"
Antecedentes:	Contenidos en la misma.
Objetivo:	Familiarizarse con la programación de los microprocesadores.
Contenido:	
IV.1	Lenguaje de Máquinas y Ensambladores.
IV.2	Monitor.
IV.3	Registro, Memoria, Buses.

TEMA V	"PROGRAMACION DEL MICROPROCESADOR"
Antecedentes:	Contenidos en la misma.
Objetivo:	Programar el microprocesador usando su conjunto de instrucciones para el desarrollo de programas de aplicación.
Contenido:	
V.1	Direccionamiento de Memorias y E/S.
V.2	Banderas y Brincos.
V.3	Conteo y Lazos de tiempo.
V.4	Funciones Aritméticas.
V.5	Conceptos Avanzados.
V.6	Interrupciones.

TEMA VI	"INTERFASE DE MICROCOMPUTADORAS CON DISPOSITIVOS EXTERNOS"
Antecedentes:	Contenidos en la misma.

Objetivo:	Conocer la conexión de la microcomputadora con otros dispositivos.

Contenido:

VI.1	Uso de líneas programadas Entrada/Salida para el control de dispositivos.
VI.2	Control de dispositivos con técnicas de Programación.
VI.3	Interfase RS-232 y lazo de corriente 20 MA.

TEMA VII	"COMUNICACION SERIE ASINCRONA Y SINCRONA"
Antecedentes:	Contenidos en la misma.
Objetivo:	Conocer la comunicación de datos en serie entre la microcomputadora y los dispositivos externos.

Contenido:

VII.1	Estructura de la comunicación Asíncrona.
VII.2	Interfases asíncronas, estudio del UART y similares.
VII.3	Estructura de la comunicación serie síncrona.
VII.4	Interfaces síncronas: los USART y similares.
VII.5	Comunicación síncrona compleja: Protocolos.

TEMA VIII	"COMUNICACION PARALELO"
Antecedentes:	Contenidos en la misma.
Objetivo:	Conocer la comunicación de datos en parelelo entre la microcomputadora y los dispositivos externos.

Contenido:

VIII.1	Estructura de la comunicación paralelo.
VIII.2	El saludo (handshake): tipos.
VIII.3	Interfases integrados: PIA, PIO, PPI y similares.
VIII.4	Interfases universales programables: VPI y similares.

TEMA IX	"APLICACIONES DE LOS MICROPROCESADORES"
Antecedentes:	Contenidos en la misma.
Objetivo:	Integrar los conocimientos adquiridos en el campo para poder aplicar los microprcesadores en el campo de la Ingeniería.

Contenido:

IX.1	Aplicación de los microprocesadores (8088, 8036, 80486, 68701, DSP566000) en instrumentación y control.

TECNICAS DE ENSEÑANZA		**ELEMENTOS DE EVALUACION**	
Exposición oral	(X)	Exámenes parciales	(X)
Exposición audiovisual	( )	Exámenes finales	(X)
Ejercicios dentro de clase	(X)	Trabajos y tareas fuera del aula	(X)
Ejercicios fuera del aula	(X)	Participación en clase	(X)
Seminarios	( )	Asistencia a prácticas	(X)
Lecturas obligatorias	( )	Otros:	( )
Trabajos de investigación	(X)		
Prácticas de taller o	(X)		

laboratorio
Prácticas de campo ( )
Otras:

ANTECEDENTES

Asignatura	Clave	Temas que se requieren
Diseño Lógico		TODOS

CONSECUENTES

Asignatura	Clave
Diseño de Sistemas con Microporcesadores	

BIBLIOGRAFIA

Texto	Temas de la materia para las que se recomienda:
Steve Ciarcia ; Construya una Microcomputadora ; Basado en el Z-80 ; Byte Books/Mc Graw Hill, 1985 350 págs.	I, II, III, IV y VI
James W. Gault/Russell L. Pimmel ; Sistemas Digitales Basados en un Microprocesador ; Mc Graw Hill de México, 1985, 461 págs.	I, II, III, IV, V y IX
H. Lilen ; Del Microprocesador al Microcomputador ; Ed. Marcombo Boixareu Editores, 1980 ; 461 págs.	I, II, III, IV, V, VI y IX
Serie Mundo Electrónico ; Microprocesadores y Microcomputadores ; Ed. Marcombo Boixareu Editores ; 2a, Edición, 1978, 182 págs.	I, II, III, VI y IX
Adam Osborne ; Introducción a las Microcomputadoras, Vol. I ; Osborne Mc Graw Hill ; 2a. Edición, 452 págs.	I, II, III, IV, V y VII
Hohewtein, Louis ; Computer Periphera for Minicomputers ; Microprocessors, and Personal Computers Mc Graw Hill, Book Xompany, 1980, 312 págs.	VI

Serie Mundo Electrónico José Mompín Poblet ; Interconexión de Periféricos a Microprocesadores ; Publicaciones Marcombo, S. A., 1984, 228 págas.	VI, VII y VIII
Kenneth L. Short ; Microprocesadores y Lógica Programada ; Colección Electrónica/Informática ; Editorial Gustavo Gili, S. A. 1981 523 págs.	I, II, III, IV, V, VI, VII y IX
Peatman, John ; Microcomputer base design ; Mc Graw Hill, 1977	TODOS
Osborne & Associates Inc. ; Z-80 Programing for Logic Design ; Some Real Products.	TODOS

Asignatura: MICROPROCESADORES

SUGERENCIAS PARA IMPARTIR LA ASIGNATURA

I. INTRODUCCION A LOS MICROPROCESADORES Y MICROCOMPUTADORAS

Se dará a conocer la diferencia entre un microprocesador y una microcomputadora, así mismo se hará un bosquejo histórico del desarrollo de los microprocesadores y microcomputadoras monolíticas, haciendo notar las diferencias existentes entre éstas.

II. ESTRUCTURA DE UN MICROCOMPUTADOR

Se presentará la estructura de una microcomputadora basada en el modelo de Van Newman y se describirá la función de cada uno de los componentes de dicha estructura, se deberán mencionar brevemente otro tipo de estructuras.

III. ESTRUCTURA DE ELEMENTOS DE MEMORIA

Se estudiarán detalladamente las memorias de lectura únicamente (ROM), dando énfasis a los diagramas de tiempo de ambas memorias, así mismo se recomienda se lean las hojas de especificaciones de algunas de estas memorias y se hagan algunos ejemplos, en los cuales pueden surgir problemas debido a la velocidad de lectura y escritura de los microprocesadores.

IV. INTRODUCCION A LA PROGRAMACION DE MICROPROCESADORES

Se presentarán los pasos para el desarrollo de programas, dandole a los diagramas de flujo, así mismo se aconseja dar la aproximación de diseño arriba-abajo (TOP DOWN) usando las herramientas estándar de programación, secuencias, y repetición.

Se dará a conocer el lenguaje de máquina, la programación en lenguaje ensamblador y su diferencia con lenguajes de alto nivel.

V. PROGRAMACION DEL MICROPROCESADOR

Se estudiará el conjunto de instrucciones de un microprocesador en particular, agruparlas por función y estudiar los diferentes tipos de direccionamiento. Se recomienda hacer ejercicios de programación en tiempo real.

VI. INTERFASE DE MICROCOMPUTADORAS CON DISPOSITIVOS EXTERNOS

Se dará a conocer las formas de decodificación de direcciones para las memorias, puertos, y otros periféricos, se recomienda ubicarlos en diferentes direcciones aleatorias para reforzar la idea del proceso de decodificación de direcciones.

VII. COMUNICACION SERIE ASINCRONA Y SINCRONA

Se recomienda estudar un puerto real, programarlo en sus diferentes modos de operación preferentemente usando protocolos y después hacer una comparación con otro tipo de puertos paralelos similares.

Bibliografía

Microcontroladores PIC – Aprenda una profesión.
Ing. HoracioD. Vallejo
Quark S.R.l
2002

MICROCONTROLADORES PIC – Diseño practico de aplicaciones
Jose M. Angulo Usategui
Ignacio Angulo Martinez
Mc Graw Hill (200#)

Circuitos electrónicos y sus aplicaciones.
Grob Bernard
Ed. McGraw Hill.
1986 M

Electrónica fundamental para científicos.
Brophy, James J.
Ed. Reverté, S. A.
1974

PIC16C77X, PIC16F7X, PIC16F8X, PIC16F87XA, PIC18FXX8
Data Sheets
Microchip Technology Inc.

PICmicro™ Mid-Range MCU Family
Reference Manual

Desarrollo de un caso practico
“programadores cpld´s y microcontroladores”
Ismael Díaz Rangel

ENLACES A PÁGINAS DE INTERÉS

Agilent Technologies	www.agilent.com
Microchip Technology Inc.	www.microchip.com
Atmel Corp.	www.atmel.com
FES Aragón	http://informatica.aragon.unam.mx/
Maxim integrated Products	www.maxim-ic.com
UNAM	www.unam.mx

2. PLAN DE ESTUDIOS.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ENEP – ARAGON

Programa de Asignatura

Carrera:	INGENIERO MECANICO – ELECTRICISTA
Area de Conocimiento:	ELECTRONICA

DIVISION DE ESTUDIOS PROFESIONALES

Fecha de aprobación del:
* Consejo Técnico de la ENEP – Aragón JUNIO 28 DE 1991
* Consejo Universitario

Programa de la asignatura:	**MICROPROCESADORES**

Clave:	0586		Núm. De créditos:	10
Duración del curso:	Semanas:	16	Asignatura:	Obligatoria
	Horas:	96	Semestre:	Octavo
	Teoría:	4.0		
	Práctica:	2.0		

OBJETIVO DEL CURSO

Comprender la estructura y funcionamiento de los microprocesadores para aplicarlos en la solución de problemas de ingenieria.

TEMAS

Núm.	Nombre	Horas	
		Teoría	Práctica
I	INTRODUCCION A LOS MICROPROCESADORES Y MICROCOMPUTADORES	6.0	2.0
II	ESTRUCTURAS DE UN MICROCOMPUTADOR	10.0	6.0
III	ESTRUCTURA DE LOS ELEMENTOS DE MEMORIA	8.0	4.0
IV	INTRODUCCION A LA PROGRAMACION DE MICROPROCESADORES	6.0	4.0
V	PROGRAMACION DEL MICROPROCESADOR	10.0	6.0
VI	INTERFASE DE MICROCOMPUTADORAS CON DISPOSITIVOS EXTERNOS	6.0	2.0
VII	COMUNICACION SERIE ASINCRONA Y SINCRONA	8.0	4.0
VIII	COMUNICACION PARALELO	6.0	2.0
IX	APLICACIONES DE LOS MICROPROCESADORES	4.0	2.0
		64.0	32.0
HORAS TOTALES:		96.0	

Asignatura: MICROPROCESADORES

ANTECEDENTES, OBJETIVOS Y CONTENIDO DE LOS TEMAS

TEMA I INTRODUCCION A LOS MICROPROCESADORES Y RMICROCOMPUTADORES"

Antecedentes: Diseño Lógico

Objetivo: Dar a conocer los elementos constitutivos de un microcomputador.

Contenido:

I.1 Elementos de un microcomputador.

- El microprocesador, las memorias y las interfases.
- Comunicación entre los módulos.

TEMA II	"ESTRUCTURA DE UN MICROCOMPUTADOR"
Antecedentes:	Diseño Lógico.
Objetivo:	Conocer el funcionamiento lógico y físico de un microcomputador.
Contenido:	
II.1	Las señales.
II.2	Los elementos internos.
II.3	El funcionamiento interno: los diagramas de estado.
II.4	Estudio detallado de un microprocesador de 8 bits.

TEMA III	"ESTRUCTURA DE LOS ELEMENTOS DE MEMORIA"
Antecedentes:	Contenidos en la misma.
Objetivo:	Conocer el manejo y la tecnología de las memorias empleadas en los sistemas microcomputarizados.
Contenido:	
III.1	Estado actual de la tecnología.
III.2	Estructura de los sistemas de memoria.
III.3	Estudio detallado de varios casos (RAM, ROM)

TEMA IV	"INTRODUCCION A LA PROGRAMACION DE LOS MICROPROCESADORES"
Antecedentes:	Contenidos en la misma.
Objetivo:	Familiarizarse con la programación de los microprocesadores.
Contenido:	
IV.1	Lenguaje de Máquinas y Ensambladores.
IV.2	Monitor.
IV.3	Registro, Memoria, Buses.

TEMA V	"PROGRAMACION DEL MICROPROCESADOR"
Antecedentes:	Contenidos en la misma.
Objetivo:	Programar el microprocesador usando su conjunto de instrucciones para el desarrollo de programas de aplicación.
Contenido:	
V.1	Direccionamiento de Memorias y E/S.
V.2	Banderas y Brincos.
V.3	Conteo y Lazos de tiempo.
V.4	Funciones Aritméticas.
V.5	Conceptos Avanzados.
V.6	Interrupciones.

TEMA VI	"INTERFASE DE MICROCOMPUTADORAS CON DISPOSITIVOS EXTERNOS"
Antecedentes:	Contenidos en la misma.

Objetivo:	Conocer la conexión de la microcomputadora con otros dispositivos.

Contenido:

VI.1	Uso de líneas programadas Entrada/Salida para el control de dispositivos.
VI.2	Control de dispositivos con técnicas de Programación.
VI.3	Interfase RS-232 y lazo de corriente 20 MA.

TEMA VII	"COMUNICACION SERIE ASINCRONA Y SINCRONA"
Antecedentes:	Contenidos en la misma.
Objetivo:	Conocer la comunicación de datos en serie entre la microcomputadora y los dispositivos externos.

Contenido:

VII.1	Estructura de la comunicación Asíncrona.
VII.2	Interfases asíncronas, estudio del UART y similares.
VII.3	Estructura de la comunicación serie síncrona.
VII.4	Interfaces síncronas: los USART y similares.
VII.5	Comunicación síncrona compleja: Protocolos.

TEMA VIII	"COMUNICACION PARALELO"
Antecedentes:	Contenidos en la misma.
Objetivo:	Conocer la comunicación de datos en parelelo entre la microcomputadora y los dispositivos externos.

Contenido:

VIII.1	Estructura de la comunicación paralelo.
VIII.2	El saludo (handshake): tipos.
VIII.3	Interfases integrados: PIA, PIO, PPI y similares.
VIII.4	Interfases universales programables: VPI y similares.

TEMA IX	"APLICACIONES DE LOS MICROPROCESADORES"
Antecedentes:	Contenidos en la misma.
Objetivo:	Integrar los conocimientos adquiridos en el campo para poder aplicar los microprcesadores en el campo de la Ingeniería.

Contenido:

IX.1	Aplicación de los microprocesadores (8088, 8036, 80486, 68701, DSP566000) en instrumentación y control.

TECNICAS DE ENSEÑANZA		**ELEMENTOS DE EVALUACION**	
Exposición oral	(X)	Exámenes parciales	(X)
Exposición audiovisual	( )	Exámenes finales	(X)
Ejercicios dentro de clase	(X)	Trabajos y tareas fuera del aula	(X)
Ejercicios fuera del aula	(X)	Participación en clase	(X)
Seminarios	( )	Asistencia a prácticas	(X)
Lecturas obligatorias	( )	Otros:	( )
Trabajos de investigación	(X)		
Prácticas de taller o	(X)		

laboratorio
Prácticas de campo ( )
Otras:

ANTECEDENTES

Asignatura	Clave	Temas que se requieren
Diseño Lógico		TODOS

CONSECUENTES

Asignatura	Clave
Diseño de Sistemas con Microporcesadores	

BIBLIOGRAFIA

Texto	Temas de la materia para las que se recomienda:
Steve Ciarcia ; Construya una Microcomputadora ; Basado en el Z-80 ; Byte Books/Mc Graw Hill, 1985 350 págs.	I, II, III, IV y VI
James W. Gault/Russell L. Pimmel ; Sistemas Digitales Basados en un Microprocesador ; Mc Graw Hill de México, 1985, 461 págs.	I, II, III, IV, V y IX
H. Lilen ; Del Microprocesador al Microcomputador ; Ed. Marcombo Boixareu Editores, 1980 ; 461 págs.	I, II, III, IV, V, VI y IX
Serie Mundo Electrónico ; Microprocesadores y Microcomputadores ; Ed. Marcombo Boixareu Editores ; 2a, Edición, 1978, 182 págs.	I, II, III, VI y IX
Adam Osborne ; Introducción a las Microcomputadoras, Vol. I ; Osborne Mc Graw Hill ; 2a. Edición, 452 págs.	I, II, III, IV, V y VII
Hohewtein, Louis ; Computer Periphera for Minicomputers ; Microprocessors, and Personal Computers Mc Graw Hill, Book Xompany, 1980, 312 págs.	VI

Serie Mundo Electrónico José Mompín Poblet ; Interconexión de Periféricos a Microprocesadores ; Publicaciones Marcombo, S. A., 1984, 228 págas.	VI, VII y VIII
Kenneth L. Short ; Microprocesadores y Lógica Programada ; Colección Electrónica/Informática ; Editorial Gustavo Gili, S. A. 1981 523 págs.	I, II, III, IV, V, VI, VII y IX
Peatman, John ; Microcomputer base design ; Mc Graw Hill, 1977	TODOS
Osborne & Associates Inc. ; Z-80 Programing for Logic Design ; Some Real Products.	TODOS

Asignatura: MICROPROCESADORES

SUGERENCIAS PARA IMPARTIR LA ASIGNATURA

I. INTRODUCCION A LOS MICROPROCESADORES Y MICROCOMPUTADORAS

Se dará a conocer la diferencia entre un microprocesador y una microcomputadora, así mismo se hará un bosquejo histórico del desarrollo de los microprocesadores y microcomputadoras monolíticas, haciendo notar las diferencias existentes entre éstas.

II. ESTRUCTURA DE UN MICROCOMPUTADOR

Se presentará la estructura de una microcomputadora basada en el modelo de Van Newman y se describirá la función de cada uno de los componentes de dicha estructura, se deberán mencionar brevemente otro tipo de estructuras.

III. ESTRUCTURA DE ELEMENTOS DE MEMORIA

Se estudiarán detalladamente las memorias de lectura únicamente (ROM), dando énfasis a los diagramas de tiempo de ambas memorias, así mismo se recomienda se lean las hojas de especificaciones de algunas de estas memorias y se hagan algunos ejemplos, en los cuales pueden surgir problemas debido a la velocidad de lectura y escritura de los microprocesadores.

IV. INTRODUCCION A LA PROGRAMACION DE MICROPROCESADORES

Se presentarán los pasos para el desarrollo de programas, dandole a los diagramas de flujo, así mismo se aconseja dar la aproximación de diseño arriba-abajo (TOP DOWN) usando las herramientas estándar de programación, secuencias, y repetición.

Se dará a conocer el lenguaje de máquina, la programación en lenguaje ensamblador y su diferencia con lenguajes de alto nivel.

V. PROGRAMACION DEL MICROPROCESADOR

Se estudiará el conjunto de instrucciones de un microprocesador en particular, agruparlas por función y estudiar los diferentes tipos de direccionamiento. Se recomienda hacer ejercicios de programación en tiempo real.

VI. INTERFASE DE MICROCOMPUTADORAS CON DISPOSITIVOS EXTERNOS

Se dará a conocer las formas de decodificación de direcciones para las memorias, puertos, y otros periféricos, se recomienda ubicarlos en diferentes direcciones aleatorias para reforzar la idea del proceso de decodificación de direcciones.

VII. COMUNICACION SERIE ASINCRONA Y SINCRONA

Se recomienda estudar un puerto real, programarlo en sus diferentes modos de operación preferentemente usando protocolos y después hacer una comparación con otro tipo de puertos paralelos similares.