PONTICIA UNIVERSIDAD CATOLICA DEL ECUADOR
SEDE AMBATO

ESCUELA DE INGENIERIA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA LA OBTENCIÓN DEL TITULO DE INGENIERÍA DE SISTEMAS

"Sistema De Seguridad Para El Accesos De Entradas Y Salidas Utilizando Microcontroladores"

Fabian Enrique Esparza Silva

DIRECTOR DE LA DISERTACIÓN: Ing. Wilberto Sánchez

AMBATO 27 DE AGOSTO DEL 2003

Miriam Viteri de Mera

23-09-03.

CAPITULO I

MARCO TEORICO

1.- Aspectos generales de los Microcontroladores.

1.1- Generalidades.

1.2- Reseña histórica de los Microprocesadores y Microcontroladores.

1.3.- Familias de lo microcontroladores

CAPITULO II

PRESTACION DEL PROYECTO

2 Selección del microcontrolador.

2.1 Definición del microcontrolador Pic16C84

2.2 Análisis de los Microcontroladores

2.3 Análisis y Diseño del Microcontrolador

2.4 Ventajas de un Microcontrolar sobre el Microprocesador.

2.5 Diseño con un Microcontrolador vs. Microprocesador

2.6 Arquitectura de los Microcontroladores Pic.

2.7 Características especiales de un Microcontrolador Pic.

2.8 Arquitectura interna del Pic16C84

CAPITULO I

MARCO TEORICO

CAPITULO I

MARCO TEORIO

1.- ASPECTOS GENERALES DE LOS MICROCONTROLADORES

Aquí hablaremos de los aspectos mas importantes de los microcontroladores y las necesidades en la vida diaria y las clases de microcontroladores mas utilizado que existen.

1.1 GENERALIDADES

Los microcontroladores están conquistando el mundo. Están presentes en nuestro trabajo, en nuestra casa y en nuestra vida, en general. Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los computadores, en los teléfonos, en los hornos microondas y los televisores de nuestro hogar.

Pero la invasión acaba de comenzar y el nacimiento del siglo XXI será testigo de la conquista masiva de estos diminutos computadores, que gobernarán la mayor parte de los aparatos que fabricaremos y usamos los humanos.

Un microcontrolador es sencillo, aunque es un completo computador con su UCP (Unidad central de proceso), memoria para albergar un programa que es fácil de instalar, memorias para uso general y entradas y salidas para poder ampliarse o comunicarse con el exterior, sistemas de control de tiempo internos y externos, puertos serial y paralelo, conversores A/D y D/A etc. todo ello contenido en un mismo circuito integrado.

Según el tipo empleado pueden diferenciarse en la cantidad y tipo de memoria, cantidad y tipo de entradas y salidas, temporizadores, módulos de control internos y externos, etc.

Resumiendo estamos en la actualidad rodeados de microcontroladores cada uno de ellos con sus características propias.

Cada tipo de microcontrolador sirve para una serie de casos y es el creador del producto el que debe de seleccionar que microcontrolador es el idóneo para cada uso.

La aplicación de un microcontrolador en un circuito que reduce el numero de averías, al disminuir en numero de componentes, así como el volumen, el stock y el trabajo.

Importancias y Necesidades en la Sociedad

El microcontrolador es uno de los logros más sobresalientes del este siglo . Hace un cuarto de siglo tal afirmación habría parecido absurda. Pero cada año, el microcontrolador se acerca más al centro de nuestras vidas, forjándose un sitio en el núcleo de una máquina tras otra. Su presencia ha comenzado a cambiar la forma en que percibimos el mundo e incluso a nosotros mismos.

Cada vez se hace más dificil pasar por alto el microcontrolador como otro simple producto en una larga línea de innovaciones tecnológicas.

Ninguna otra invención en la historia se ha diseminado tan aprisa por todo el mundo o ha tocado tan profundamente tantos aspectos de la existencia humana. Hoy existen casi 15,000 millones de microchips de alguna clase en uso.

¿Quién puede dudar que el microcontrolador no sólo está transformando los productos que usamos, sino también nuestra forma de vivir y, por último, la forma en que percibimos la realidad?

No obstante que reconocemos la penetración del microcontrolador en nuestras vidas, ya estamos creciendo indiferentes a la presencia de esos miles de máquinas diminutas que nos encontramos sin saberlo todos los días. Así que, antes de que se integre de manera demasiado imperceptible en nuestra diaria existencia, es el momento de celebrar al microcontrolador y la revolución que ha originado, para apreciar el milagro que es en

realidad cada uno de esos chips de silicio diminutos y meditar acerca de su significado para nuestras vidas y las de nuestros descendientes.

Primero, la revolución. Si desecháramos el microchip de todas y cada una de las aplicaciones en las que ahora encuentra un hogar, terminaríamos aturdidos y aterrorizados por la pérdida. La cocina moderna quedaría casi inservible porque el horno de microondas, la máquina lavavajillas y la mayoría de otros aparatos domésticos no funcionarían más. El televisor y la videocasete se reducirían a la negrura, el equipo estereofónico se volvería mudo y la mayoría de los relojes se detendrían.

El automóvil no arrancaría. Los aviones no podrían despegar del suelo. El sistema telefónico quedaría muerto, al igual que la mayoría de las luces de las calles, termostatos y, desde luego, unos 500 millones de computadoras. Y éstas son tan sólo las aplicaciones más evidentes.

Todas las fábricas del mundo industrial pararían y también la red eléctrica, las bolsas de valores y el sistema bancario global. Pero vayamos más a fondo: los marcapasos se detendrían también, al igual que el equipo quirúrgico y los sistemas de supervisión fetal.

Todo debido a la pérdida de un diminuto cuadradito de silicio del tamaño de la uña de un dedo, que pesa menos que una estampilla postal, y construido tan sólo de cristal, fuego, agua y metal.

Desde luego, éste es el milagro. Decenas de miles de microcontroladores se integran todos los días en las plantas de manufactura más avanzadas jamás conocidas, donde un simple gránulo de polvo puede significar el desastre, donde los procesadores ocurren en ambientes más limpios que ningún otro sitio en la tierra. Incluso el agua que utiliza para enjuagar las superficies de los chips terminados es más pura que la que se utiliza en la cirugía a corazón abierto.

El más grande atributo del microcontrolador es que puede integrar inteligencia casi a cualquier artefacto. Se le puede entrenar para adaptarse a su entorno, responder a condiciones cambiantes y volverse más eficiente y que responda a las necesidades

únicas de sus usuarios. Desmonte cualquier rincón de la vida moderna, retire la capa exterior de cajas y material de construcción y luces parpadeantes, y como semillas en una maceta, aparecerán microcontroladores por millones.

Nombraremos algunos puntos importantes de los microcontroladores:

- **Prestaciones**

  Su reducido tamaño y bajo costo permiten que se pueda incorporar en sistemas que antes no tenían controladores. Por ejemplo en automotores.

- **Fiabilidad**

  Al tener menos componentes, se disminuye el riesgo de fallas y precisa menos calibraciones.

- **Flexibilidad**

  Como el control se hace mediante un programa, su modificación solo precisa cambios de programación.

- **Aplicaciones**

  Cada vez existen más productos que incorporan un microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir tamaño y costo, mejorar su fiabilidad y disminuir el consumo.

Los campos más destacados en los que se emplean microcontroladores son los siguientes:

- Periféricos y dispositivos auxiliares de las computadoras.

- Electrodomésticos.

- Aparatos portátiles y de bolsillo.
- Máquinas expendedoras y juguetería.
- Instrumentación.
- Industria automotriz.
- Control industrial y robótica.
- Sistemas de navegación espacial.
- Seguridad y alarma.

Es muy importante porque hoy en día es común encontrar microcomputadores en las cafeteras, hornos microondas, videograbadoras, alarmas, automóviles, etc.

Las aplicaciones son infinitas, el único límite es la imaginación. La posibilidad de manejar señales de entrada y de salida, así como su capacidad para procesar datos y tomar decisiones, lo convierten en uno de los elementos electrónicos más importantes, necesario y versátiles que existen.

Cuando se habla de dispositivos de entrada se hace referencia a todos los elementos que pueden cambiar de estado ante alguna determinada condición y genera una señal que puede ser utilizada por el microcontrolador para tomar alguna decisión, por ejemplo un teclado, un interruptor, un sensor, etc.

La posibilidad de manejar señales de entrada y de salida, así como su capacidad para procesar datos y tomar decisiones, lo convierten en uno de los elementos electrónicos más versátiles que existen.

Historia de los Microprocesadores y Microcontroladores.

A partir de 1970, el panorama de la electrónica cambió radicalmente con la aparición del microprocesador. Vendría la época de oro del Z-80, el 8085, el 6800 y otros microprocesadores utilizados como elementos centrales en aparatos de control, y se consolidarían las técnicas de integración, el estudio de las memorias, la programación en lenguaje de máquina y la adaptación de periféricos de todo tipo.

El primer microprocesador fue el Intel 4004 diseñado para reemplazar grandes cantidades de circuitos integrados, producido en 1971. Se desarrolló originalmente para una calculadora, y resultaba revolucionario para su época. Contenía 2.300 transistores en un microprocesador de 4 bits que sólo podía realizar 60.000 operaciones por segundo.

Poco tiempo después, sin embargo, el 1 de abril de 1972, Intel anunciaba una versión mejorada de su procesador. Se trataba del 8008, que contaba como principal novedad con un bus de 8 bits, y la memoria direccionable se ampliaba a los 16 Kb. Además, llegaba a la cifra de los 3500 transistores, casi el doble que su predecesor, y se le puede considerar como el antecedente del procesador que serviría de corazón al primer ordenador personal.

Justo dos años después, Intel anunciaba ese tan esperado primer ordenador personal, de nombre Altair, cuyo nombre proviene de un destino de la nave Enterprise en uno de los capítulos de la popular serie de televisión Star Trek la semana en la que se creó el ordenador. Este ordenador tenía un coste de entorno a los 400 dólares de la época, y el procesador suponía multiplicar por 10 el rendimiento del anterior, gracias a sus 2 MHz de velocidad (por primera vez se utiliza esta medida), con una memoria de 64 Kb. En unos meses, logró vender decenas de miles de unidades, en lo que suponía la aparición del primer ordenador que la gente podía comprar, y no ya simplemente utilizar.

Sin embargo, como todos sabemos, el ordenador personal no pasó a ser tal hasta la aparición de IBM, el gigante azul, en el mercado. Algo que sucedió en dos ocasiones en los meses de junio de 1978 y de 1979. Fechas en las que respectivamente, hacían su aparición los microprocesadores 8086 y 8088, que pasaron a formar el denominado

IBM PC, que vendió millones de unidades de ordenadores de sobremesa a lo largo y ancho del mundo.

El éxito fue tal, que Intel fue nombrada por la revista "Fortune" como uno de los mejores negocios de los años setenta.

De los dos procesadores, el más potente era el 8086, con un bus de 16 bits (por fin), velocidades de reloj de 5, 8 y 10 MHz, 29000 transistores usando la tecnología de 3 micras y hasta un máximo de 1 Mega de memoria direccionable. El rendimiento se había vuelto a multiplicar por 10 con respecto a su antecesor, lo que suponía un auténtico avance en lo que al mundo de la informática se refiere. En cuanto al procesador 8088, era exactamente igual a éste, salvo la diferencia de que poseía un bus de 8 bits en lugar de uno de 16, siendo más barato y obteniendo mejor respaldo en el mercado.

En 1980, aproximadamente, los fabricantes de circuitos integrados dieron a conocer un nuevo chip llamado microordenador, el cual contenía toda la estructura de un microcomputador, es decir, unidad de proceso (CPU), memoria RAM, memoria ROM y circuitos de entrada/ salida.

En el año 1982, concretamente el 1 de febrero, Intel daba un nuevo vuelco a la industria con la aparición de los primeros 80286. Como principal novedad, cabe destacar el hecho de que por fin se podía utilizar la denominada memoria virtual, que en el caso del 286 podía llegar hasta 1 Giga. También hay que contar con el hecho de que el tiempo pasado había permitido a los ingenieros de Intel investigar más a fondo en este campo, movidos sin duda por el gran éxito de ventas de los anteriores micros. Ello se tradujo en un bus de 16 bits, 134000 transistores usando una tecnología de 1.5 micras, un máximo de memoria direccionable de 16 Megas y unas velocidades de reloj de 8, 10 y 12 MHz. En términos de rendimiento, podíamos decir que se había multiplicado entre tres y seis veces la capacidad del 8086, y suponía el primer ordenador que no fabricaba IBM en exclusiva, sino que otras muchas compañías, alentadas por los éxitos del pasado, se decidieron a crear sus propias máquinas. Como dato curioso, baste mencionar el hecho

de que en torno a los seis años que se le concede de vida útil, hay una estimación que apunta a que se colocaron en torno a los 15 millones de ordenadores en todo el mundo.

Los microprocesadores modernos tienen una capacidad y velocidad mucho mayores. Entre ellos figuran el Intel Pentium Pro, con 5,5 millones de transistores; el UltraSparc-II, de Sun Microsystems, que contiene 5,4 millones de transistores; el PowerPC 620, desarrollado conjuntamente por Apple, IBM y Motorola, con 7 millones de transistores, y el Alpha 21164A, de Digital Equipment Corporation, con 9,3 millones.

La tecnología de los microprocesadores y de la fabricación de circuitos integrados está cambiando rápidamente. En la actualidad, los microprocesadores más complejos contienen unos 10 millones de transistores.

Se cree que el factor limite en la potencia de los microprocesadores acabará siendo el comportamiento de los propios electrones al circular por los transistores. Cuando las dimensiones se hacen muy bajas, los efectos cuánticos debidos a la naturaleza ondulatoria de los electrones podrían dominar el comportamiento de los transistores y circuitos.

Puede que sean necesarios nuevos dispositivos y diseños de circuitos a medida que los microprocesadores se aproximan a dimensiones atómicas. Para producir las generaciones futuras de microchip se necesitarán técnicas como la epitaxia por haz molecular, en la que los semiconductores se depositan átomo a átomo en una cámara de vacío ultra elevado, o la microscopía de barrido de efecto túnel, que permite ver e incluso desplazar átomos individuales con precisión.

Microprocesador, circuito electrónico que actúa como unidad central de proceso de un ordenador, proporcionando el control de las operaciones de cálculo. Los microprocesadores también se utilizan en otros sistemas informáticos avanzados, como impresoras, automóviles o aviones.

El microprocesador es un tipo de circuito sumamente integrado. Los circuitos integrados, también conocidos como microchip o chips, son circuitos electrónicos

complejos formados por componentes extremadamente pequeños formados en una única pieza plana de poco espesor de un material conocido como semiconductor. Los microprocesadores modernos incorporan hasta 10 millones de transistores (que actúan como amplificadores electrónicos, osciladores o, más a menudo, como conmutadores), además de otros componentes como resistencias, diodos, condensadores y conexiones, todo ello en una superficie comparable a la de un sello postal.

Un microprocesador consta de varias secciones diferentes. La unidad aritmético-lógica (ALU, siglas en inglés) efectúa cálculos con números y toma decisiones lógicas; los registros son zonas de memoria especiales para almacenar información temporalmente; la unidad de control descodifica los programas; los buses transportan información digital a través del chip y de la computadora; la memoria local se emplea para los cómputos realizados en el mismo chip.

Este se concibió como un dispositivo programable que puede ejecutar un sinnúmero de tareas y procesos.

La familia de microcontroladores PIC fue introducida en el mercado en el año 1985 por la empresa Microchip Technology. Esta familia incluye un gran número de microcontroladores, por lo que resulta posible encontrar un PIC adecuado para casi cualquier tipo de aplicación.

Algunos microcontroladores más especializados poseen además convertidores análogo digital, temporizadores, contadores y un sistema para permitir la comunicación en serie y en paralelo.

Se pueden crear muchas aplicaciones con los microcontroladores. Estas aplicaciones de los microcontroladores son ilimitadas (el límite es la imaginación) entre ellas podemos mencionar: sistemas de alarmas, juego de luces, paneles publicitarios, etc. Controles automáticos para la Industria en general. Entre ellos control de motores DC/AC y motores de paso a paso, control de máquinas, control de temperatura, control de tiempo, adquisición de datos mediante sensores, etc.

En la actualidad a estos microcontroladores tales como MOTOROLA 6805, INTEL 8051, MICROCHIP16CXX, etc.; se los encuentra comúnmente en hornos microondas, vídeo grabadoras, en alarmas y en otros sistemas que se requieran controlar diferentes procesos.

1.3 Familias de microcontroladores

Actualmente existen en el mercado varias marcas reconocidas como las principales, dadas sus características, difusión y usos en productos de consumo masivo. Entre ellas están Motorola, Intel, Philips, National y Microchip. A continuación mencionaremos algunas.

Familia de Philips 8051

La familia de microcontroladores y periféricos que tiene por base el *8051* está basada totalmente en el patrón industrial para 8 bits, de alta performance, que tiene una arquitectura optimizada para aplicaciones en control secuencia en tiempo real.

Los componentes de esta familia encuentran aplicaciones que van desde el control de máquinas industriales y de instrumentación hasta el control automotriz.

Con excepción de la 83C751 , todos los dispositivos de esta familia pueden manejar hasta 64 bytes, tanto de programa como de memoria de datos.

En la tabla 1 tenemos los dispositivos que componen esta familia de microcontroladores.

Tabla1 –8051 Familia de los Microcontroladores

Nombre	Versión sin ROM	Versión con RAM	BYTES ROM	BYTES RAM	TIMERS 16 Bits	TIPOS
8051	8031	-------	4k	128	2	NMOS
80C053	-------	87C054	8k	192	2	CMOS
83CL410	80CL410	-------	4k	128	2	CMOS

83C451	83C451	87C451	4k	128	2	CMOS
83C528	83C528	87C528	32k	512	3 + Wd	CMOS
82C550	82C550	87C550	4k	128	2 + Wd	CMOS
83C552	83C552	87C552	8k	256	3 + Wd	CMOS
83C562	83C562	-------	8k	256	3 + Wd	CMOS
83C652	83C652	87C652	8k	256	2	CMOS
83C654	-------	87C654	16k	256	2	CMOS
83C751	-------	87C751	2k	64	1	CMOS
83C752	-------	87C752	2k	64	1	CMOS
83C851	83C851	-------	4k	128	2	CMOS

El microcontrolador 80C51 es la versión CMOS del 80C51, siendo totalmente compatible con el 8051 en términos de funcionamiento.

Sin embargo, como se trata de un dispositivo CMOS (a diferencia de el 8051 , que es NMOS) el consumo es mucho menor.

Familia National

Los microcontroladores COPSAx7 OTP son miembros de la Familia de los *COP8* y utilizan como corazón un arquitectura con chip de 8 bits. Estos dispositivos fueron fabricados bajo el proceso EPROM de alta densidad, de la National Semiconductor.

Las características "Clave" incluye arquitectura mapeada de memoria de 8 bits, un controlador y un timer de 16 bits con dos registro de 16 bits que se vinculan con tres selectores (Generación PWM de Procesador Independiente, Contador Externa de Eventos y Disponibilidad de Captura de Entradas), dos selectores HALT/IDLE con capacidad para guardar y con la posibilidad de la presencia de un interruptor multimedia para despertarse, un oscilador R/C tipo chip, salidas de corriente elevadas, opciones seleccionables para usuario, tales como WATCHDOG, configuración del Oscilador y potencia de reseteo.

La Línea COP emplea una arquitectura interna tipo Harvard (con buses de instrucciones y datos separados), que permite que la mayor parte de las instrucciones (el 77%) puedan ser ejecutadas en un único ciclo de instrucción de un microsegundo, al realizar simultáneamente el acceso al espacio de instrucciones (EPROM) y al espacio de datos (memoria RAM y periféricos).

A diferencia entre la línea COP y otros microcontroladores obtenibles en plaza que también presenta arquitectura Harvard, la complejidad propia de este tipo de arquitectura ha sido adoptada sin limitar, por ello, cantidad y versatilidad de las instrucciones disponibles, ni eliminar periféricos.

Poseen abundantes pines de entrada / salidas paralela (16 I/O en los chips de 20 patas y 24 I/O en los de 28 patas), donde ciertas entradas son de tipo SCHMITT Trigger y pueden ser programadas para incorporar internamente una resistencia de pullup y operarán como salidas.

Finalmente, el diseño total de los COP ha sido realizado con un control de velocidad de transistores (slew rate) y niveles de señal internos al chip.

Familia de los Microcontroladores National

Dispositivo	**RAM**	**ROM**	**Encapsulado & I/O**	
			Encapsulado	**Pines**
COP8SAC7	4K	128	20 DIP	16
COP8SAB7	2K	128	20 DIP	16
COP8SAA7	1K	64	16 DIP	12

Familia Intel 8051

El 8051 es el primer microcontrolador de la familia introducida por Intel Corporation. La familia 8051 de microcontroladores son controladores de 8 bits capaces de direccionar hasta 64 kbytes de memoria de programa y una separada memoria de datos de 64 kbytes. El 8031(la versión sin ROM interna del 8051, siendo esta la única

diferencia) tiene 128 bytes de RAM interna (el 8032 tiene RAM interna de 256 bytes y un temporizador adicional).

El 8031 tiene dos temporizadores/contadores, un puerto serie, cuatro puertos de entrada/salida paralelas de propósito general (P0, P1, P2 y P3) y una lógica de control de interrupción con cinco fuentes de interrupciones. Al lado de la RAM interna, el 8031 tiene varios Registros de Funciones especiales(SFR)(Special Function Registers) que son para control y registros de datos.

Los SFRs también incluyen el registro acumulador, el registro B, y el registro de estado de programa(Program Status Word)(PSW), que contienen los Flags del CPU.

Bloques separados de memoria de código y de datos se denomina como la Arquitectura Harvard.

El 8051 tiene dos señales de lectura separadas, los pines RD(P3.7, pin 17) y PSEN(pin 29). El primero es activado cuando un byte va ser leído desde memoria de datos externo; el otro, cuando un byte va ser leído desde memoria de programa externo.

Ambas de estas señales son señales activas en nivel bajo. Esto es, ellos son aclarados a nivel lógico 0 cuando están activados. Todo código externo es buscado desde memoria de programa externo. En adición, bytes de memoria de programa externo pueden ser leídos por instrucciones de lectura especiales, tal como la instrucción MOVC. Hay también instrucciones separadas para leer desde memoria de datos externo, tal como la instrucción MOVX.

Esto significa que las instrucciones determinan que bloque de memoria es direccionado, y la señal de control correspondiente, o RD o PSEN, es activado durante el ciclo de lectura de memoria. Un único bloque de memoria puede ser mapeado para actuar como memoria de datos y de programa.

Esto es lo que se llama la arquitectura Von Neuman. Para leer desde el mismo bloque usando o la señal RD o la señal PSEN, las dos señales son combinadas con una

operación AND lógico. La arquitectura Harvard es algo extraño en sistemas de evaluación, donde código de programa necesita ser cargado en memoria de programa.

Adoptando la arquitectura Von Neuman, el código puede ser escrito a la memoria como bytes de datos y luego ejecutado como instrucciones de programa.

La ROM interna del 8051 y el 8052 no pueden ser programados por el usuario. El usuario debe suministrar el programa al fabricante, y el fabricante programa los microcontroladores durante la producción. Debido a costos, la opción de la ROM programado por el fabricante no es económica para producción de pequeñas cantidades.

El 8751 y el 8752 son las versiones Erasable Programmable Read Only Memory(EPROM) del 8051 y el 8052. Estos pueden ser programados por los usuarios.

Durante la década pasada muchos fabricantes introducieron miembros mejorados del microcontrolador 8051. Las mejoras incluyen más memoria, más puertos, convertidores análogo / digital; más temporizadores, más fuentes de interrupción, watchdog timers, y subsistemas de comunicación en red. Todos los microcontroladores de la familia usan el mismo conjunto de instrucciones, el MCS-51. Las características mejoradas son programadas y controladas por SFRs adicionales.

Esta familia de microcontrolador de 8 bus contiene varias referencias, cada una de ellas; acondicionada para aplicaciones específicas. Todas las versiones existentes tienen la misma CPU, memoria RAM, temporizadores, puertos paralelos y entradas/salidas de tipo serial.

El 8051 tiene 4 Kbytes de memoria ROM que debe programarse durante el proceso de fabricación del circuito integrado. En el 875 1, la memoria ROM se ha reemplazado por una memoria EPROM que el usuario puede programar y borrar con luz ultravioleta.

Referencia	**Memoria Rom**	**Memoria Ram**	**Timers**
8051	4k	128	2
8052	8k	256	3

8031	Externa	128	2
8032	Externa	256	3
8751	4k(Eprom)	128	2

Microcontroladores de la familia Intel 8051

El 8031 es un caso especial; no tiene memoria ROM interna y por lo tanto, la memoria de programa se debe colocar externamente. Para comunicarse con la memoria externa, el micro debe usar tres de los cuatro puertos paralelos de entrada/salida.

Familia Motorola

El 68hc11 de la familia Motorola, es un potente microcontrolador de 8 bits en su bus de datos, 16 bits en su bus de direcciones, con un conjunto de instrucciones que es similar a los más antiguos miembros de la familia 68xx (6801, 6805, 6809). Dependiendo del modelo, el 68hc11 tiene internamente los siguientes dispositivos: EEPROM o OTPROM, RAM, digital I/O, timers, A/D converter, generador PWM, y canales de comunicación sincrónica y asincrónica (RS232 y SPI). La corriente típica que maneja es menor que 10ma.

El CPU tiene 2 acumuladores de 8 bits (A y B) que pueden ser concatenado para suministrar un acumulador doble de 16 bits(D). Dos registros índices de 16 bits son presentes (X, Y) para suministrar indexsamiento para cualquier lugar dentro del mapa de memoria. El tener dos registros índices significa que el 68hc11 es muy bueno para el procesamiento de datos.

Aunque es un microcontrolador de 8 bits, el 68hc11 tiene algunas instrucciones de 16 bits (add, subtract, 16 * 16 divide, 8 * 8 multiply, shift, y rotates). Un puntero de pila de 16 bits está también presente, y las instrucciones son suministradas para manipulación de la pila. Típicamente el bus de datos y direcciones están multiplexados.

El temporizador comprende de un único contador de 16 bits y hay un preescalador programable para bajarlo si es requerido. Viene con un convertidor A-D que es típicamente de 8 canales y 8 bits de resolución, aunque el G5 tiene un A/D de 10 bits.

Viene con una Interface de comunicaciones serie (SCI) – comunicaciones serie asíncrona; formato de datos 1 bit start, 8 o 9 bits de datos, y un bit de parada. Velocidad en baudios desde 150 hasta 312500 (312500 es usando un reloj E de 4mhz). Tiene una Interface periférico serie (SPI) - comunicaciones serie sincrónica.

Es una de las más utilizadas en el mundo. Ha sido optimizada para aplicaciones de control especializado, en lugar de procesamiento de datos, y forma parte de dispositivos de producción masiva como juguetes, equipos de vídeo, impresoras, módem, electrónica automotriz y electrodomésticos.

Cada año aparecen nuevos modelos que reemplazan a los anteriores. Todos los dispositivos básicos están construidos a partir de la misma CPU de 8 bits y tiene RAM, ROM, puertos de entrada/salida y temporizadores; algunos tienen, además, puertos seriales, convertidores análogo a digital y memorias EEPROM o EPROM.

Referencia	Memoria Rom	Memoria Ram	Timers	Otros
68HC05C4	4k	176	1	
68HC0508	8k	176	1	
68HC05C2	2k	176	1	
68HC705C4	4k(Eprom)	176	1	
68HC705K1	504	32 y 64	1	
68HC05BM	2k	176	1 Mejorado	Conversor a/d

Microcontroladores de la familia Motorola 6805

Familia Siemens

El Siemens SAB80C515 es un miembro mejorado de la familia 8051 de microcontroladores. El 80C515 es de tecnología CMOS que tipicamente reduce los

requerimientos de energía comparado a los dispositivos no-CMOS. Las características que tiene frente al 8051 son más puertos, un versátil convertidor análogo a digital, un optimizado Timer 2, un watchdog timer, y modos de ahorro de energía sofisticados. El 80C515 es completamente compatible con el 8051. Esto es, usa el mismo conjunto de instrucciones del lenguaje assembly MCS-51.

Las nuevas facilidades del chip son controladas y monitoreadas a través de SFRs adicionales. El 80C515 tiene todas las SFRs del 8051, y de este modo puede correr cualquier programa escrito para el 8051 con la excepción del uso del registro prioridad de interrupción IP. Por tanto si un programa 8051 usa prioridades de interrupción, debe ser modificado antes de que se ejecute sobre el 80C515. El agobio de modificar código 8051 existente es fácilmente justificado por la disponibilidad de más fuentes de interrupción y prioridades del 80C515.

Familia Microchip

Los microcontroladores PIC de Microchip Technolohy Inc. combinan una alta calidad, bajo coste y excelente rendimiento. Un gran número de estos microcontroladores son usados en una gran cantidad de aplicaciones tan comunes como periféricos del ordenador, datos de entrada automoción de datos, sistemas de seguridad y aplicaciones en el sector de telecomunicaciones.

Tanto la familia del PIC16XX como la del PIC17XX están apoyadas por un rango de usuario de sistemas de desarrollo amistosos incluso programadores, emuladores y tablas de demostración.

Así mismos ambas familias están apoyadas por una gran selección de software incluyendo ensambladores, simuladores, etc.

Dado que las aplicaciones sencillas precisan pocos recursos y las aplicaciones más complejas requieren numerosos y potentes recursos, Microchip construye diversos modelos de microcontroladores orientados a cubrir las necesidades de cada proyecto. Siguiendo esta filosofía Microchip oferta diferentes gamas de microcontroladores:

TIPO	PINES	BITS	MEMORIA	OTROS
PIC12C5XX	8-Pin	8-Bit	EEPROM	CMOS
PIC12CE5XX	8-Pin	8-Bit	EEPROM	CMOS-A/D
PIC12CE67X	8-Pin	8-Bit	EEPROM	CMOS-A/D
PIC16C55X	8-Pin	8-Bit	EPROM-Based	CMOS
PIC16X62X	8-Pin	8-Bit	EPROM-Based	CMOS
PIC16C71X	18/20Pin	8-Bit	EEPROM	CMOS
PIC16C78X	18-Pin	8-Bit	EEPROM	USB, PS/2,CMOS
PIC16C77X	28/40-Pin	12-bit.	EEPROM	CMOS,A/D, F.
PIC16X8X	18/40-Pin	8-Bit	EEPROM	CMOS FLASH
PIC17C75X	18-pin	8-Bit	EPROM	CMOS
PIC18F0XX	28/40-Pin	8-Bit	EEPROM	PLVD y PWM,F.
PIC18FXX31	28/40-Pin	8-Bit	EEPROM	PLVD, PBOR,F.
PIC18FXX5	28/40-Pin	10-Bit	EEPROM	USB y A/D

Dentro de cada gama se dispone de una gran variedad de modelos y encapsulados, pudiendo seleccionar el que mejor se adapte a cada proyecto. En el año 2000 se comercializaron más de un centenar de modelos de PIC que cubren desde los "enanos" de ocho patillas y mínimos recursos hasta los "avanzados" de 84 patillas enormemente potentes.

Está formada por una amplia variedad de componentes con diferentes tamaños de memoria, diferentes velocidades, diferentes tipos de encapsulado y diferente número de pines de entrada/salida.

El conjunto de instrucciones es de sólo 35, por eso se dice que es un microcontrolador de tipo RISC (Reduced Instruction Set Computer Computador con Set de Instrucciones Reducido) a diferencia de las anteriores familias que utilizan la tecnología CISC (Complex Instruction Set Computer - Computador con Set de Instrucciones Complejo).

Referencia	Memoria Rom	Memoria Ram	Timers	Otros
16C54	512	32	12	
16C57	2k	80	20	

16C61	1k	48	12	Interrupción
16C71	1k	48	12	Conversor a/d
16C84	1k(EEprom)	48	12	
12C509	1k	48	6	Encapsulado 8pin

Microcontroladores de la familia PIC16CXX de Microchip.

Esta familia sobresale por estar muy difundida actualmente a nivel mundial; la mayoría de revistas de electrónica en el mundo la usan para sus proyectos y le dedican artículos periódicamente. Su flexibilidad, configuraciones para todas las necesidades y bajo costo, la hacen muy atractiva para los consumidores a gran escala y para los estudiantes o diseñadores independientes.

Familia Altair

ALTAIR es el nombre genérico de una familia de microcontroladores de propósito general compatibles con la familia 51. Todos ellos son programables directamente desde un equipo PC mediante nuestro lenguaje macroensamblador, o bien mediante otros lenguajes disponibles para la familia 51 (BASIC, C, ...).

Microcontrolador	Bits	Direcionamiento	Mememoria
ALTAIR	**8 bits**	**128 Kbytes**	**EPROM**

Microcontroladores de la familia ALTAIR

Unos microcontroladores ALTAIR se diferencian de otros por el número de entradas salidas, periféricos (DAC, ADC, WATCHDOG, PWM, velocidad de ejecución, etc.).

Por lo que la elección de un modelo u otro dependerá de las necesidades. Como entrenador o sistema de iniciación recomendamos la utilización de un ALTAIR 32 BASICO o bien un ALTAIR 535A completo. En proyectos avanzados o desarrollos profesionales puede ser preferible un ALTAIR 537 A.

Tanto al 535 como al 537 se pueden complementar con nuestra EMULADOR EPROM PARA 535/537, que actuará como un emulador de EPROMs. Con ello facilitará notablemente la puesta a punto de las aplicaciones.

CAPITULO II

PRESTACION DEL PROYECTO

CAPITULO II

PRESENTACION DEL PROYECTO

2. Selección Del Microcontrolador

A la hora de escoger el microcontrolador a emplear en un diseño concreto hay que tener en cuenta multitud de factores, como la documentación y herramientas de desarrollo disponibles y su precio, la cantidad de fabricantes que lo producen y por supuesto las características del microcontrolador (tipo de memoria de programa, número de temporizadores, interrupciones, etc.):

Antes de seleccionar un microcontrolador es imprescindible analizar los requisitos de la aplicación:

- **Procesamiento de datos**

  Puede ser necesario que el microcontrolador realice cálculos críticos en un tiempo limitado. En ese caso debemos asegurarnos de seleccionar un dispositivo suficientemente rápido para ello por lo cual debe tener una arquitectura Hardvard.

- **Entrada Salida**

  Para determinar las necesidades de Entrada/Salida del sistema es conveniente dibujar un diagrama de bloques del mismo, de tal forma que sea sencillo identificar la cantidad y tipo de señales a controlar. Una vez realizado este análisis puede ser necesario añadir periféricos hardware externos o cambiar a otro microcontrolador más adecuado a ese sistema.

- **Consumo**

Algunos productos que incorporan microcontroladores están alimentados con baterías y su funcionamiento puede ser tan vital como activar una alarma antirrobo. Lo más conveniente en un caso como éste puede ser que el microcontrolador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla.

- **Memoria**

Para detectar las necesidades de memoria de nuestra aplicación debemos tener memoria no volátil como (EEPROM). Este último tipo de memoria puede ser útil para incluir información específica de la aplicación como un número de serie o parámetros de calibración.

- **Diseño de la placa**

La selección de un microcontrolador concreto condicionará el diseño de la placa de circuitos. El mejor consejo es elegir un chip del que podamos disponer todas las herramientas de desarrollo a un precio abordable, y además buena documentación. A nivel de experimentar en casa, el Intel 8051, Motorola 68HCX o Microchip PIC son una buena elección.

Indicaremos algunas características acerca de los microcontroladores seleccionados.

8051(INTEL)

El 8051, pertenece a la segunda generación de microcontroladores Intel, ha marcado muchas de las características que tienen los microcontroladores en la actualidad.

Tiene un diseño un poco raro, pero es muy potente y sencillo de programar (una vez que se conoce).Su arquitectura es Hardvard Modificada con espacio de direcciones separadas para memoria de programa y memoria de datos.

La ROM interna del 8051 y el 8052 no pueden ser programados por el usuario, debe suministrar el programa al fabricante, y el fabricante programa los microcontroladores durante la producción. Puede direccionar hasta 64k de memoria de datos externa, y solo puede acceder a ella mediante direccionamiento indirecto.

El 8051 es el primer microcontrolador de la familia introducida por Intel Corporation. La familia 8051 de microcontroladores son controladores de 8 bits capaces de direccionar hasta 64 kbytes de memoria de programa y una separada memoria de datos de 64 kbytes.

Debido a costos, la opción de la ROM programado por el fabricante no es económica para producción de pequeñas cantidades.

68HC05 (MOTOROLA)

Está basado en el antiguo 6800, tiene arquitectura Von-Neuman donde las instrucciones, datos, entrada/salida y temporizadores ocupan un mismo espacio de memoria por tal motivo son muy lentos sus procesos.

El 68hc11 de la familia Motorola, es un potente microcontrolador de 8 bits en su bus de datos, 16 bits en su bus de direcciones, con un conjunto de instrucciones que es similar a los más antiguos miembros de la familia 68xx (6801, 6805, 6809).

Tiene internamente EEPROM o OTPROM, RAM, digital I/O, timers, A/D converter, generador PWM, y canales de comunicación sincrónica y asincrónica (RS232 y SPI). La corriente típica que maneja es menor que 10ma.

PIC16F84(MICROCHIP)

Tiene una arquitectura Hardvard también consta con una estructura Risc que le permite realizar las operaciones mas rápidas que los de mas microcontroladores.

En el caso del PIC 16F84 se trata de una memoria EEPROM de 1K palabras de 14 bits cada una. El PIC 16F84 tiene la misma capacidad de memoria de instrucciones, pero de tipo flash. Ambos disponen de 64 bytes de EEPROM como memoria de datos auxiliar y opcional.

La memoria EEPROM y la Flash son eléctricamente gravables, lo que permite escribir y borrar el programa bajo prueba manteniendo el microcontrolador en el mismo zócalo y usando el mismo dispositivo para grabar y borrar. Esta característica supone una gran ventaja con la mayoría de los microcontroladores, que tienen como memoria de programa reescribible una tipo EPROM. Estas se graban eléctricamente, pero para borrarlas hay que someterlas durante cierto tiempo a rayos ultravioleta, lo que implica sacar del zócalo el circuito integrado y colocarlo en un borrador de EPROM. El hecho de utilizar una memoria flash es porque tiene mayores posibilidades de aumentar su capacidad con relación a la EEPROM. También por su mayor velocidad y menor consumo. La memoria EEPROM es capaz de soportar 1.000.000 de ciclos de escritura / borrado, frente a los 1.000 de la Flash.

Tabla acerca de los requerimientos de un Microcontrolador

Microcontroladores	**Memoria**		**I/O Pines**	**Arquitectura**	**Costo**
	RAM	**ROM**			
8051	4k	128 (ROM)	18	Hardvard	18
68HC05 C2	2k	176 (EPROM)	18	Von-Neuman	15
COP8SAC7	4K	128(EPROM)	16-40	Hardvard	12
16F8X	1k	128(EEPROM)- Flash	18- 40	Hardvard	10

Conclusión: Después de haber analizados todos los microcontroladores hemos elegido al Pic 16F84 por que reúne las necesidades requeridas por el usuario como son:

- Sus altas prestaciones y su reducido juego de instrucciones (RISC)
- Así como su estructura interna del tipo Harvard.

- La arquitectura permite un gran ahorro de instrucciones.

- Sencillez de manejo: Tienen un juego de instrucciones reducido; 35 en la gama media.
- Buena información, fácil de conseguir y económica.

- Precio: Su coste es comparativamente inferior al de sus competidores.

- Poseen una elevada velocidad de funcionamiento. Buen promedio de parámetros: velocidad, consumo, tamaño, alimentación, código compacto, etc.

- Herramientas de desarrollo fáciles y baratas. Muchas herramientas software se pueden recoger libremente a través de Internet.

- Existe una gran variedad de herramientas hardware que permiten grabar, depurar, borrar y comprobar el comportamiento de los PIC.

- Diseño rápido.

- La gran variedad de modelos de PIC permite elegir el que mejor responde a los requerimientos de la aplicación.

- Aumento de la fiabilidad: al reemplazar el microcontrolador por un elevado número de elementos disminuye el riesgo de averías y se precisan menos ajustes.

- Reducción del tamaño en el producto acabado: La integración del microcontrolador en un chip disminuye el volumen, la mano de obra y los stocks.

- Mayor flexibilidad: las características de control están programadas por lo que su modificación sólo necesita cambios en el programa de instrucciones.

Queremos constatar que para las aplicaciones más habituales (casi un 90%) la elección de una versión adecuada de PIC es la mejor solución; sin embargo, dado su carácter general, otras familias de microcontroladores son más eficaces en aplicaciones específicas, especialmente si en ellas predomina una característica concreta, que puede estar muy desarrollada en otra familia.

2.1 Definición de Microcontroladores Pic

Recibe el nombre de controlador el dispositivo que se emplea para el gobierno de uno o varios procesos. Por ejemplo, el controlador que regula el funcionamiento de un horno dispone de un sensor que mide constantemente su temperatura interna, cuando sobrepasa los límites prefijados, genera las señales adecuadas para intentar llevar a la temperatura al rango estipulado.

Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado notablemente. Hace tres décadas, los controladores se construían con componentes de lógica discreta; posteriormente se emplearon los microprocesadores, que se rodeaban con chips de memoria y E/S sobre una tarjeta de circuito impreso. En la actualidad, todos los elementos del controlador se han podido incluir en un solo chip, el cual recibe el nombre de microcontrolador.

PIC (***Programmable Integrated Circuits***) es un circuito integrado programable es decir son componentes sumamente útiles en la Electrónica de Consumo. Aún cuando son conocidos desde hace más de veinte años, existen en la actualidad nuevos tipos que cumplen con una serie de requisitos y características sumamente útiles.

Los Microcontroladores Pic son en el fondo procesadores similares a otros tipos, como por ejemplo la familia de procesadores X86, 80486, Pentium y muchos otros que usan arquitectura interna. En este tipo de arquitectura los datos y la memoria del programa se encuentran en el mismo espacio de direcciones.

En realidad un Microprocesador y un Microcontrolador no son la misma cosa. Los Pics son microcontroladores es decir, una unidad que posee en su interior al microprocesador

y a los elementos indispensables para que pueda funcionar como una minicompuntadora en un solo chip.

Realmente consiste en una sencilla pero completa computadora contenida en el corazón de un circuito integrado.

El microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador. Se dice que es "la solución en un chip" porque su reducido tamaño minimiza el número de componentes y el costo.

Un microcontrolador dispone normalmente de los siguientes componentes:

- Procesador o UCP (Unidad Central de Proceso).
- Memoria RAM para Contener los datos.
- Memoria para el programa tipo ROM/PROM/EPROM.
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, Puertas Serie y Paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema.

Un microcontrolador es la unión de tres tipos de dispositivos en un chip: un microprocesador, memorias y otros dispositivos periféricos. Evidentemente, el corazón del microcontrolador es un microprocesador.

Es decir un microcontrolador es un circuito integrado programable que contiene internamente todos los componentes de un computador. Este se utiliza para controlar el funcionamiento de una tarea determinada. Sus pines de entradas y salidas se utilizan para conectar motores, relays, actuadores, etc. Una vez que el microcontrolador esta programado, se encargara de ejecutar al pie de la letra la tarea encomendada.

Casi todos los microcontroladores actuales utilizan la estructura de Von Neuman en la que la memoria de programa contiene instrucciones y datos mezclados y se dispone de un único bus llamado bus de datos por el que circulan a la vez los códigos de las instrucciones y los datos asociados a ellas.

Esta arquitectura presenta algunos problemas cuando se demanda rapidez, con lo que es preferible utilizar la arquitectura denominada Harvard en la que instrucciones y datos están perfectamente diferenciados y se emplean buses distintos.

La arquitectura Harvard dispone de dos memorias independientes una, que contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura).

Esta estructura no modifica nada desde el punto de vista del usuario y la velocidad de ejecución de los programas es impresionante.

RISC: (*Reduced Instruction Set Computer*) Computadores de juego de instrucciones reducido, en los que el repertorio de instrucciones es muy reducido (en nuestro caso 35), las instrucciones son muy simples y suelen ejecutarse en un ciclo máquina. Además los RISC deben tener una estructura *pipeline* y ejecutar todas las instrucciones a la misma velocidad.

El PIC 16C84 es un circuito RISC con lo que ganan en velocidad de trabajo y no es preciso trabajar con un centenar de instrucciones, sino tan solo 35, sin que esto disminuya las prestaciones.

Es decir un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes o secuencias que están grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica dentro del ordenamiento del mismo y a su vez permiten obtener configuraciones diferentes. Esto se puede diferenciar según el tamaño y cantidad de sus elementos básicos.

2.2. ANÁLISIS DE LOS MICROCONTROLADORES

Los microcontroladores están conquistando el mundo. Están presentes en nuestro trabajo, en nuestra casa y en nuestra vida, en general será testigo de la conquista masiva de estos diminutos computadores, que gobernarán la mayor parte de los aparatos que fabricaremos y usamos los humanos.

Análisis y Diseño de los Microcontroladores Antiguos.

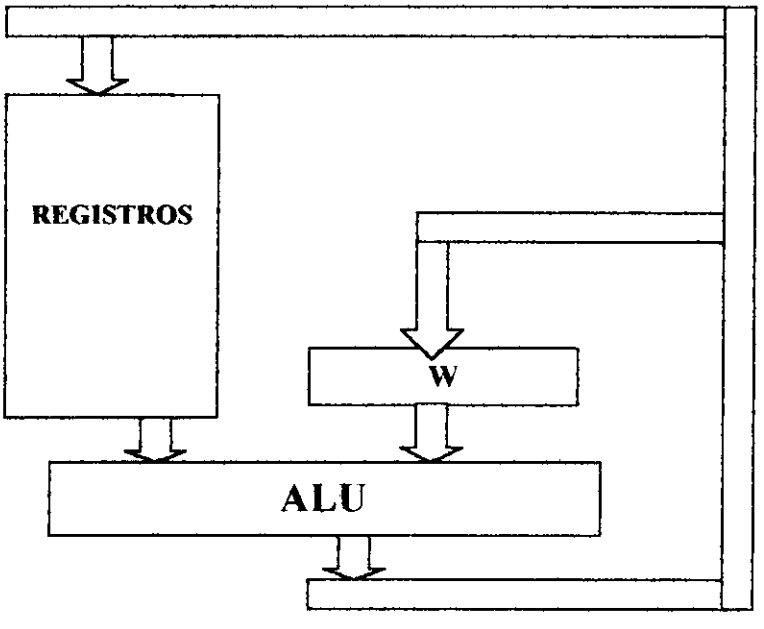


Fig2.2.1 Análisis del microcontrolador Antiguo

En los microcontroladores tradicionales todas las operaciones se realizan sobre el acumulador, la salida del acumulador esta conectada a una de las entradas de la Unidad Aritmética y Lógica (ALU), y por lo tanto éste es siempre uno de los dos operandos de cualquier instrucción. Por convención, las instrucciones de simple operando (borrar,

incrementar, decrementar, complementar), actúan sobre el acumulador. La salida de la ALU va solamente a la entrada del acumulador, por lo tanto el resultado de cualquier operación siempre quedara en este registro.

Para operar sobre un dato de memoria, luego realizar la operación siempre hay que mover el acumulador a la memoria con una instrucción adicional.

En los microcontroladores PIC, la salida de la ALU va al registro W y también a la memoria de datos, por lo tanto el resultado puede guardarse en cualquiera de los dos destinos.

En las instrucciones de doble operando, uno de los dos datos siempre debe estar en el registro W, como ocurría en el modelo tradicional con el acumulador. En las instrucciones de simple operando el dato en este caso se toma de la memoria (también por convención).

El diagrama general de un sistema microcontrolador moderno.

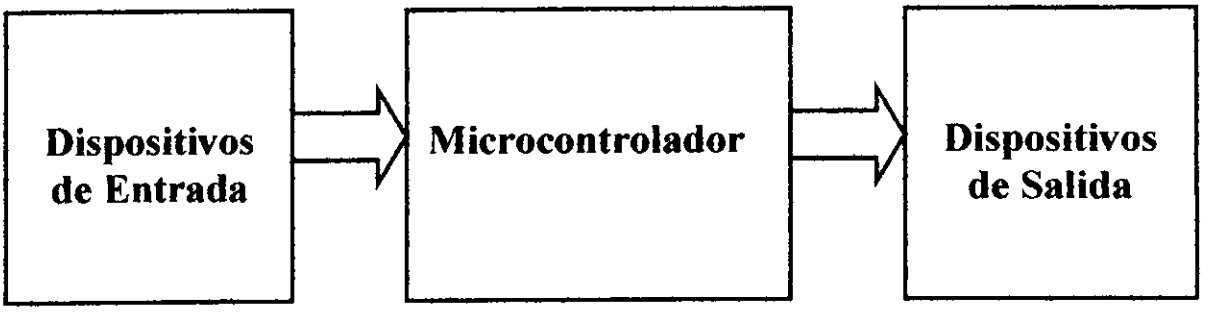


Fig2.2.2. Diagrama general de un sistema microcontrolador actual

Según el tipo de memoria de programa que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente. Existen 5 tipos de memorias no volátiles:

1. **ROM con máscara**

   Es una memoria no volátil de solo lectura cuyo contenido se graba durante la

fabricación del chip. El elevado costo del diseño de la máscara solo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

2. OTP (One Time Programming)

El microcontrolador contiene una memoria no volátil de solo lectura "programable solo una vez" por el usuario. Es al usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde una PC.

La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas. Tienen menor costo que la tecnología FLASH con la misma geometría.

3. EPROM (Erasable Programmable Read Only Memory)

Los microcontroladores que disponen de memorias EPROM (también llamada UV-PROM) pueden borrarse y grabarse muchas veces. La grabación se realiza como en el caso de las 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 caras que las OTP.

4. EEPROM (Electrical Erasable Programmable Read Only Memory)

Borrables eléctricamente. Pueden ser borradas y re-programadas sin retirarse del circuito. Es la solución más flexible a la hora de realizar un prototipo. El costo es relativamente más alto que las OTP en la misma geometría. La tensión de programación ronda los 13,5V.

5. FLASH

Similar a las EEPROM pero de mayor densidad (más capacidad) y más veloces.

Recientemente han aparecido microcontroladores con memorias FLASH con tensiones de trabajo de 2.0v a 5.5v y tensión de programación de 5v. El consumo también es mucho más bajo que las anteriores.

Existen tres orientaciones en cuanto a la arquitectura :

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 las instrucciones son simples y, 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).

Los PIC16CXX tiene rasgos especiales para reducir componentes externos; se reducen los costos, reforzando la fiabilidad del sistema y el consumo puede ser reducido. Hay

cuatro opciones del oscilador, el oscilador de RC proporciona una solución de bajo costo, el oscilador de LP minimiza el consumo de poder, el XT es un cristal normal, y el HS es para los cristales de alta velocidad.

También cuentan con la instrucción SLEEP (bajo poder), el modo Sleep ofrece economía en consumo de corriente al circuito. El usuario puede despertar el PIC a través de interrupciones externas o internas, además los PIC de esta familia proporcionan protección contra la lectura el software.

2.3 Análisis y Diseño del Microprocesador

El microprocesador es un circuito electrónico que actúa como unidad central de proceso de un ordenador, proporcionando el control de las operaciones de cálculo. Podríamos decir de él que es el cerebro del ordenador. Los microprocesadores también se utilizan en otros sistemas informáticos avanzados, como impresoras, automóviles o aviones.

El microprocesador es un tipo de circuito sumamente integrado. Los circuitos integrados, también conocidos como microchips o chips, son circuitos electrónicos complejos formados por componentes extremadamente pequeños formados en una única pieza plana de poco espesor de un material conocido como semiconductor. Los microprocesadores modernos incorporan hasta 10 millones de transistores (que actúan como amplificadores electrónicos, osciladores o, más a menudo, como conmutadores), además de otros componentes como resistencias, diodos, condensadores y conexiones, todo ello en una superficie comparable a la de un sello postal.

Un microprocesador consta de varias secciones diferentes. La unidad aritmético-lógica (ALU) efectúa cálculos con números y toma decisiones lógicas; los registros son zonas de memoria especiales para almacenar información temporalmente; la unidad de control descodifica los programas; los buses transportan información digital a través del chip y de la computadora; la memoria local se emplea para los cómputos realizados en el mismo chip.

Los microprocesadores más complejos contienen a menudo otras secciones; por ejemplo, secciones de memoria especializada denominadas memoria caché, que sirven para acelerar el acceso a los dispositivos externos de almacenamiento de datos. Los microprocesadores modernos funcionan con una anchura de bus de 64 bits (un bit es un dígito binario, una unidad de información que puede ser un uno o un cero): esto significa que pueden transmitirse simultáneamente 64 bits de datos.

Un cristal oscilante situado en el ordenador proporciona una señal de sincronización, o señal de reloj, para coordinar todas las actividades del microprocesador. La velocidad de reloj de los microprocesadores más avanzados es de unos 800 megahercios (MHz) unos 800 millones de ciclos por segundo, lo que permite ejecutar más de 2.000 millones de instrucciones cada segundo.

Los microprocesadores suelen tener dos velocidades: Velocidad interna: velocidad a la que funciona el micro internamente (500, 600, 800 MHz). Velocidad externa o de bus (FSB): velocidad con la que se comunican el micro y la placa base (generalmente 60, 66 ó 100 MHz).

Un micro consta de las siguientes partes:

- El coprocesador matemático, que realiza los cálculos matemáticos.

- La memoria caché, memoria ultrarrápida que ayuda al micro en operaciones con datos que maneja constantemente.

- El encapsulado, que lo rodea para darle consistencia, impedir su deterioro y permitir el enlace con los conectores externos.

2.4 Ventajas del microcontrolador sobre el microprocesador

Un microprocesador no es un ordenador completo. No contiene grandes cantidades de memoria ni es capaz de comunicarse con dispositivos de entrada como un teclado, un joystick o un ratón o dispositivos de salida como un monitor o una impresora.

Un tipo diferente de circuito integrado llamado microcontrolador es de hecho una computadora completa situada en un único chip, que contiene todos los elementos del microprocesador básico además de otras funciones especializadas. Los microcontroladores se emplean en videojuegos, reproductores de vídeo, automóviles y otras máquinas.

La principal ventaja de una computadora basada en microcontrolador y la que lo está en un microprocesador en un microcontrolador está formada por un solo circuito integrado, reduciendo notablemente el tamaño y el costo. Una computadora basada en un microprocasador se compone de varios circuitos integrados para soportar las memorias y los módulos de E/S. Tiene mayor tamaño y costo, y menor fiabilidad.

Si usted tuvo la oportunidad de realizar un diseño con un microprocesador pudo observar que dependiendo del circuito se requerían algunos circuitos integrados adicionales además del microprocesador como por ejemplo: memorias RAM para almacenar los datos temporalmente y memorias ROM para almacenar el programa que se encargaría del proceso del equipo, un circuito integrado para los puertos de entrada y salida y finalmente un decodificador de direcciones.

Un microcontrolador es un solo circuito integrado que contiene todos los elementos electrónicos que se utilizaban para hacer funcionar un sistema basado con un microprocesador; es decir contiene en un solo integrado la Unidad de Proceso, la memoria RAM, memoria ROM, puertos de entrada, salidas y otros periféricos.

La arquitectura permite un gran ahorro de instrucciones ya que el resultado de cualquier instrucción que opere con la memoria, ya sea de simple o doble operando, puede dejarse en la misma posición de memoria o en el registro W, según se seleccione con un bit de la misma instrucción.

Las operaciones con constantes provenientes de la memoria de programa (literales) se realizan solo sobre el registro W.

En la memoria de datos de los PIC's se encuentran ubicados casi todos los registros de control del microprocesador y sus periféricos autocontenidos, y también las posiciones de memoria de usos generales.

Estas ventajas son reconocidas inmediatamente para aquellas personas que han trabajado con los microprocesadores y después pasaron a trabajar con los microcontroladores. Estas son las diferencias mas importantes:

Por ejemplo la configuración mínima básica de un microprocesador estaba constituida por un Micro de 40 Pines, Una memoria RAM de 28 Pines, una memoria ROM de 28 Pines y un decodificador de direcciones de 18 pines; pero un microcontrolador incluye todo estos elementos en un solo Circuito Integrado por lo que implica una gran ventaja en varios factores:

En el circuito impreso por su amplia simplificación de circuitería, el costo para un sistema basado en microcontrolador es mucho menor y, lo mejor de todo, el tiempo de desarrollo de su proyecto electrónico se disminuye considerablemente.

Existen unos microcontroladores mas avanzados que otros por los componentes especiales que estos incluyen. Algunos solamente contienen puertos de entrada y de salida, otros incluyen pines hasta de 12 Bits para conversiones analógicas digitales entre otros.

Podemos mencionar algunas características especiales que poseen los microcontroladores actuales: Modulación por ancho de pulso, Comunicación Serial Sincrona, Comunicacion Serial Asincrona, Temporizadores, Contadores, etc.

Como se puede ver, existen algunas ventajas importantes cuando se realiza el diseño de un circuito utilizando un microcontrolador:

- El circuito impreso es mucho más pequeño ya que muchos de los componentes se encuentran dentro del circuito integrado.

• El costo del sistema total es mucho menor, al reducir el número de componentes.

• Los problemas de ruido que pueden afectar los sistemas con microprocesador se eliminan, debido a que todo el sistema principal se encuentra en un sólo encapsulado.

• El tiempo de desarrollo de un sistema se reduce notablemente.

2.5 Diseño con microprocesador vs. Microcontrolador

Antes de existir el microcontrolador, se utilizaban para control los sistemas con microprocesador, el cual necesitaba de varios elementos externos para llevar a cabo sus funciones.

Diseño con microprocesador

El microprocesador es un circuito integrado que contiene la Unidad Central de Proceso (UCP), también llamada procesador, de un computador. La UCP está formada por la Unidad de Control, que interpreta las instrucciones, y el Camino de Datos, que las ejecuta.

Las patitas de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que se destine.

Un microprocesador es un sistema abierto esto es, se le pueden acoplar dispositivos para obtener las características deseadas

MICROPROCESADOR

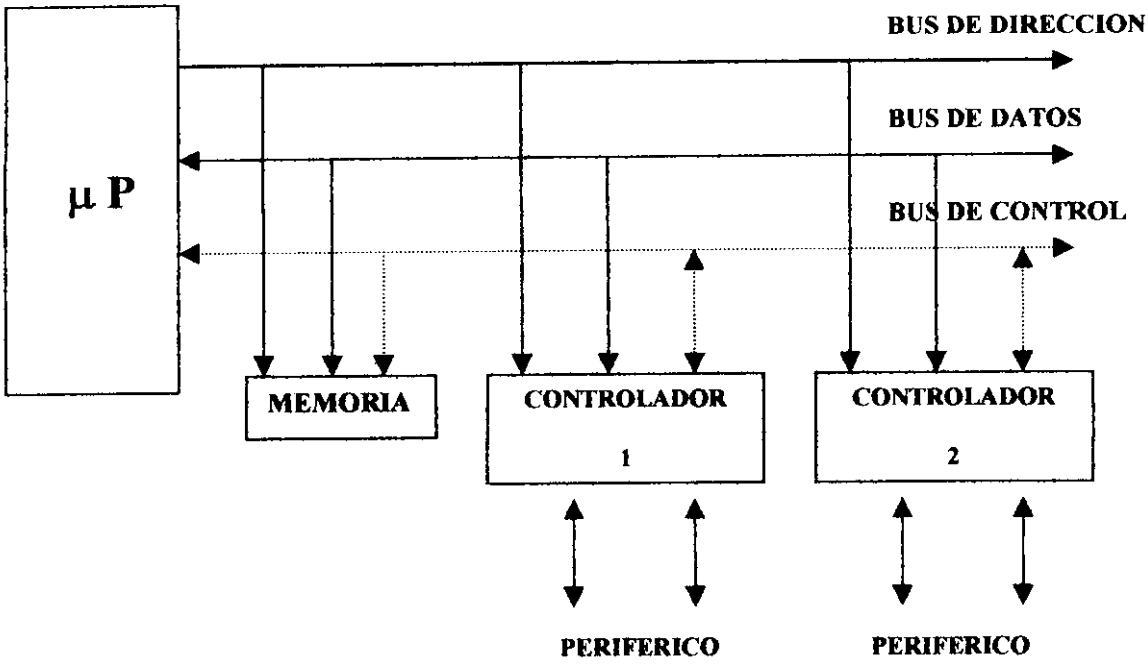


Fig2.5.1. Diagrama del Diseño Microprocesador

Arquitectura Abierta

Estos microcontroladores se caracterizan porque, además de disponer de una estructura interna determinada, pueden emplear sus líneas de E/S para sacar al exterior los buses de datos, direcciones y control, con lo que ***se posibilita la ampliación de la memoria y las E/S*** con circuitos integrados externos.

Memorias ROM (Memoria de sólo lectura). Se utilizan para almacenar el programa y no pierden la información aunque se retire la alimentación al sistema.

Memorias RAM (Memoria de acceso aleatorio). Se utilizan para guardar los datos temporales que se necesitan en la ejecución del programa. Estas memorias se conocen como memorias volátiles porque pierden la información cuando se retira la alimentación.

Decodificador de direcciones. Sirve para accesar correctamente a las memorias y

a los dispositivos periféricos del microprocesador. El proceso de diseño involucra los siguientes pasos :

1. Selección de los circuitos

2. Diseño del mapa de memoria y de entrada / salida

3. Diseño del decodificador de direcciones

4. Montaje del circuito y programación

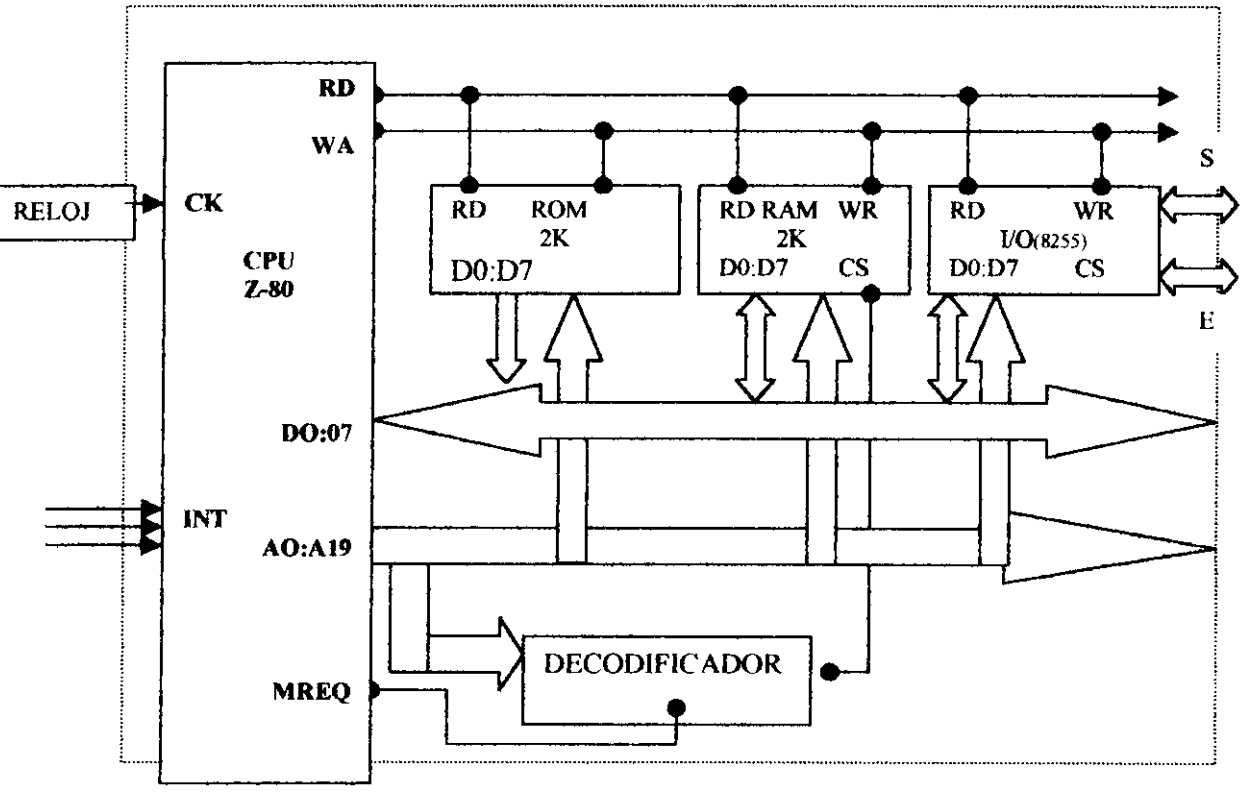


Fig2.5.2 Diseño de un Microprocesador Z-80

Si consideramos la estructura del microcomputador de la figura 2.5.2, se puede ver que cumple con los requerimientos descritos anteriormente: memorias, puertos (conexiones con el exterior), decodificador, etc.

Lo que muestra la explicación anterior es el carácter constante de la estructura de un

sistema con microprocesador en un circuito de control. No es difícil comprender entonces por qué los fabricantes de circuitos integrados decidieron producir un chip.

Un microcontrolador es un sistema cerrado que contiene un computador completo y prestaciones limitadas que no se pueden modificar.

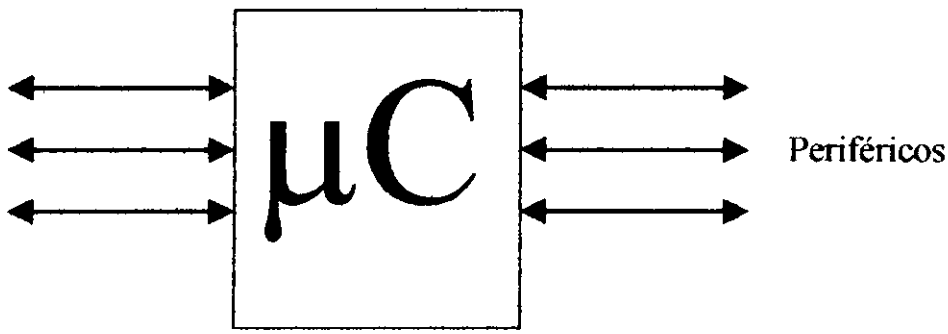


Figura 2.5.3. Sistema Cerrado

El microcontrolador es un sistema cerrado. Todas las partes del computador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.

Toda la estructura necesaria para tener un microcomputador completo. Este chip es el que se ha llamado microcontrolador.

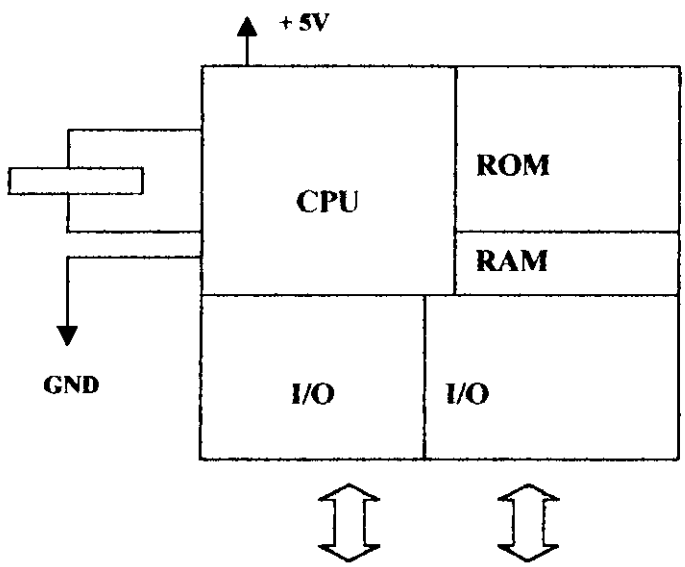


Figura 2.5.4. Idea general de un microcontrolador

Arquitectura cerrada

Cada modelo se construye con un determinado procesador (CPU), cierta capacidad de memoria de datos, cierto tipo y capacidad de memoria de instrucciones, un número de E/S y un conjunto de recursos auxiliares muy concreto. El modelo **no admite variaciones ni ampliaciones**. La aplicación a la que se destina debe encontrar en su estructura todo los que precisa y, en caso contrario, hay que desecharlo.

En la figura 2.5.4, muestra todos los componentes que se hallan dentro del microcontrolador. Ahora, para un diseñador, la idea de un microcomputador se limita al esquema mostrado en la figura 2.5.4, donde se tiene una fuente de alimentación, un circuito de reloj y el chip microcontrolador. Solamente se requiere grabar el programa en la memoria ROM; los puertos ya están listos para conectarse al mundo exterior .

2.6 ARQUITECTURA DE LOS MICROCONTROLADORES PIC

Existen dos arquitectura que utilizan los microcontroladores como son Harvard y Von Newman.

Arquitectura Harvard La arquitectura tradicional

La arquitectura tradicional de computadoras y microprocesadores se basa en el esquema propuesto por John Von Neumann, en el cual la unidad central de proceso, o CPU, esta conectada a una memoria única que contiene las instrucciones del programa y los datos fig.2.6.1.

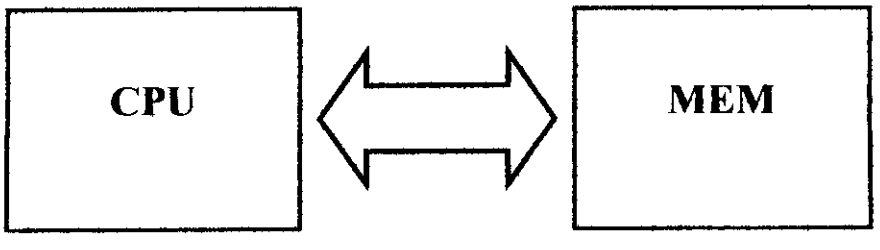


FIG. 2.6.1 Arquitectura Von Newmann

El tamaño de la unidad de datos o instrucciones esta fijado por el ancho del bus de la memoria.

Es decir que un microprocesador de 8 bits, que tiene además un bus de 8 bits que lo conecta con la memoria, deberá manejar datos e instrucciones de una o más unidades de 8 bits (bytes) de longitud. Cuando deba acceder a una instrucción o dato de más de un byte de longitud, deberá realizar más de un acceso a la memoria. Por otro lado este bus único limita la velocidad de operación del microprocesador, ya que no se puede buscar de memoria una nueva instrucción, antes de que finalicen las transferencias de datos que pudieran resultar de la instrucción anterior.

Es decir que las dos principales limitaciones de esta arquitectura tradicional son :

a) Que la longitud de las instrucciones esta limitada por la unidad de longitud de los datos, por lo tanto el microprocesador debe hacer varios accesos a memoria para buscar instrucciones complejas, que la velocidad de operación (o ancho de banda de operación) esta limitada por el efecto de cuello de botella que significa un bus único para datos e instrucciones que impide superponer ambos tiempos de acceso.

b) La arquitectura von Neumann permite el diseño de programas con código automodificable, práctica bastante usada en las antiguas computadoras que solo tenían acumulador y pocos modos de direccionamiento, pero innecesaria, en las computadoras modernas.

La arquitectura Harvard y sus ventajas:

La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que el CPU esta conectado a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos (figura 6.2.2).

Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de la memoria de programa pueden diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar.

Se puede observar claramente que las principales ventajas de esta arquitectura son:

a) que el tamaño de las instrucciones no esta relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.

b) que el tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

Una pequeña desventaja de los procesadores con arquitectura Harvard, es que deben poseer instrucciones especiales para acceder a tablas de valores constantes que pueda ser necesario incluir en los programas, ya que estas tablas se encontraran fisicamente en la memoria de programa (por ejemplo en la EPROM de un microprocesador).

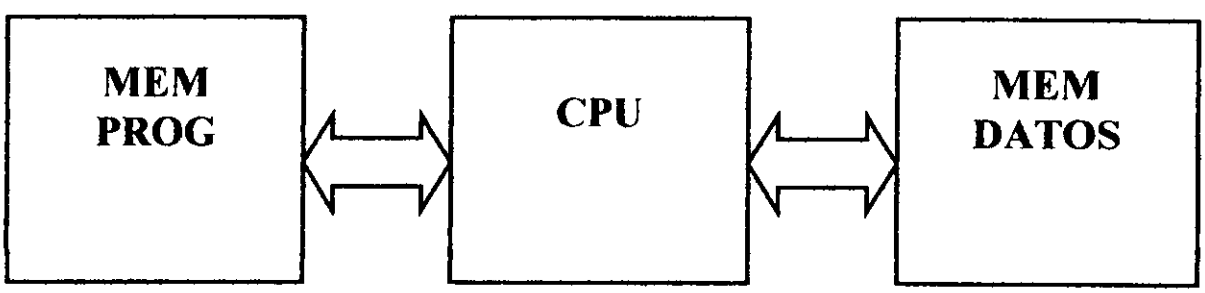


FIG. 6.2.2 Arquitectura Harvard

Los microcontroladores PIC 16C5X, 16CXX y 17CXX poseen arquitectura Harvard, con una memoria de datos de 8 bits, y una memoria de programa que, según el modelo, puede ser de 12 bits para los 16C5X, 14 bits para los 16CXX y 16 bits para los 17CXX.

2.7 Características especiales del Microcontrolador

Circuito de vigilancia (Watchdog Timer)

Su función es restablecer el programa cuando éste se ha perdido por fallas en la programación o por alguna razón externa. Es muy útil cuando se trabaja en ambientes con mucha interferencia o ruido eléctrico.

En las primeras prácticas no se utiliza para facilitar el trabajo; por eso, en el momento de programar el microcontrolador se debe seleccionar en los fusibles de configuración "watchdog timer off".

Temporizador de encendido (Power-up Timer)

Este proporciona un reset al microcontrolador en el momento de conectar la fuente de alimentación, lo que garantiza un arranque correcto. En el momento de grabar el micro se debe seleccionar el fusible de configuración "Power up Timer ON",. su tiempo de retardo es de 72 milisegundos.

Modo sleep

Está característica permite que el microcontrolador entre en un estado pasivo donde consume muy poca potencia. En este modo el oscilador se detiene. Se entra en ese estado por la ejecución de una instrucción especial (llamada sleep) y se sale de él cuando se cumple alguna de varias condiciones que se tratan más adelante.

2.8 Arquitectura interna del PIC16C84

Este término se refiere a los bloques funcionales internos que conforman el

microcontrolador y la forma en que están conectados, por ejemplo la memoria EEPROM (de programa), la memoria RAM (de datos), los puertos, la lógica de control que permite que todo el conjunto funcione, etc.

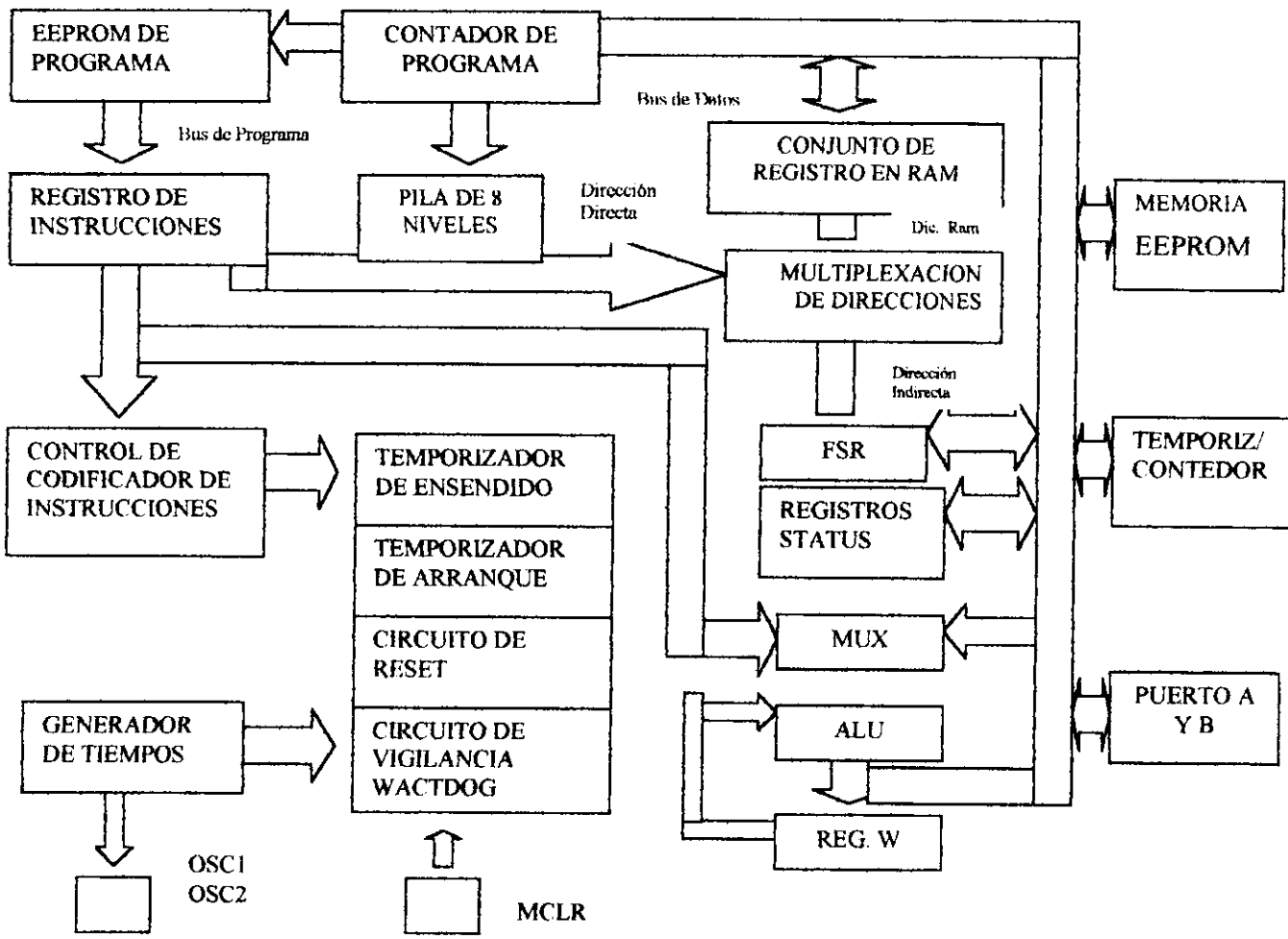


Figura 2.8.1. Arquitectura interna del PIC16C84

La figura 2.8.1 muestra la arquitectura general del PIC16C84; en ella se pueden apreciar los diferentes bloques que lo componen y la forma en que se conectan. Se muestra la conexión de los puertos, las memorias de datos y de programa, los bloques especiales como el watchdog, los temporizadores de arranque, el oscilador, etc.

Todos los elementos se conectan entre si por medio de buses. Un bus es un conjunto de líneas que transportan información entre dos o más módulos. Vale la pena destacar que el PIC16C84 tiene un bloque especial de memoria de datos de 64 bytes del tipo EEPROM, pero su uso y manejo se deja para un nivel más avanzado.

El PIC16C84 tiene dos bloques de memoria, el de programa y el de datos o registros.

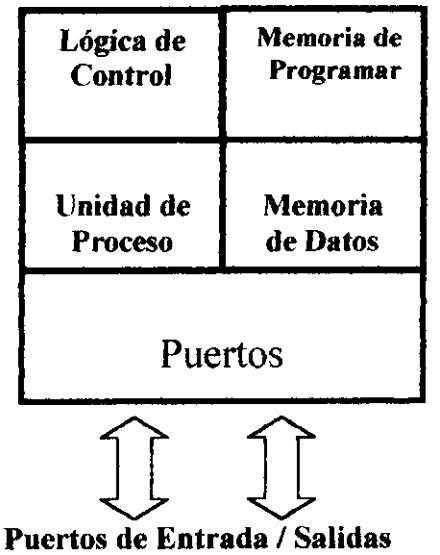


Figura 2.8.2. Arquitectura simplificada del PIC16F84

Memoria de programa.

Es una memoria de 1 Kbyte de longitud y con palabras de 14 bits. Es del tipo EEPROM y por eso puede programar y borrar eléctricamente, lo que facilita el desarrollo de los programas y la experimentación. En ella se graba, o almacena, el programa o códigos que el microcontrolador debe ejecutar, figura 2.8.3.

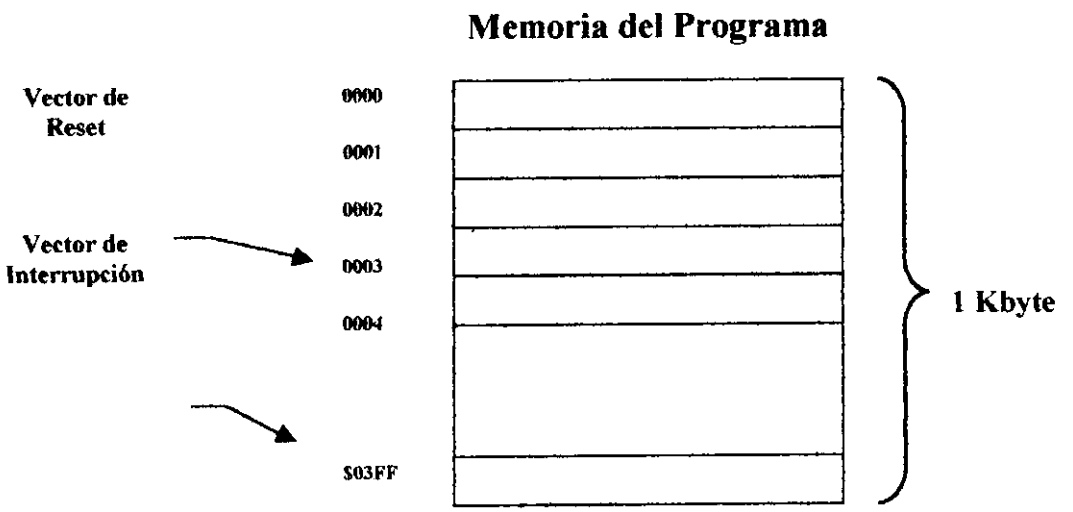


Figura 2.8.3. Memoria de programa del PIC16C84

Vector de reset. Cuando ocurre un reset el microcontrolador de programas se pone en ceros (000H). Por esta razón, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

Vector de interrupción. Cuando el microcontrolador recibe una señal de interrupción, el contador de programa apunta a la dirección 0411 de la memoria de programa, por eso, allí se debe escribir toda la programación necesaria para atender dicha interrupción.

Registros (Memoria RAM). El PIC16C84 puede direccionar 128 posiciones de memoria RAM, pero sólo tiene implementados físicamente los primeros 48 (0-2F en hexadecimal). De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 36 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que esta ejecutando.

$00	**Direccionamiento Indirecto**	R0 00H Registro de Direccionamiento
01	**TMRO**	R1 01H Registro Temporizador (TMRO)
02	**PCL**	R2 02H Contador de programa, parte baja (PCL)
03	STATUS	R3 03H Registro de estados
04	**Selector de Registro**	R4 04H Registro selector, se usa para direccionamento indirecto
05	**Datos del Puerto A**	R5 05H Puerto A, se usan los 5 bits màs bajos
06	**Datos del Puerto B**	R6 06H Puerto B, el bit 0 también puede ser de interrupción
07		R7 07H No se usa
08	**Ignore**	R8 08H No se usa todavia, mas adelante se describe
09	**Ignore**	R9 09H No se usa todavía, más adelante se describe
0A	**PCLATH**	RA 0AH Contiene los 5 bits de la parte del programa(PCLATH)
0B	**INTCON**	RB 0BH Registro del control de la interrupciones(INTCON)
0C		RC 0CH Registro de propósito general
		* * *
		* * *
$2F		R2F 2FH Registro de propósito general

Figura2.7.4. Registro del PIC16F84

Nota: Los registros 81H, 85H, y 86H son ignorados por ahora. Estos se utilizan enalgunas instrucciones de control, pero aun no es necesario utilizarlas, ya que se reemplazan por otras más sencillas. En un nivel más avanzado se tratan en detalle.

Registro de trabajo W. Este es el registro de trabajo principal, se comporta de manera similar al acumulador en los microprocesadores. Su uso lo explicaremos posteriormente.

Registro Option. Los bits de este registro se utilizan para controlar las resistencias de pull-up del puerto B, el flanco de la señal de interrupción, el valor de la preescala para el temporizador / contador o para el watchdog timer y el origen de la señal que entra al temporizador / contador, figura 2.8.5.

Por facilidad de aprendizaje del microcontrolado, se escribirá en el registro Option con una instrucción del mismo nombre (Option). Pero más adelante, en el nivel avanzado, veremos como se ha transformado su estructura y aprenderemos a usarlos de otra manera que nos permitirá tener más compatibilidad con las nuevas familias de microcontroladores de Microchip que se desarrollen.

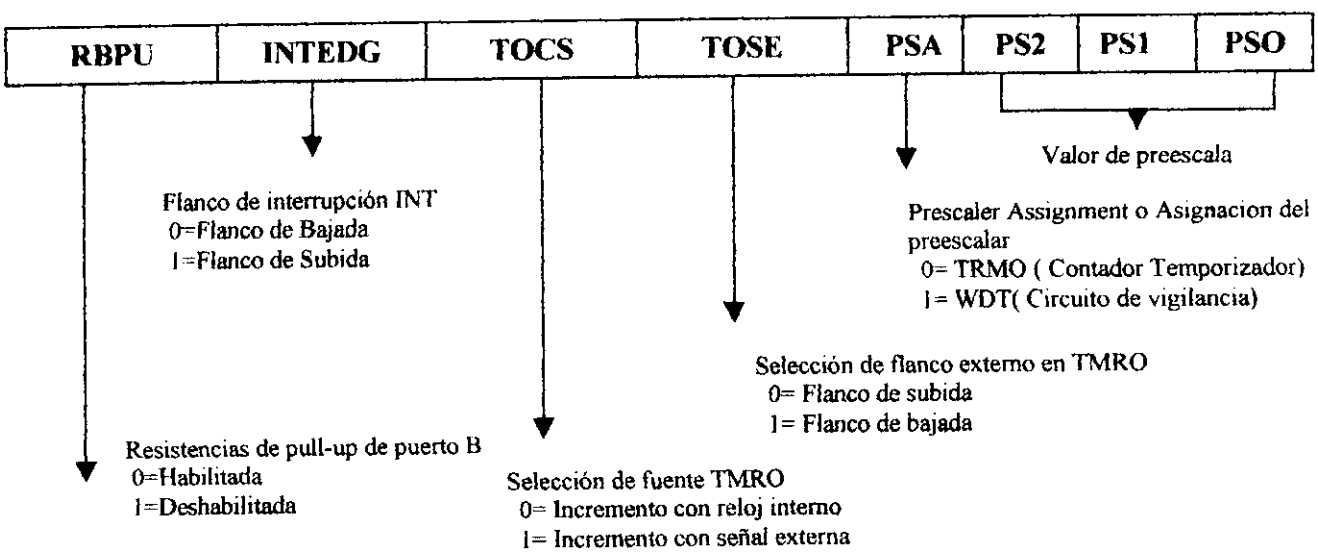


Figura 2.8.5 Registro Opción

e programa (PC). El PIC16F84 tiene un contador de programa de 13 bits suficiente para direccionar una memoria de 8K, pero solo se tiene ıtado el primer 1K. Cuando se ejecuta una instrucción GOTO o CALL se usan bits más bajos, de manera que se pueden hacer saltos en el programa y direccionar toda la memoria (1 K) completamente; en la figura 2.6.8 se muestra su contenido.

La parte alta del contador de programa (PCH) no se puede accesar directamente; ella debe cargarse desde los 5 bits más bajos del registro llamado PCLATCH. El PIC16F84 no usa el esquema de paginación de memoria que se tiene en otros microcontroladores; solamente cuando se tiene una tabla que usa el PC se debe tener cuidado de no pasar las páginas de 256 bytes, porque estas instrucciones solo afectan el contenido de la parte baja del PC, para hacer tablas si preocuparse. Por esto, se debe tener un conocimiento de paginación usando el PCLATCH, el cual se explicará posteriormente.

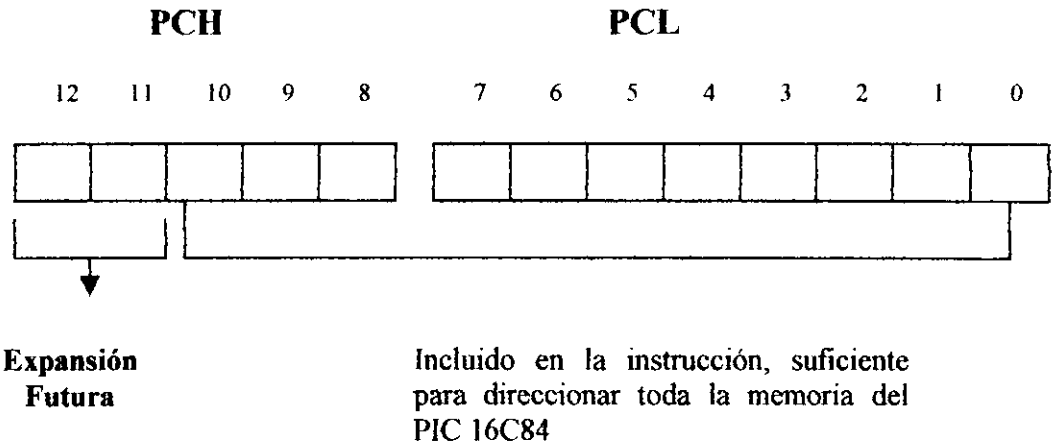


Figura 2.8.6. Contador de programa (13 bits)

Registros de estados. Es un registro donde se tiene la información el resultado que dejó la última operación efectuada; si el resultado de una operación fue positivo o negativo, si fue cero o diferente de cero, etc. El estado de sus bits se puede consultar usando las instrucciones BTFSS (puede el bit y salte si es uno) y BTFSC (pruebe el bit y salte si es cero), figura 2.8.7.

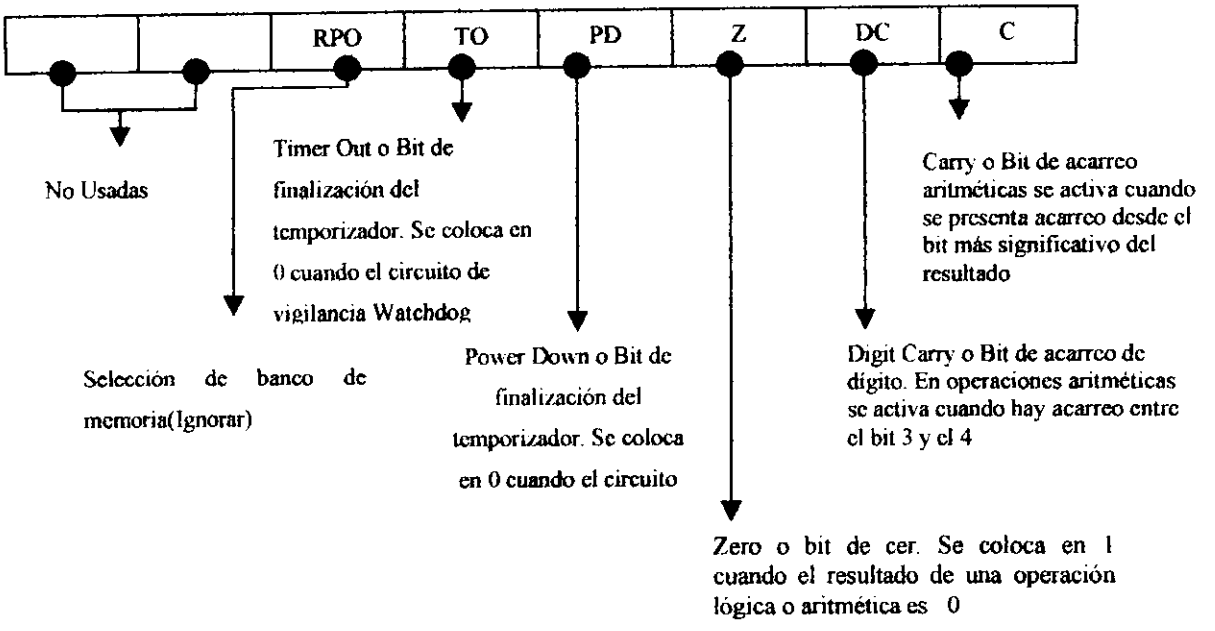


Figura 2.8.7. Registro de estados (6 bits)

Pila (stack)

Estos registros no forman parte de ningún banco de memoria y no son accesables por parte del usuario. Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina o cuando se atiende una interrupción; luego, cuando el micro regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila. El PIC16C84 tiene una pila de 8 niveles.

Componentes especiales en un microcontrolador

En muchas ocasiones, se requiere algo más que los pines de entrada y salida para controlar algún proceso, como se muestra en la figura 2.8.8. Pensando en ello los fabricantes de microcontroladores han adicionado algunos componentes especiales en algunos de sus modelos. Las posibilidades son amplísimas y el usuario puede escoger a la carta:

Algunos microcontroladores tienen conversores análogo a digital (A/D), en caso de que se requiera medir señales no digitales, como por ejemplo temperatura, voltaje, luminosidad, etc.

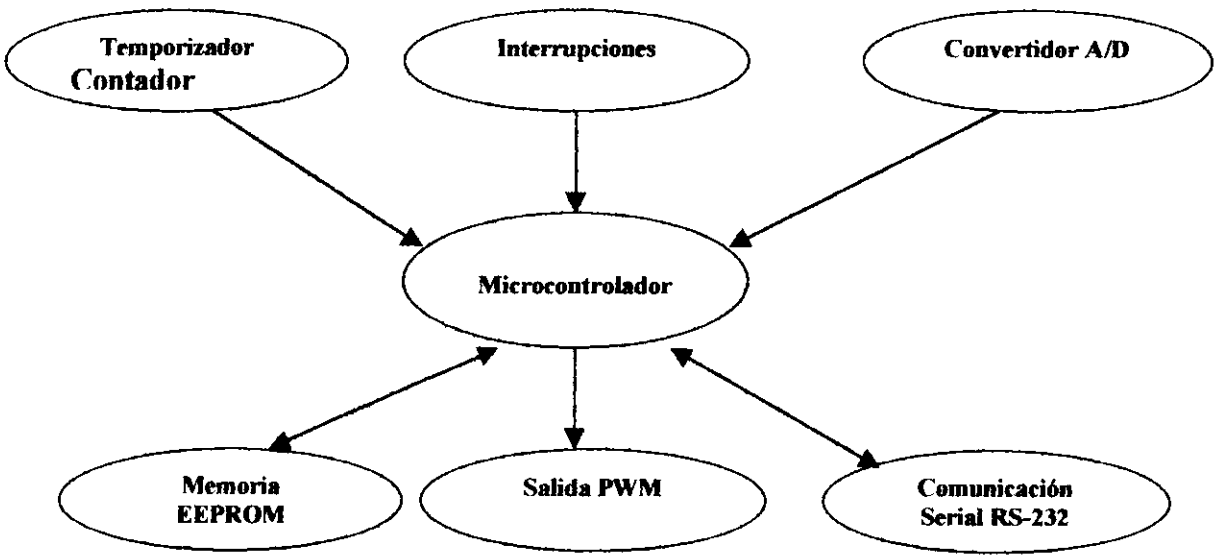


Figura 2.8.8 Funciones adicionales en los microcontroladores

En algunas amplía las capacidades 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, minimizará el costo, el hardware y el software.

Los principales recursos que incorporan los μC son:

- Timers o temporizadores
- Watchdog o perro guardián
- Brownout o protección ante fallo de alimentación
- Sleep o estado de reposo o bajo consumo
- Conversor analógico a digital
- Conversor digital a analógico

- Comparador analógico
- PWM o Modulador de anchura de pulso
- Puertas de E/S digitales
- Puertos de comunicación serial
- Protección de código

Timers

Controlan períodos de tiempo (temporizadores).

Llevan la cuenta de acontecimientos que suceden en el exterior (contadores).

Watchdog

Los programas frecuentemente pueden fallar, tanto por problemas de diseño o por ruidos externos al sistema. Por lo general, el procesador queda en un lazo infinito dejando de atender al resto del programa. La única alternativa que nos queda en estos casos es resetear el sistema.

El perro guardián o watchdog se encarga de resetar al sistema automáticamente, en el momento que el sistema quede "colgado".

Brownout

Es un circuito de protección que resetea al μC cuando la tensión de alimentación es inferior a un mínimo.

Si el μC no posee este recurso, se puede construir uno externo.

El siguiente circuito cumple esta función:

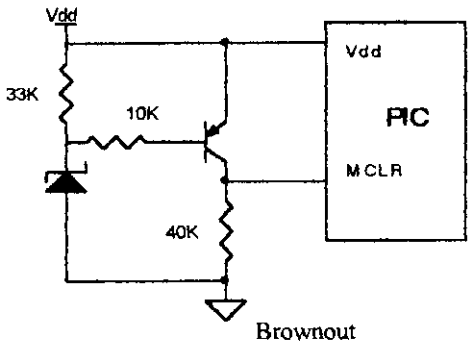


Figura 2.8.9 Diagrama del Brownout

Sleep o Bajo consumo

Son abundantes las situaciones reales de trabajo en que el μC debe esperar sin hacer nada, a que se produzca algún acontecimiento externo que lo ponga de nuevo en funcionamiento. Para ahorrar energía, los μC disponen de una instrucción especial que los pasa a este modo de reposo. En dicho estado se detiene el reloj principal y se congelan los circuitos asociados. Sale de este estado al interrumpirse por el acontecimiento esperado.

En los PIC se ingresa a este modo ejecutando la instrucción SLEEP.

Si está habilitado, el watchdog se resetea pero continúa activo y el oscilador del reloj se detiene. Los puertos de E/S mantienen su estado.

Para despertar al μC y sacarlo de este estado deberá ocurrir uno de estos eventos:

1- Un RESET externo en el pin MCLR.

2- Un RESET interno producido por el watchdog.

Ambos eventos causan un reset del dispositivo microcontrolador.

Conversor A/D y D/A

Los µC que poseen conversores, pueden manejar estas señales analógicas. Suelen disponer de un multiplexor para manejar varias entradas analógicas.

Comparador analógico

Algunos µC poseen un amplificador operacional que actúa como comparador entre una señal fija de referencia y otra variable. La salida del comparador proporciona un 0 o un 1 según la señal sea mayor o menor que la de referencia.

PWM

Son circuitos que proporcionan en su salida impulsos de ancho variable, que se ofrecen al exterior a través de las patitas del encapsulado.

Puertas digitales de E/S

Todos los µC disponen de algunas patitas de E/S digitales. Por lo general se agrupan de a 8 formando puertas. Pueden configurarse como entrada o salida cada patita independientemente de las otras.

Puertos de comunicación

Con el objeto de dotar al µC de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, redes, etc., algunos modelos disponen de estos recursos entre los que se destacan:

- UART: Adaptador de Comunicación Serie Asincrónica.
- USART: Adaptador de Comunicación Serie Sincrónica y Asincrónica.
- USB (Universal Serial Bus): Moderno bus serie para los PC.

- Bus I2C: Interfaz serie a dos hilos (Philips).

- CAN (Controller Area Network): Interfaz utilizada por automóviles.

Protección de código

El código o programa ingresado en los μC puede ser protegido contra lectura por razones de seguridad.

También posee 4 bytes destinados a identificación, donde el usuario puede colocar una palabra única de identificación. Esta palabra se puede leer durante el proceso de verificación de la grabación.

•Si se requiere medir períodos de tiempo entre eventos, generar temporizaciones o salidas con frecuencia específica, algo muy común en electrónica, se puede disponer de uno o varios temporizaciones programables (timers).

• Cuando se necesita establecer comunicación con otro microcontrolador o con un computador se puede disponer de una interface serial RS-232.

• Para desarrollar una aplicación donde los datos no se alteren a pesar de que se retire la alimentación, se puede usar un microcontrolador con memoria EEPROM, que es un tipo de memoria ROM que se puede programar o borrar eléctricamente sin necesidad de circuitos especiales.

• Para quienes utilizan salidas PWM (modulación por ancho de pulso) en el control de motores DC o cargas resistivas, existen microcontroladores que pueden ofrecer varias de ellas.

• Cuando se requiere atender eventos en tiempo real o se tienen procesos que no dan espera, se debe utilizar la técnica llamada de «Interrupciones».

Cuando una señal externa activa una línea de interrupción, el microcontrolador deja de lado la tarea que se encuentra ejecutando para atender una situación especial y luego puede regresar a continuar con la labor que estaba desempeñando.

PINES Y FUNCIONES

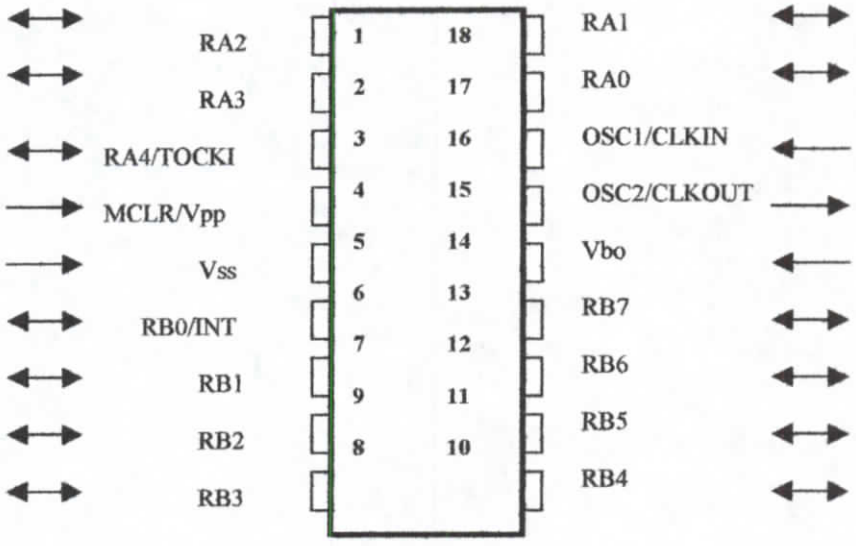


Fig. 2.8.10 Diagrama de pines del PIC16C84

El PIC16C84 es un microcontrolador de Microchip fabricado en tecnología CMOS, su consumo de potencia es muy bajo y es completamente estático: el reloj puede detenerse y los datos de la memoria no se pierden.

El encapsulado más común para el microcontrolador es el DIP (Dual Inline Pin) de 18 pines, propio para usarlo en experimentación La referencia completa es 16F84-04/P, para el dispositivo que utiliza reloj de 4 MHz. Sin embargo, hay otros tipos de encapsulado que se pueden utilizar según el diseño y la aplicación que se quiere realizar. Por ejemplo, el encapsulado tipo surface mount (montaje superficial) tiene un reducido tamaño y bajo costo, que lo hace propio para producciones en serie o en lugares de espacio muy reducido.

Puertos del microcontrolador

Los puertos son el puente entre el microcontrolador y el mundo exterior. Son líneas digitales que trabajan entre cero y cinco voltios y se pueden configurar como entradas o como salidas.

El PIC16C84 tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8 líneas, figura 2.8.10. Cada pin se puede configurar como entrada o como salida independiente. Usando una instrucción especial llamada TRIS, se puede lograr el efecto descrito. En esa instrucción un "0" configura el pin correspondiente como salida y un "1" lo configura como entrada.

El puerto B tiene internamente unas resistencias de pu11-up conectadas a sus pines (sirven para fijar el pin a un nivel de cinco voltios), su uso puede ser habilitado o deshabilitado bajo control del programa. Todas las resistencias de pullup se conectan o se desconectan a la vez, usando el bit llamado RBPU que se encuentra en el registro (posición de memoria RAM) llamado OPTION. La resistencia de pu11-up es desconectada automáticamente en un pin si este se programa como salida. Cuando ocurre un reset sé deshabilitan todas las resistencias de pull-up.

El pin RA4/TOCKI del puerto A puede ser configurado como un pin de entrada/ salida o como temporizador / contador. Cuando este pin se programa como entrada digital, funciona como un disparador de Schimitt (Schimitt trigger), puede reconocer señales un poco distorsionadas y llevarlas a niveles logicos (cero y cinco voltios). Cuando se usa como salida digital se comporta como colector abierto; por lo tanto se debe poner una resistencia de pu11-up (resistencia externa conectada a un nivel de cinco voltios).

Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega a la salida un " 1 " lógico. Este pin como salida no puede manejar cargas como fuente sólo en el modo sumidero.

Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectados a alguna parte, nunca dejarlos al aire porque se puede dañar el integrado. Los pines que no se estén usando se deben conectar a la fuente de alimentación de +5V, como se muestra en la figura 2.8.10.

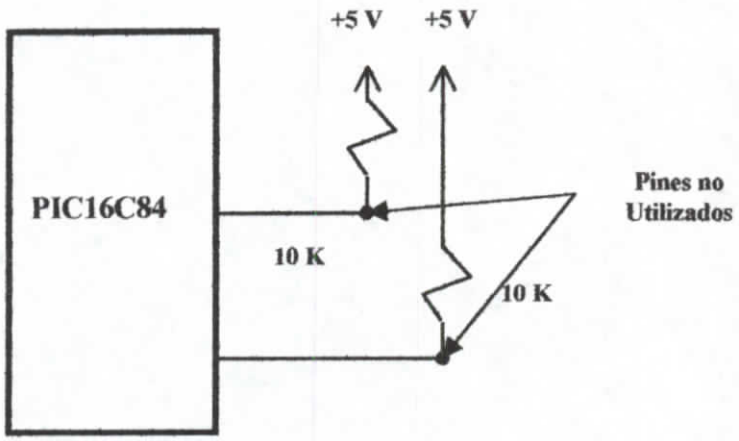


Fig. 2.8.11 Los puertos no utilizados se deben conectar a la fuente

La máxima capacidad de corriente de cada uno de los pines de los puertos en modo sumidero (sink) es de 25 mA y en modo fuente (source) es de 20 mA, Tabla 2.6.1. La máxima capacidad de corriente total de los puertos es:

Tabla de Corriente de los puertos

	PUERTO A	**PUERTO B**
Modo sumidero	80 mA	150 mA
Modo fuente	50 mA	100 mA

El consumo de corriente del microcontrolador para su funcionamiento depende del voltaje de operación, la frecuencia y de las cargas que tengan sus pines. Para un reloj de 4 MHz el consumo es aproximadamente 2 mA; aunque este se puede reducir a 40 microamperios cuando se está en el modo sleep (en este modo el micro se detiene y disminuye el consumo de potencia. Se sale de ese estado cuando se procede alguna condición especial que veremos más adelante).

El oscilador externo

Todo microcontrolador requiere un circuito externo que le indique la velocidad a la que debe trabajar. Este circuito, que se conoce como oscilador o reloj, es muy simple pero de vital importancia para el buen funcionamiento del sistema. El PIC16C84 puede utilizar cuatro tipos de reloj diferentes. Estos tipos son:

- RC. Oscilador con resistencia y condensador.
- XT. Cristal.
- HS. Cristal de alta velocidad.
- LP. Cristal para baja frecuencia y bajo consumo de potencia.

En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados "fusibles de configuración".

El tipo de oscilador que se sugiere para las prácticas es el cristal de 4 MHz, porque garantiza mayor precisión y un buen arranque del microcontrolador. Internamente esta frecuencia es dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se ejecuta en un microsegundo. El cristal debe ir acompañado de dos condensadores y se conecta como se muestra en la figura 2.6.14.

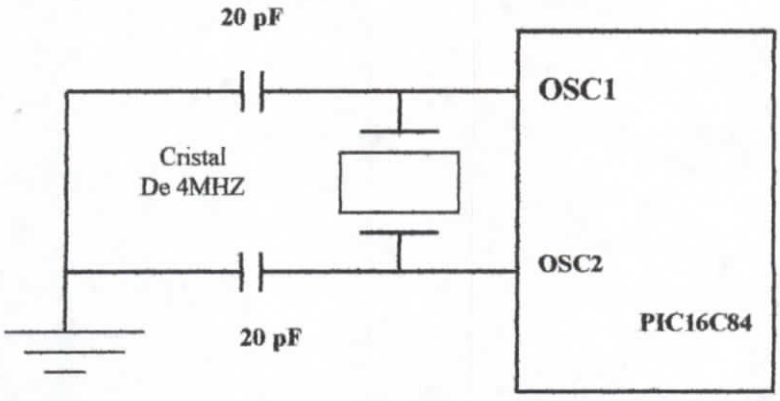


Figura 2.8.12. Conexión de un oscilador a cristal

Dependiendo de la aplicación, se pueden utilizar cristales de otras frecuencias; por ejemplo se usa el cristal de 3,579545 MHz porque es muy económico el de 32,768 KHz cuando se necesita crear bases de tiempo de un segundo muy precisas.El limite de velocidad en estos microcontroladores es de 10 MHz.

Si no se requiere mucha precisión en el oscilador y se quiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra en la figura 2.8.13.

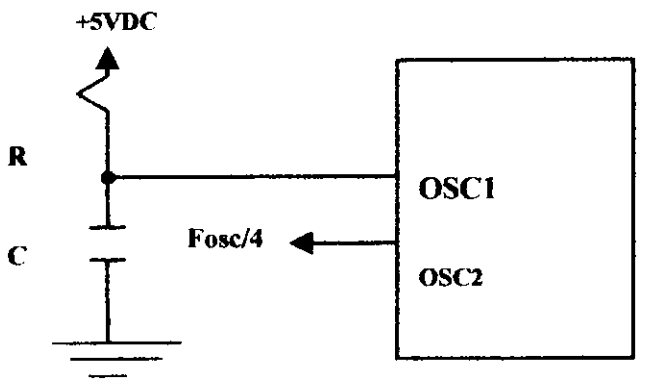


Figura 2.8.13.Conexión de un oscilador a cristal.

Reset

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento de] sistema cuando sea necesario, ya sea por una falla que se presente o porque así se halla diseñado el sistema. El pin de reset en los PIC es llamado MCLR (master clear).

El PIC 16C84 posee internamente un circuito temporizador conectado al pin de reset que funciona cuando se da alimentación al micro, se puede entonces conectar el pin de MCLR a la fuente de alimentación.

Esto hace que al encender el sistema el microcontrolador quede en estado de reset por un tiempo mientras se estabilizan todas las señales del circuito y así garantizar un buen arranque. Cuando se quiere tener control sobre el reset del sistema se puede conectar un botón como se muestra en la figura 2.8.14.

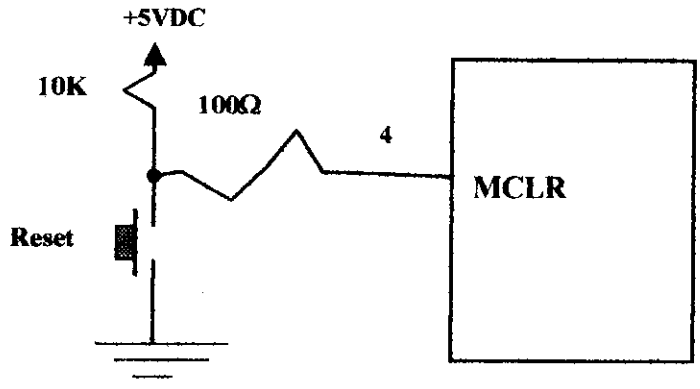


Figura 2.8.14. Conexión del botón de reset

2.9 Clases de lenguajes para programar

En la actualidad existen varias clases de software para programar al Pic, en este punto hablaremos acerca de la ventajas y desventaja de los software para programar.

MPLAB

En el Website de Microchip podrá encontrar hojas de datos, notas de aplicación y, lo mejor de todo el MPLAB, un programa de desarrollo para simular y verificar programas Pic el mismo está diseñado para ser ejecutado bajo Windows.

El software MPLAB le permitirá editar programas en lenguaje Assembly (también llamados códigos de fuente), compilarlos en un códigos de objeto y luego analizar el código binario resultante para ver qué acción realizará el microcontrolador.

De este modo, podrá localizar los errores lógicos en su programación antes de ejecutar cualquier código en el hardware.

Prepárese para recibir mensajes de error cuando compile el programa. El MPASM se quejará moderadamente de que no utilizará la instrucción TRIS. Microchip le ha restado importancia a esta instrucción y algunos procesadores PIC del futuro podrían no registrarla.

Desde la perspectiva del software, uno de los mejores rasgos de PIC es su llamada compatibilidad de código de fuente. Si su diseño supera los recursos de chips con los cuales inició el programa, podrá utilizar otro chip con mayores recursos sin necesidad de reescribir el programa desde el principio.

Para nuestro propósitos, la utilización de la instrucción TRIS en el 16C84, 16F84 y 16F83 trabaja de modo aceptable. Además, la vía alternativa de configurar el Puerto B para la salida es mucho más complicada.

NOPPP

Otro programa más sencillo para " programar " nuestro PIC se denomina NOPPP y puede ser fácil obtenerlo.

Este programa MS-DOS se ejecuta bajo Windows 3.x o Windows 95. De cualquier modo, si ejecuta el programa bajo Windows 3.1, trabajará mejor si lo ejecuta bajo la aplicación de "pantalla completa".

El timing es esencia para los pulsos de programación y las aplicaciones DOS de pantalla completa toman el control total de la computadora.

Si por alguna razón tiene dificultades en ejecutar el programa NOPPP bajo Windows 3.1, intente salir a un entorno DOS y ejecútelo desde allí. También puede ejecutar el programa bajo OS/2; si lo hace , asegúrese de configurar el HW_TIMER en "on" en las configuraciones DOS para el programa.

Para hacer la programación, el primer paso es conectar el Programador PIC al port de la impresora de la PC e iniciar el programa NOPPP sin ninguna tensión conectada al programador. Si la línea de 5V está conectada a la tierra, el software no podrá detectar el diodo D1, y asumirá que el programador no está conectado al port de la impresora.

El menú de opciones es auto-explicativo. En general, debería cargar un archivo de código de objeto (con una extensión .HEX en el nombre de archivo) en la memoria,

seleccionar el tipo de PIC que programará, programar el software y luego verificar que el código fue correctamente programado en el chip.

Una Advertencias:

Nunca inserte o remueva un PIC de un programador mientras la tensión enviada al programador se encuentre activada. Cuando programe un Pic, el software le indicará qué acciones debe ejecutar y cuándo realizarlos. Dado que el software del programador requiere un timing engañoso, fue escrito para ejecutarse como un programa DOS.

Recuerde que los pulsos de medición para la programación del PIC tiene que durar por lo menos 0,1 µs. En la práctica, son lo suficiente largos como para superar cualquier señal de "rebotes" en los cables.

De todos modos, no deben ser demasiado largos, para que no tornen extremadamente lento el proceso de programación. También es importante que el tiempo del pulso no dependa de la velocidad de la CPU de la computadora. El software fue escrito especialmente para ser ejecutado en cualquier IBM compatible, desde una XT a las últimas Pentium.

Cabe aclarar una vez más que para programar o leer in PIC se debe tener la correspondiente hoja de datos, para saber en qué pata se introducen los datos y cómo se realiza el proceso de programación, también es necesario conocer el set de instrucciones del microcontrolador y saber "pasar al papel" las ideas que uno tenga en la mente para realizar las diferentes programaciones.

PICBASIC PRO COMPILER 2.3

El PicBasic Pro posee todos aquellas variantes que nuestros clientes nos han solicitado para la programación profesional de PICs en lenguaje BASIC, incorporando especialmente los comandos utilizados en las BASIC Stamp II, y utilizando los pines de los PORTA, C, D, E, como así también el PORTB, y la posibilidad de utilizar mas variables y mayor espacio de programa.

Algunas de sus características:

Con el, podrá desarrollar sistemas basados en PIC con el mismo lenguaje utilizado para las microcomputadoras BSII, conteniendo la gran mayoría de las funciones de las BSI y BSII. Siendo el PicBasic un compilador, podrá desarrollar sistemas mas rápidos y de mayor longitud que para las BS.

El PBP no posee el mismo nivel de compatibilidad que poseía el Picbasic con la BSI. Uno de los cambios principales es el de la incorporación de sentencias de comparación mas reales como el IF .. THEN .. ELSE .. ENDIF , en vez del IF..THEN (GOTO) de las BS.

Por default PBP genera ejecutables para ser cargados en un PIC16F84-04/P a 4Mhz de reloj. Solo se requieren unos pocos componentes extra para poner el sistema en marcha: 2 capacitores de 22pf para el cristal de 4Mhz, y un resistor de 4.7K entre VCC y el pin /MCLR. Cualquier microcontrolador pic puede ser utilizado con el PicBasic Pro.

Microcontroladores PIC soportados

El PBP genera código que puede ser ejecutado por una amplia gama de microcontroladores PIC de 8 a 68 pines y con varios recursos internos como ser canales A/D, timers, y puertos serie.

Hay algunos microcontroladores que no podrán ser utilizados con el PBP, como por ejemplo la línea PIC16C5x incluyendo 16C54 y 58. Estos PIC trabajan con instrucciones de 12bits. El PBP requiere que el micro soporte instrucciones de 14 bits y 8 niveles de stack.

Hay una gran variedad de PICs pin compatible con los 16C5x, que pueden ser utilizados.

Actualmente la lista de microcontroladores incluye: PIC16C554, 558, 620, 621, 622, 63, 64, 64A, 71, 710, 711, 715, 72, 73, 73A, 74, 84, 923, 924, el PIC 16F83 y 84,

16F877/74/73, el PIC12C671 y 672 y el PIC14000. Ahora tambien: 18C242, 18C252, 18C442 y 18C452.

Para reemplazo directo del 16C54 o 58, pueden utilizarse los PIC12C554, 558, 620 y 622 los cuales solo tienen una pequeña diferencia en precio.

En el caso de los PICs 16F84 y C84 posee 64 bytes de memoria de datos no volátil, que puede ser utilizado por el usuario para el almacenamiento de variables y parámetros del sistema, cuando el sistema sea desenergizado. las direcciones de esta memoria serán muy fácilmente accesibles con PBP a través de las instrucciones READ y WRITE.

Ahora con nuevas funciones SERIN2 y SEROUT2, mas flexibles, para comunicación serie. Con control de transmisión, paridad, mas formatos de datos y velocidades de transmisión. También nuevas funciones HSERIN y HSEROUT que manejan interfases incluidas en el PIC.

Librerías de BASIC Stamp I & II - Salto de pagina automático 2K - Acceso directo a pines y registros internos - Real If..Then..Else..Endif - Soporta jerarquías en las expresiones (importante en ecuaciones matemáticas) - Soporta interrupciones en BASIC & Assembler - Soporta Built-in LCD - Soporte de osciladores de 3.58Mhz a 20Mhz - Compatible con MPASM/ICE

Tipos de variables:

- BIT
- BYTE
- BIT ARRAYS
- BYTE ARRAYS
- WORD ARRAYS

2.10 Análisis de los lenguajes para los Microcontroladores.

En la actualidad es muy fácil construir productos electrónicos con bajo costo, espacio reducido y con características multifuncionales; de hecho, son muchos los circuitos decodificadores de señales de TV construidos con pequeños controladores programas integrados que contiene programa sencillos. En este capítulo, explicaremos cómo se programa un PIC con la ayuda de una computadora, sin que para ello se necesiten complicados circuitos adicionales.

Afortunadamente, en los últimos años, diseñar un producto con esos atributos se ha tornado más fácil gracias al desarrollo de dispositivos programables como la familia de microcontroladores PIC de Tecnología Microchip.

Nuestro objetivo es que pueda programar un PIC con ayuda de una computadora, sin que para ello se necesiten complicados circuito adicional.

El costo de un dispositivos suele ser muy alto, razón por la cual el lector o usuario le " huye" al diseño de sistemas electrónicos con estos componentes.

Desventaja De Los Programas Picbasic Pro Compiler Y Noppp

El lenguaje Picbasic Pro Compiler Y Noppp no resuelve todos los problemas de programación. Uno de ellos es la tremenda diferencia entre el set de instrucciones.

No existe mucha información sobre estos programas y difícil de conseguir el software.

Hay algunos microcontroladores que no podrán ser utilizados por estos programas.
No chequea si un valor analógico leído se excedió de un cierto umbral.

También no buscar y reaccionar ante un comando particular de una consola o teletipo.

Ventaja del programa MPLAB

- Abrir otras ventanas para chequear el seguimiento de errores

- El editor de archivo del programador con el MPLAB tiene un conjunto de características para la escritura y edición del código fuente.

- 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.

- Los mas importante el software es gratis.

- Existe bastante información sobre el software en Internet.

2.11 Descripción del Lenguaje a Utilizar

El MPLAB es un software que junto con un emulador y un programador de los múltiples que existen en el mercado, forman un conjunto de herramientas de desarrollo muy completo para el trabajo y/o el diseño con los microcontroladores PIC desarrollados y fabricados por la empresa Arizona Microchip Technology (AMT).

El MPLAB incorpora todas las utilidades necesarias para la realización de cualquier proyecto y, para los que no dispongan de un emulador, el programa permite editar el archivo fuente en lenguaje ensamblador de nuestro proyecto, además de ensamblarlo y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían de forma real tanto sus registros internos, la memoria RAM y/o EEPROM de usuario como la memoria de programa, según se fueran ejecutando las

instrucciones. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

En las siguientes líneas se pretende ayudar a todos aquellos que se enfrentan por primera vez a este programa, tanto en su instalación como en la utilización de esta potente herramienta que nos proporciona Arizona Microchip Tecnology.

De las dos versiones, nosotros vamos a centrarnos en la V.12.00, por ser esta la que menos recursos de software y harware necesita para trabajar con ella, además está pensada para trabajar con las herramientas de desarrollo MPLAB-ICD y el PICSTART que se encuentras ya muy difundidas, mientras que la versión V.99.07 está pensada para trabajar con el MPLAB-ICE 2000 soportado en NT, esta versión presenta algunas modificaciones en las ventanas de configuración del programa respecto a la anterior, además para su correcto funcionamiento es necesario disponer de la gama alta de los sistemas operativos que se indican en el siguiente apartado además del hardware más potente.

INSTALACIÓN DEL PROGRAMA

Los requerimientos mínimos para la instalación de los programas son:

- Procesador 386, 486 o Pentium*
- Windows 3.1/ 95/ 98, Windows NT 3.51/4.0, Windows 2000 ,MACOS 7.0, o Unix compatible OS
- 16 MB de memoria RAM para sistema con Windows 95.
- 24 MB de RAM para Windows NT systems.
- 32 MB para sistemas con Windows 2000.
- Unidad de CD-ROM.

- Navegador (3.0 HTML) Se recomienda por AMT:
- Procesador Pentium
- 32 MB de memoria RAM El CD-ROM de Microchip requiere para su navegación de un programa *HTML*.

Estos programas pueden obtenerse gratuitamente en las correspondientes web:

COMO EMPEZAR.

Cuando se pulsa el icono del MPLAB aparece una pantalla como la que se muestra en la Figura 2.11.1.

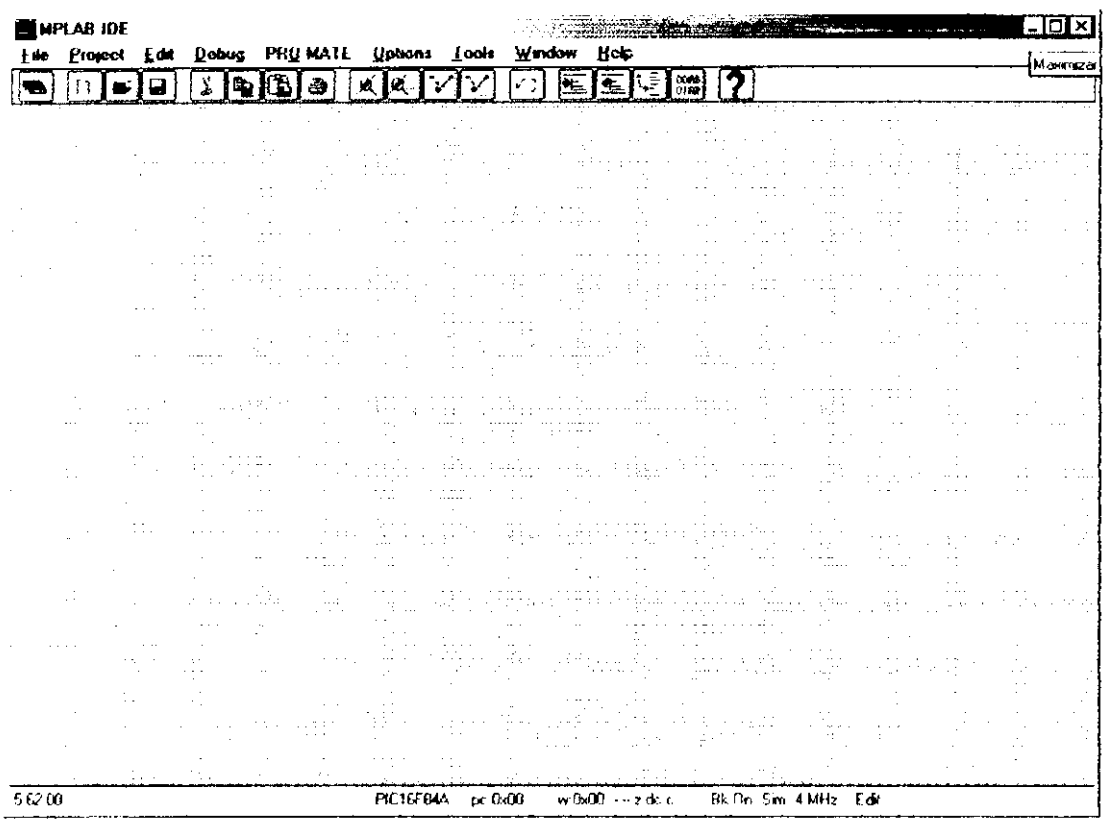


Figura 2.11.1.- Escritorio del MPLAB

Lo primero que haremos es seleccionar el modo de trabajo como simulador y el tipo de microcontrolador con el que queremos trabajar. Para ello se selecciona el botón de *Options* de la barra del control que aparece en el escritorio y del menú desplegable la opción *Developement Mode*, con lo que aparece la pantalla de la Figura 2.11.2 en la que se activa el modo *MPLAB-SIM simulator* y el microcontrolador con el que se desea trabajar, que en nuestro caso será el *PIC16F84*, por último, pulsamos el botón de *Reset* para aceptar los cambios.

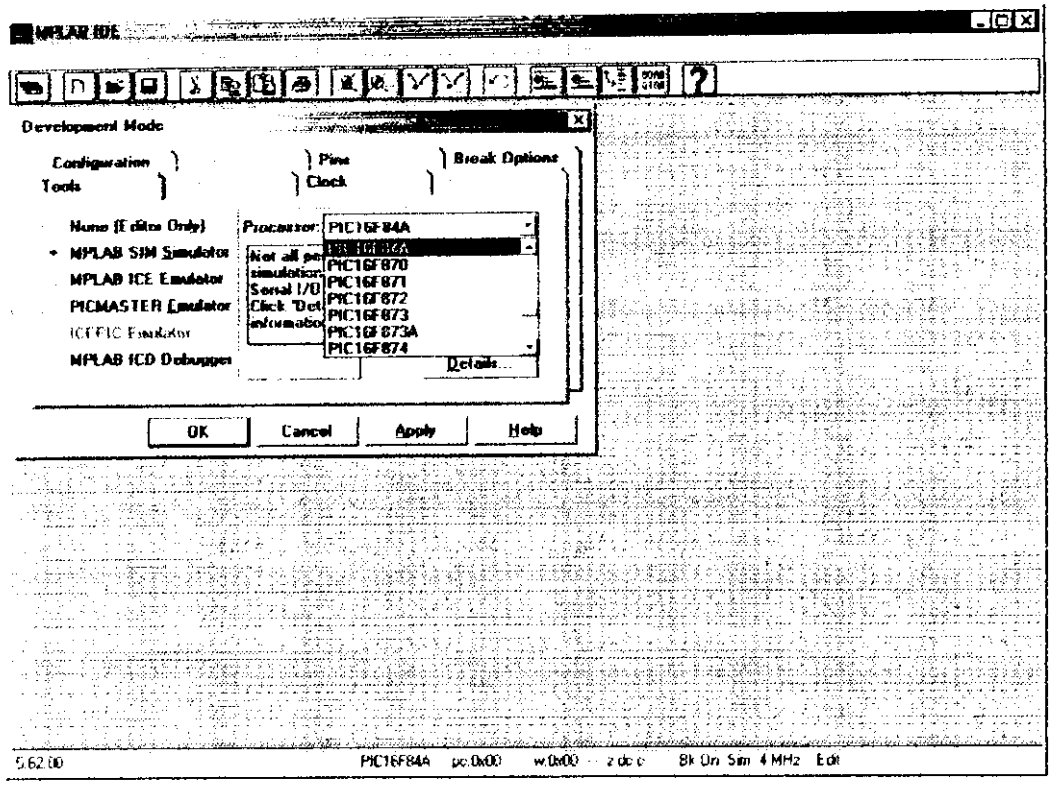


Figura 2.11.2 Selección de la opción de trabajo como simulador y el tipo de microcontrolador

Los iconos que aparecen en la barra de herramientas, son funciones que se encuentran incluidas en el menú de control, pero como en todos los programas de Windows se incluyen para manejar de forma más cómoda el programa.

Seguidamente comentaremosque significa cada uno de los iconos de la barra de herramientas que aparece en esta pantalla, mas adelante veremos que hay más barras de herramientas que pueden ser conmutadas .

Figura 2.11.3.- Barra de herramientas de edición

NUESTRO PRIMER PROYECTO

Bueno, pues ya estamos en condiciones de crear nuestro primer proyecto, para ello comenzamos por activar en el menú de control la opción *File - New* o bien activamos el icono de *crear nuevo documento* en la barra de herramientas. El programa contestará con el cuadro de diálogo de la Figura 2.11.4.

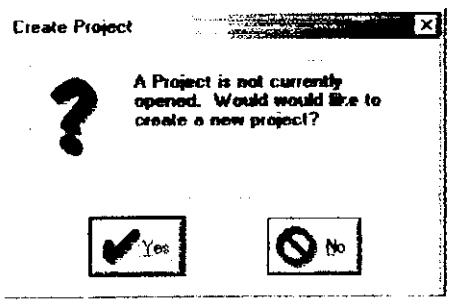


Figura 2.11.4.- No hay ningún proyecto abierto

¿Quiere crear un nuevo proyecto?

Activamos el botón de *Yes* y aparece un cuadro de dialogo como el de la Figura 2.11.5 en el que se nos pide el nombre delproyecto que tendrá extensión **.pjt* , como este es nuestro primer proyecto lellamaremos ejer1.pjt y lo guardaremos en la carpeta de trabajo que habíamoscreado anteriormente.

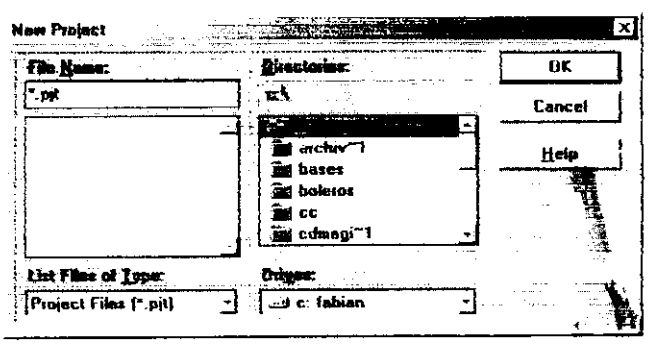


Figura 2.11.5.- Creación de un nuevo proyecto

El programa devuelve el cuadro de diálogo de la Figura 2.11.6

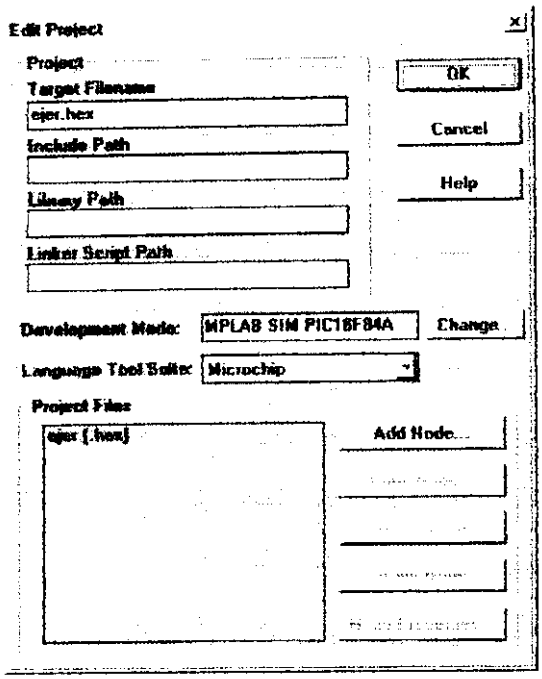


Figura 2.11.6.- Propiedades de edición del proyecto

Activamos el botón de *OK* y estamos en condiciones de empezar a escribir nuestro primer proyecto al aparecer una pantalla como la de la Figura 2.11.7

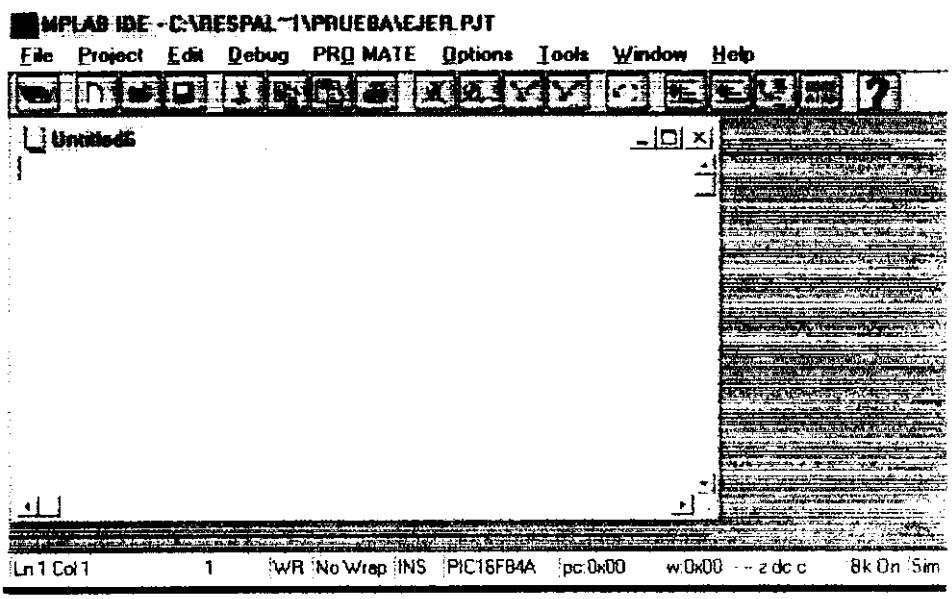


Figura 2.11.7.- Apertura del documento para comenzar a escribir nuestro proyecto

Como ensamblar un programa

Indicaremos paso a paso como ensamblar un programa con la ayuda de las herramientas del software Mplab para poder ensamblar.

EL EDITOR

Comencemos por lo tanto a escribir en lenguaje ensamblador nuestro primer programa que llamaremos ejer1.asm y que se muestra en la Figura 2.11.8. El programa realiza la suma en binario de dos números (7+8 =15) y para escribirlo usamos el editor de textos. La extensión **.asm* es la que deben llevar todos los programas escritos en ensamblador.

Deberemos de tener en cuenta que la primera columna del editor está reservada para las etiquetas que son expresiones alfanuméricas escogidas por el usuario que definen valores de posiciones de memoria. Estas deben empezar siempre por una letra. Además se debe de tener en cuenta que no pueden usarse expresiones que ya utiliza el ensamblador tales como:

- Instrucciones
- Directivas del propio ensamblador
- Nombres de registros especiales (SFR)

Nombre de cada uno de los bit de los registros especiales. En las siguientes columnas, se puede comenzar a escribir el nemónico de la instrucción o las directivas del ensamblador.

Por último hay que decir que se pueden y se deben añadir comentarios que son elementos indispensables en muchos casos para seguir el razonamiento de los programas sin perderse, para ello cuando el MPLAB encuentra un ";"(punto y coma) no se genera código máquina.

En todos estos campos los espacios en blanco no son significativos y las líneas en blanco tampoco.

Para una mejor legibilidad del programa, se recomienda acceder a cada campo utilizando el tabulador.

El uso de mayúsculas y minúsculas en los programas obedece a una serie de reglas o normas de estilo, comunes entre los programadores en ensamblador, que si bien no son obligatorias, facilitan la lectura del código fuente. Estas reglas son:

- Las directivas del ensamblador se escriben en mayúsculas
- Los nombres de las variables se escriben en mayúsculas.
- Los nemónicos de las instrucciones se escriben en minúsculas.

El programa se escribe utilizando los tabuladores para definir las distintas columnas, tales como etiquetas, comienzo de líneas de programa y columna donde empiezan los omentarios separados por un ";" (punto y coma).

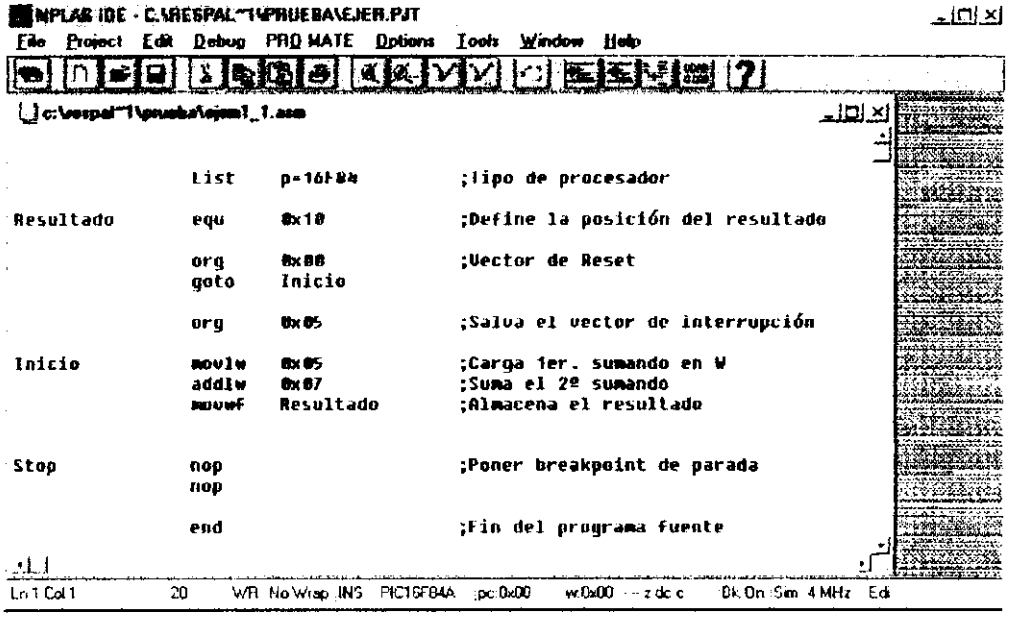


Figura 2.11.8.- Nuestro primer programa ejer1.asm

Cuando terminemos de escribir el programa seleccionamos *File - Save* con lo que aparece el cuadro de diálogo de la Figura 2.11.9, donde le damos el nombre a nuestro programa ejer1.asm , dentro de nuestra carpeta Trabajo.

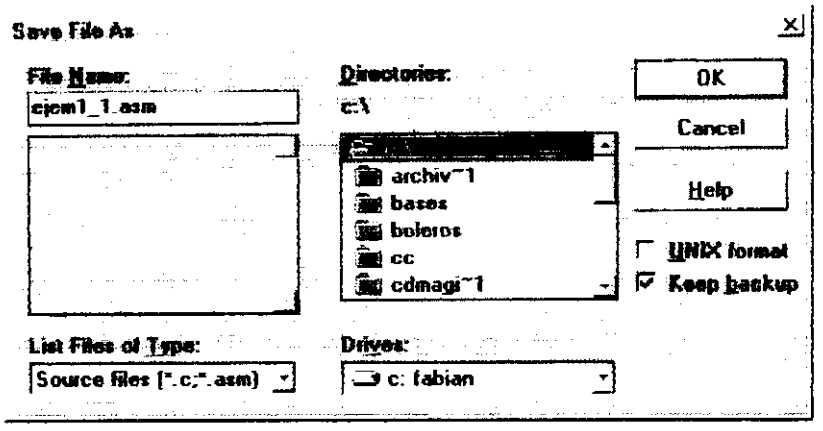


Figura 2.11.9.- Salvar el programa ejer1.asm en la carpeta de trabajo

El siguiente paso será volver a editar nuestro proyecto seleccionando en el menú de control *project - edit project* , lo que provoca que aparezca el menú de la Figura 2.11.10.

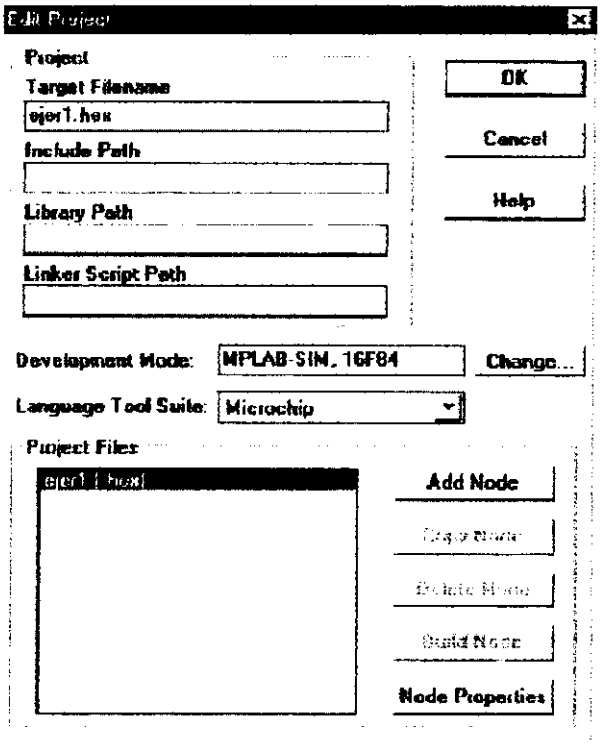


Figura 2.11.10.- Pantalla de edición del proyecto

Pulsamos sobre *ejer1[.hex]*, y se activa el botón de *Node Properties*, que hasta el momento aparecía de color gris, si lo activamos aparece el cuadro de diálogo de la Figura 2.11.12,donde están reflejadas todas las propiedades del nodo actual. Sin modificar ninguna de estas propiedades se pulsa el botón de *OK* para continuar, lo que nos lleva de nuevo a la pantalla de la Figura 2.11.10. Ahora seleccionamos el botón *Add Node* (añadir elementos al nodo), lo que provoca que aparezca un nuevo cuadro de diálogo como el de la Figura 2.11.11, en el que seleccionaremos el archivo ejer1.asm

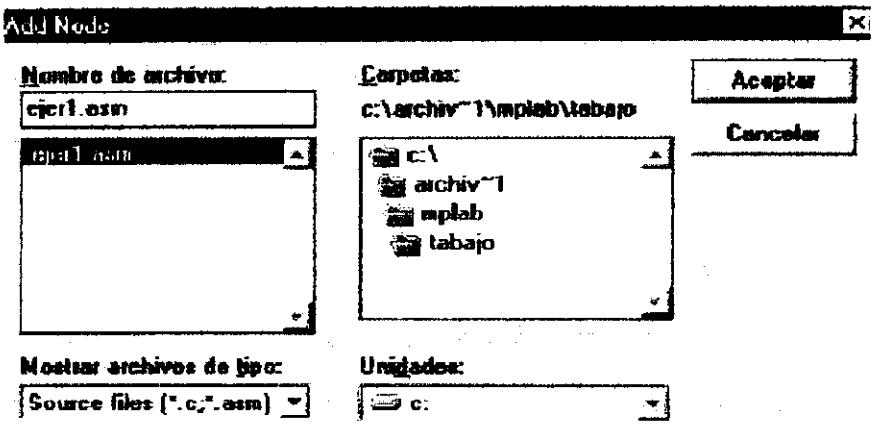


Figura 2.11.11.- Nombre del archivo a incluir en el proyecto ejer1.asm

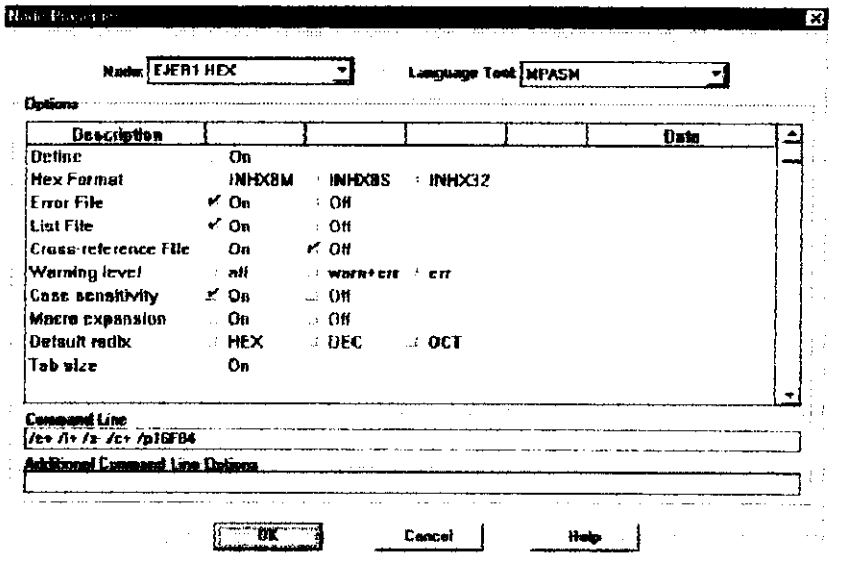


Figura 2.11.12 Propiedades del nodo de nuestro proyecto donde se seleccionan los ficheros y formatos que se obtendrán al ensamblar el programa.

Pulsamos el botón de *Aceptar* y se vuelve a la pantalla de la Figura 2.11.10 en la que ha aparecido el fichero *ejer1[.asm]* junto al fichero *ejer1[.hex]* que aparecía antes en el campo de *Project files*. Seguidamente pulsamos el botón de *OK*, lo que nos llevará de vuelta a la pantalla de la Figura 2.11.8.

Para ensamblar el programa seleccionamos en el menú de control la opción *Project - Build All* (también podríamos haber pulsado el botón correspondiente de la barra de herramientas del simulador , como luego veremos), y si no se han cometido errores al introducir los códigos, aparece una pantalla como la de la Figura 2.11.13, lo que nos indica que el programa se ha ensamblado con éxito y ya estamos en condiciones de iniciar la simulación del programa. Si por el contrario, se han detectado errores, en dicha pantalla será mostrado el error; si se hace doble clic sobre la línea que muestra el error, el cursor saltará directamente a la línea de código donde se encuentra el error.

Una vez subsanados los errores habrá que volver a compilar el programa.

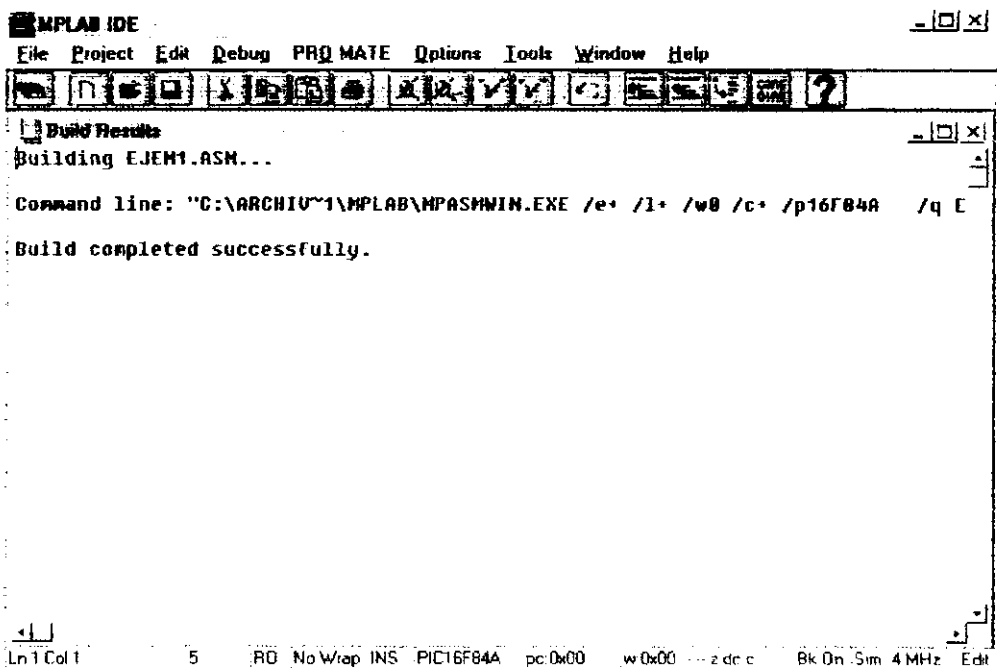


Figura 2.11.13.- Pantalla del MPLAB una vez ensamblado correctamente el programa fuente

LA BARRA DE MENÚS

Seguidamente analizaremos las distintas posibilidades de la barra de menú del MPLAB, si bien ya hemos utilizado algunas de las posibles opciones que presenta la barra de herramientas, ahora analizaremos estas una por una.

1.- *Windows*

Al activar esta opción de la barra de menú, aparece el menú desplegable de la Figura 2.11.14.

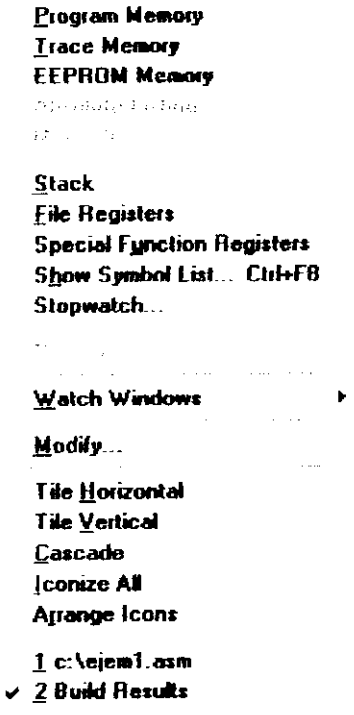


Figura 2.11.14.- Menú desplegado de la opción Windows de la barra de herramientas.

1.1.- Program Memory : Al selecciónar esta opción aparece la pantalla de la Figura 2.11.15 en la que se puede apreciar las posiciones de memoria que ocupa cada una de las instrucciones, el código de operación de cada instrucción y la posición de memoria que se le ha dado a cada etiqueta.

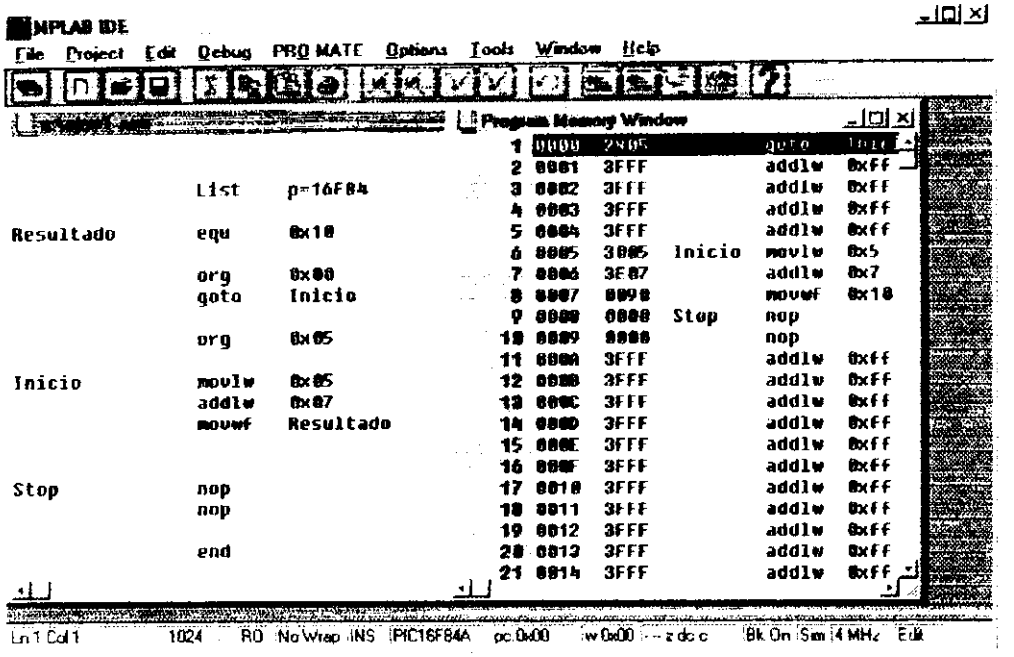


Figura 2.11.15 .- Pantalla de la ventana de Memoria de programa.

Si hacemos clic sobre la barra de *menú del sistema*, activando el icono que hay en la parte superior izquierda de esta ventana, aparece el menú desplegable de la Figura 2.11.16, en el que se puede seleccionar entre tres formas de ver la memoria de programa:

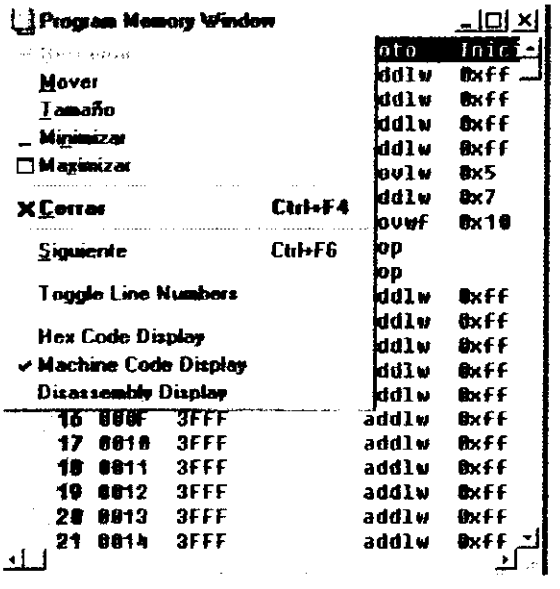


Figura 2.11.16.- Despliegue de opciones del menú de sistema.

Hex Code Display: representa la memoria de programa con los datos en hexadecimal. Esta opción es muy útil al usar el programador del dispositivo y comprobar si se grabaron bien los datos. La pantalla que se obtiene es la que se muestra en la Figura 2.11.17.

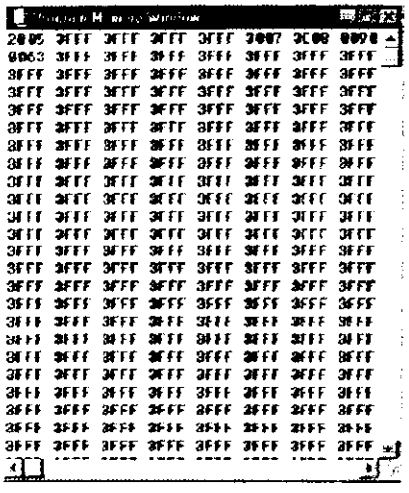


Figura 2.11.17.- Memoria de programa en código hexadecimal.

Machine Code Dispaly: esta opción presenta el código máquina ensamblado tal y como se ve en la Figura 2.11.15. Con la información de las etiquetas y . direcciones de memoria que tienen asignadas.

Disassembly Display: despliega el código hexadecimal desensamblado con los símbolos. Cuando la ventana está en la opción

Machine Code o *Disassembly Display*, la instrucción a la que apunta el contador de programa, está resaltada.

1.2.- Trace Memory : La ventana de memoria de traza toma "una instantánea" de la ejecución del programa, cuando este está corriendo en tiempo real.

Para emuladores que tienen un buffer de traza, que se utiliza cuando el programa corre en tiempo real y este no se puede detener en algunas aplicaciones, nos muestra los

puntos por los que pasa el programa. Algunos problemas sólo aparecer cuando la aplicación está corriendo, es decir, estos no dan la cara cuando se ejecuta en modo paso a paso.

La memoria de traza es una herramienta de depuración para probar tales aplicaciones.

Para más información es recomendable mirar en la guía de usuario del emulador que se esté utilizando. En el simulador, el buffer de traza o memoria de traza, es útil para visualizar un registro a lo largo de la ejecución del programa, de manera, que se puede registrar por donde pasa el programa y después analizarlo. El simulador toma datos de forma un poco distinta que el buffer del emulador.

Antes de activar la *opción Trace Memory*, para poder obtener los datos en la memoria de traza en el simulador, es necesario marcar con el ratón las líneas de código de programa de las cuales queremos obtener los datos al ejecutarse el programa, estas pueden estar en un bloque de instrucciones o bien colocadas en el programa de forma discontinua. Seguidamente, se pulsa el botón de la derecha del ratón, de manera que aparece el menú desplegable de la Figura 2.11.18.

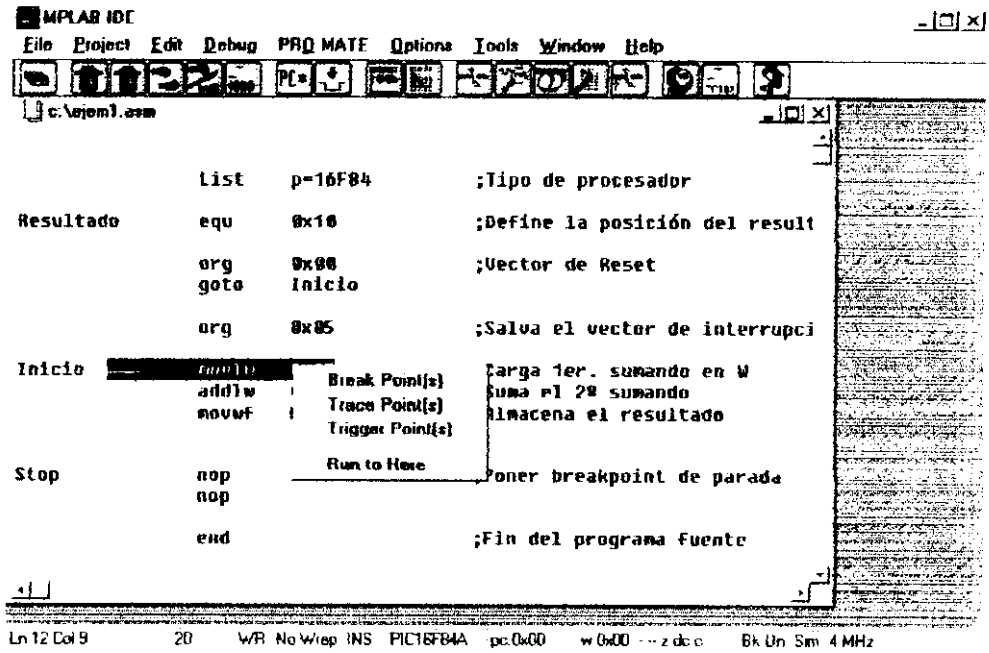


Figura 2.11.18.- Selección de las líneas de programa para cargar el buffer de traza.

Al seleccionar la opción *Trace Point(s)* aparecen resaltadas las líneas en color verde. Seguidamente se activa el icono del semáforo verde (*Run*), lo que hará ejecutar la simulación en "tiempo real"(no olvidemos que en el simulador emula el funcionamiento del microcontrolador y es mucho más lento que este), y después de unos segundos, si activamos el icono del semáforo rojo *Halt the processor*, se detiene la ejecución del programa. Si ahora se activa dentro de la opción *Windos>Trace Memory*, se pueden ver la traza obtenida, que en nuestro caso en la que se muestra en la Figura 2.11.19.

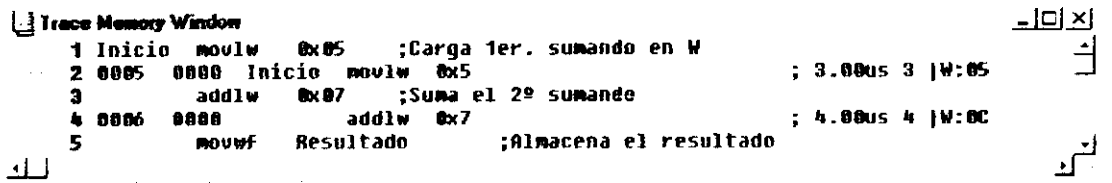


Figura 2.11.19.- Traza de memoria obtenida en el programa con los valores marcados en la Figura 2.11.18

El simulador muestra en esta ventana el valor del tiempo que tarda en ejecutar cada línea de programa y también cualquier variación sobre los registros al ejecutarse el código de instrucción.

1.3.- EEPROM Memory: Si el dispositivo emulado tiene EEPROM o memoria Flash, como es el caso del PIC16C84 o el 16F84 respectivamente, el contenido de la memoria EEPROM puede verse seleccionando la opción *Window > EEPROM*.

La memoria de EEPROM no puede modificarse a través de esta ventana. Para ello hay que utilizar el menú de dialogo al que se accede selecionando *Window > Modify...*, que se describe más adelante.

1.4.- Absolute Listing: La Ventana de "Listado de Programa", realmente nos presenta el archivo de nuestro proyecto con extensión **.lst*, donde se puede ver el archivo generado por el ensamblador o compilador. El listado muestra el código fuente en modo absoluto con el código objeto generado, tal y como se puede ver en la Figura 2.11.20.

```
MPASM 02.30 Released          EJER1.ASM   2-20-2008  10:28:18        PAGE  1

LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001 ;*****************************************************************
                     00002 ; Programa Ejer1.ASM                              Fecha : 1 - Marzo
                     00003 ; Este programa suma dos valores inmediatos (en este caso 7+8) el
                     00004 ; en la posición 0x10
                     00005 ; Revisión : 0.0                                  Programa para PIC1
                     00006 ; Velocidad del Reloj:  4 MHz                     Reloj Instrucción:
                     00007 ; Perro Guardian :      deshabilitado             Tipo de Reloj : XT
                     00008 ; Protección del código : OFF
                     00009 ;*****************************************************************
                     00010
                     00011                 LIST    p=16F84         ;Tipo de procesador
                     00012
  00000010           00013 RESULTADO       EQU     0x10            ;Define la posición del re
                     00014
0000                 00015                 ORG     0x00            ;Vector de Reset
0000 2805            00016                 goto    INICIO
                     00017
0005                 00018                 ORG     0x05            ;Salva el vector de interr
                     00019
0005 3007            00020 INICIO          movlw   0x07            ;Carga 1er. sumando en W
```

Figura 2.11.20.- Archivo ejer1.lst

Además al final de este archivo aparece la información de las etiquetas utilizadas en el programa, en que linea de programa se encuentran, la memoria utilizada, la memoria libre además de los errores, *warnings* y mensajes reportados por el ensamblador.

1.5.- Stack: El contenido de la pila puede verse al seleccionar la opción *Window - Stack*. Los contenido de la pila puede mostrarse con o sin número de línea. El formato de presentación se selecciona a través del menú del sistema. Si la Pila se desborda, el MPLAB indica con su rebosamiento con el mensaje *underflow*.

Al menú del sistema se accede pulsando el botón de la esquina de la ventana.

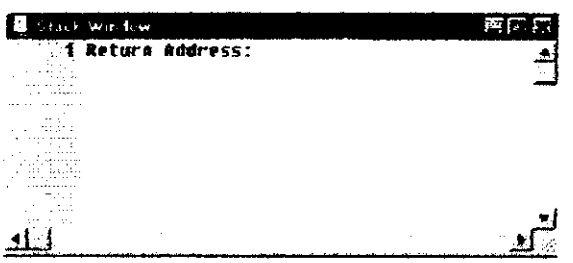


Figura 2.11.21.- Ventana de la Pila

1.6.- File Registers : La lista de registros de propósito general (GPR) del microcontrolador, que son de memoria SRAM, se pueden ver seleccionando la opción *Window - File*. Esta ventana al desplegarse presenta una lista con todos los registros de proposito general del dispositivo emulado, tal y como se muestra en la Figura 2.11.22

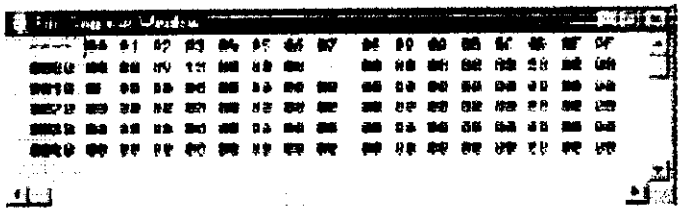


Figura 2.11.22.- Listado de registros de propósito general del sistema

Este listado de registros pueden visualizarse de tres maneras. El formato deseado se elige a través del menú del sistema.

•*Hex Display.*- Esta opción presenta los registros con datos en hexadecimal, tal y como se ve en la Figura 2.11.22.

•*Symbolic Display.*- Este formato presenta un archivo con los registros de propósito general con sus etiquetas si las tienen y su contenido en hexadecimal, decimal, binario y formato carácter tal y como se puede ver en la Figura 2.11.23.

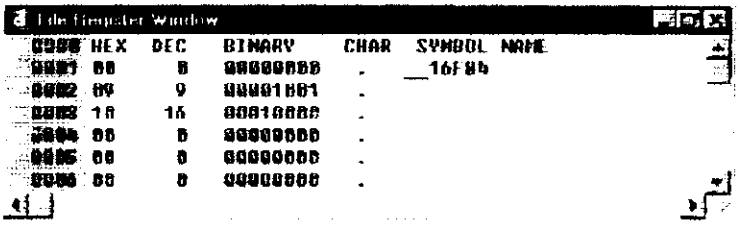


Figura 2.11.23.- Formato *Simbolic Display* del listado de registros de propósito general

•*ASCII Display* .– Esta opción presenta un listado de los registros de propósito general con el contenido de los datos en código ASCII.

Se puede modificar el contenido de uno o varios registro en esta ventana. Par ello se pone el puntero del ratón sobre el primer registro que se quiere modificar y pulsando el botón izquierdo se marca el bloque de los registros que se quieren alterar, si sólo se quiere modificar, bastará con ponerse encima de él, seguidamente se pulsa el botón de la derecha y se activa la opción *Fill Register(s)*, lo que hará aparecer la ventana de la Figura 2.11.33, en la que se puede ver como aparece la dirección del registro a modificar, una líneas mas adelante se analizan las posibilidades de esta opción.

1.7.- Special Function Registers

(SFRs): El contenido de los registros de funciones especiales (FSR) pueden verse seleccionando *Window>Special Function*

Registers. El formato proporcionado por esta ventana es más útil para analizar el estado de los FSRs en cada momento, además como puede verse en la Figura 2.11.24 se muestra cada uno de los registros con el nombre que tiene asignado además de su contenido en distintos códigos: hexadecimal, decimal, binario y formato carácter o ASCII.

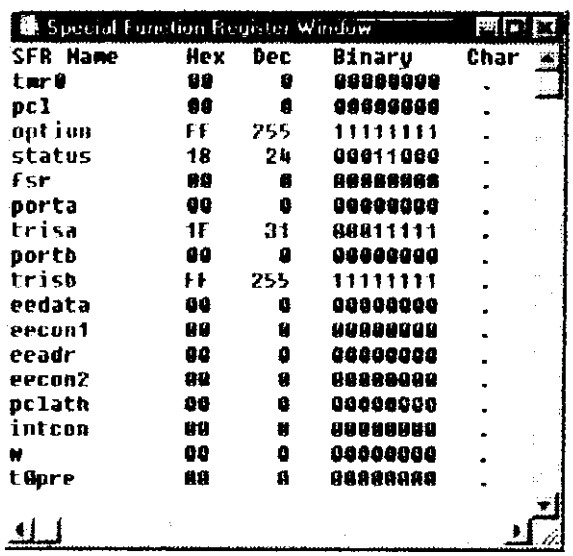


Figura 2.11.24.- Ventana de los registros especiales (FSRs).

Para modificar un SFR, hacer doble clic sobre el nombre del registro, esta acción hará aparecer el cuadro de dialogo de la opción *Modify* (Figura 2.11.33) en la que aparecerá ya la dirección del registro seleccionado.

1.8.- Show Symbol List :

Esta ventana muestra un listado de los símbolos, es decir variables y etiquetas, utilizadas en el código fuente del programa.

Estos símbolos están en el archivo *.COD de nuestro proyecto. En la Figura 2.11.25 se muestra el listado de símbolos de nuestro programa.

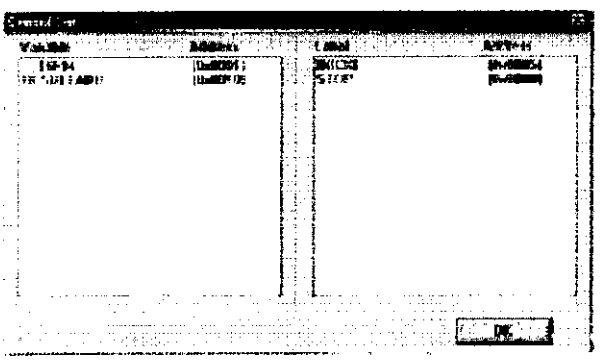


Figura 2.11.25.- Ventada del listado de símbolos utilizados en nuestro programa fuente.

1.9.- Stopwatch and Clock Frecuency :

Para calcular el tiempo de ejecución de nuestro programa o de una subrutina, podemos contar el número de instrucciones que se realizan y multiplicarlo por 4 veces la frecuencia de la señal de reloj (tiempo de un ciclo máquina) o por 8 en el caso de que las instrucciones sean de salto. Esto en algunas ocasiones es engorroso, pero el MPLAB con esta opción de cronómetro nos permite medir tiempo de ejecución de las instrucciones de nuestro programa sin equivocaciones.

El cronómetro calcula el tiempo basándose en la frecuencia del reloj del microcontrolador PIC que estamos simulando, para ello previamente debemos fijar la frecuencia del oscilador empleado. Esto se realiza haciendo los siguientes pasos:

Activamos desde el menú *Options*> *Processor Setup*>*Clock frequency* tal y como se muestra en la Figura 2.11.26

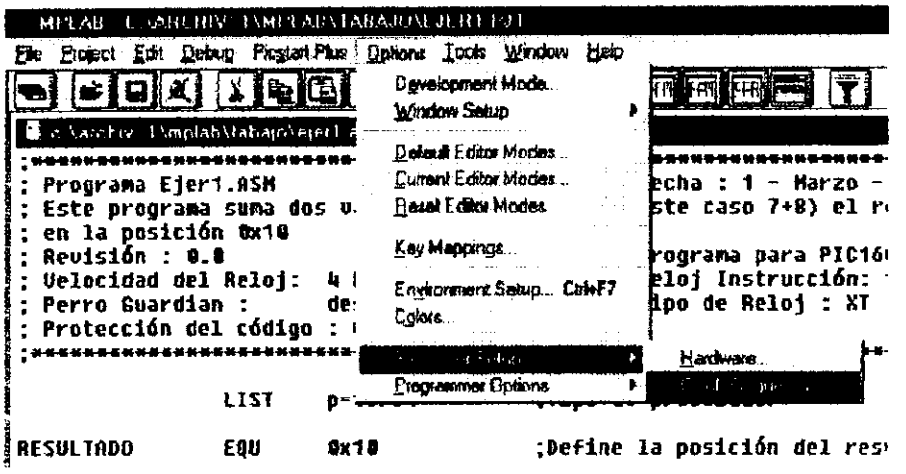


Figura 2.11.26.- Camino a seguir para definir la frecuencia del microcontrolador

Options>*Processor*

Setup>*Clock frequency*

Inmediatamente se abre un cuadro de dialogo como la de la Figura 2.11.26, donde se fija la frecuencia del reloj.

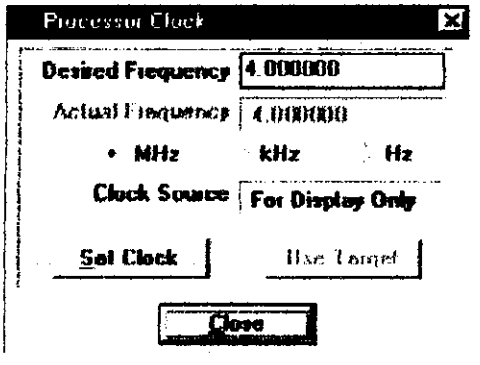


Figura 2.11.27. Definición de la frecuencia de oscilador del microcontrolador.

Después se activa la opción *Windows -StopWatch*, con esto conseguimos tener siempre abierta la ventana que muestra el tiempo transcurrido y los ciclos máquina empleados en la ejecución de cada instrucción, como puede verse en la Figura 2.11.28.

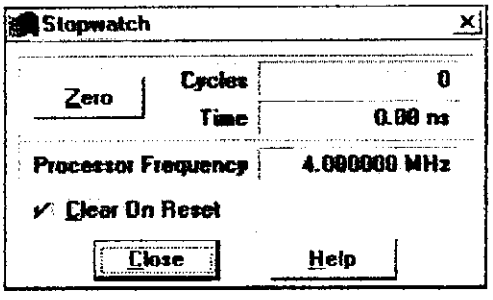


Figura 2.11.28.- Cronometro para contar el tiempo que tarda en ejecutarse un programa o parte de él.

1.10.- Project Windows : La Ventana del Proyecto sólo está disponible cuando hay un proyecto abierto. Presenta la lista de archivos que actualmente hay en dicho proyecto. Si el proyecto se ha ensamblado o compilado la ventana del proyecto muestra una lista de todos los archivos incluidos en el proyecto.

Por otra parte, la ventana del Proyecto sólo presenta el archivo del proyecto principal. Un doble clic en cualquier archivo resaltado en la ventana del proyecto, abrirá dicho archivo para su revisión.

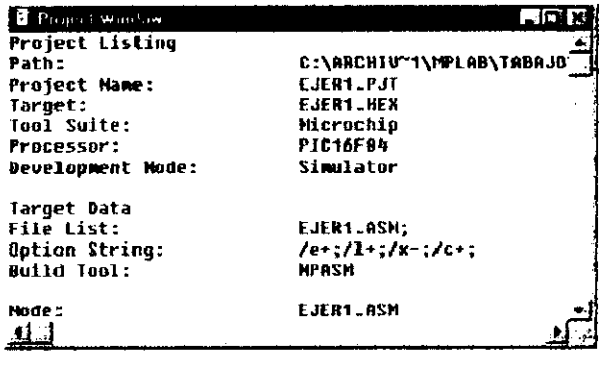


Figura 2.11.29.- Ventana de Proyecto

1.11.- Watch Windows :

MPLAB permite supervisar los contenidos de los registros del archivo a través de una ventana temporal. Para abrir una ventana temporal, se selecciona *Window - Watch Windows*. El programa responde con un cuadro de diálogo como el de la Figura. 2.11.30

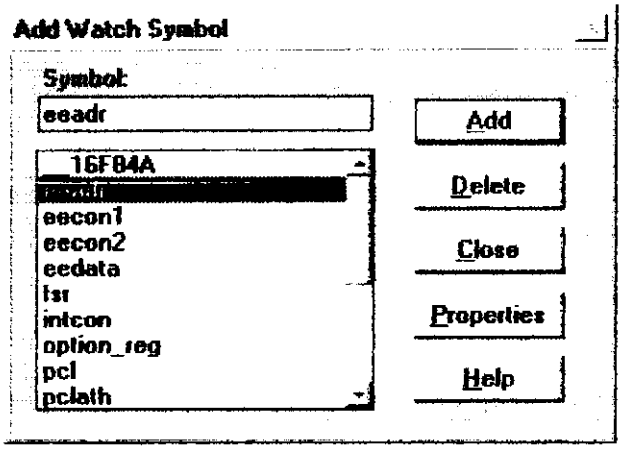


Figura 2.11.30.- Cuadro de diálogo de los símbolos de la ventana temporal

Para agregar los registros a visualizar, poner el ratón encima de uno de ellos pulsar el botón de la izquierda, seguidamente activar el botón de *Add*. También se pueden anular los símbolos poniendose sobre ellos y pulsando el botón izquierdo del ratón y seguidamente el botón de *Delete*. Cuando estén todos los registros seleccionados pulsar el botón de *Close* y aparecerá una ventana, en este primer caso, Watch_1, como puede verse en la Figura 2.11.31, en la que se ven los símbolos (etiquetas y variables) seleccionados.

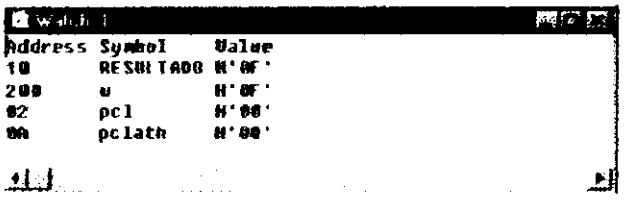


Figura 2.11.31- Ventana Watch_1

Para ver y cambiar las propiedades de un símbolo, hay que pulsar el botón de propiedades que aparece en el cuadro de diálogo de la Figura 2.11.30, al hacerlo aparece un nuevo cuadro de diálogo como el de la Figura 2.11.32

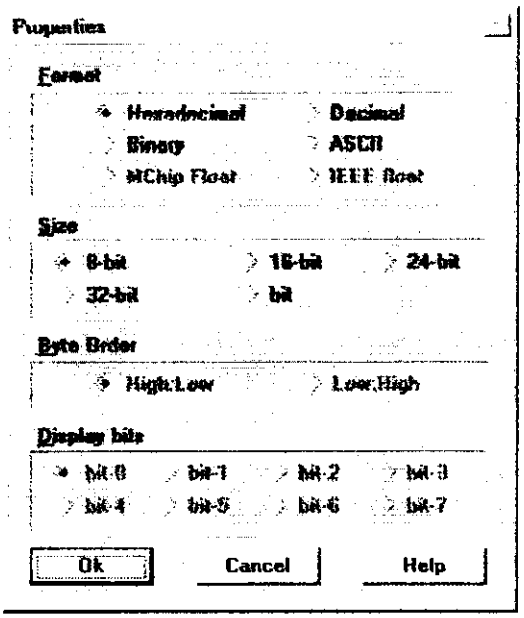


Figura 2.11.32.- Cuadro de diálogo de las propiedades de los datos de las ventanas temporales.

El contenido de la ventana del reloj puede desplegarse mostrando o sin mostrar los números de la línea. Para elegir el formato deseado se hace a través del menú del sistema (pulsando el icono de la parte superior izquierda de la ventana).El menú del sistema también se usa para revisar la información en la ventana temporal.

La ventana de diálogo permite seleccionar el formato en que el se presentan los símbolos:

•*Format.*- Determina qué tipo de numeración se desea visualizar.

•*Size.*- Determina cuántos bytes serán incluidos en la visualización del número: hexadecimal, decimal, binario, ASCII o float.

Byte Order.- Determina el orden de visualización de cada byte, disponible sólo para los números de 16 bits.

•*Display Bits.*- Determina en qué momento se visualiza el bit seleccionado al activarse.

1.12.- Modify: Al activar la opción *Window>Modify...* aparece el cuadro de diálogo *Modify* como el que se muestra en la Figura 2.11.33. En este cuadro se permite leer y escribir una posición de memoria o el rango de una posición de memoria. *Modify* puede trabajar en las áreas de memoria siguientes:

- Data
- Stack
- Program
- EEPROM (Si tiene)

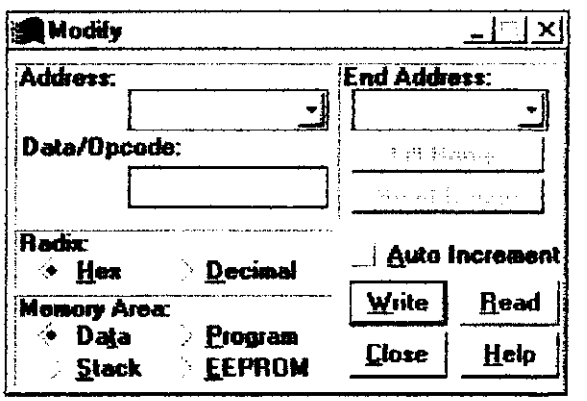


Figura 2.11.33.- Cuadro de diálogo de la opción Modify

Como resumen a todo lo que hemos contado hasta el este momento, podemos decir que el comando *Windows*, puede presentar una visión de todos los registros del microcontrolador en cada momento y podemos tener al final una pantalla en la que

visualicemos según nos interese las ventanas mas adecuadas para el seguimiento de nuestra aplicación, como puede ser la de la Figura 2.11.34.

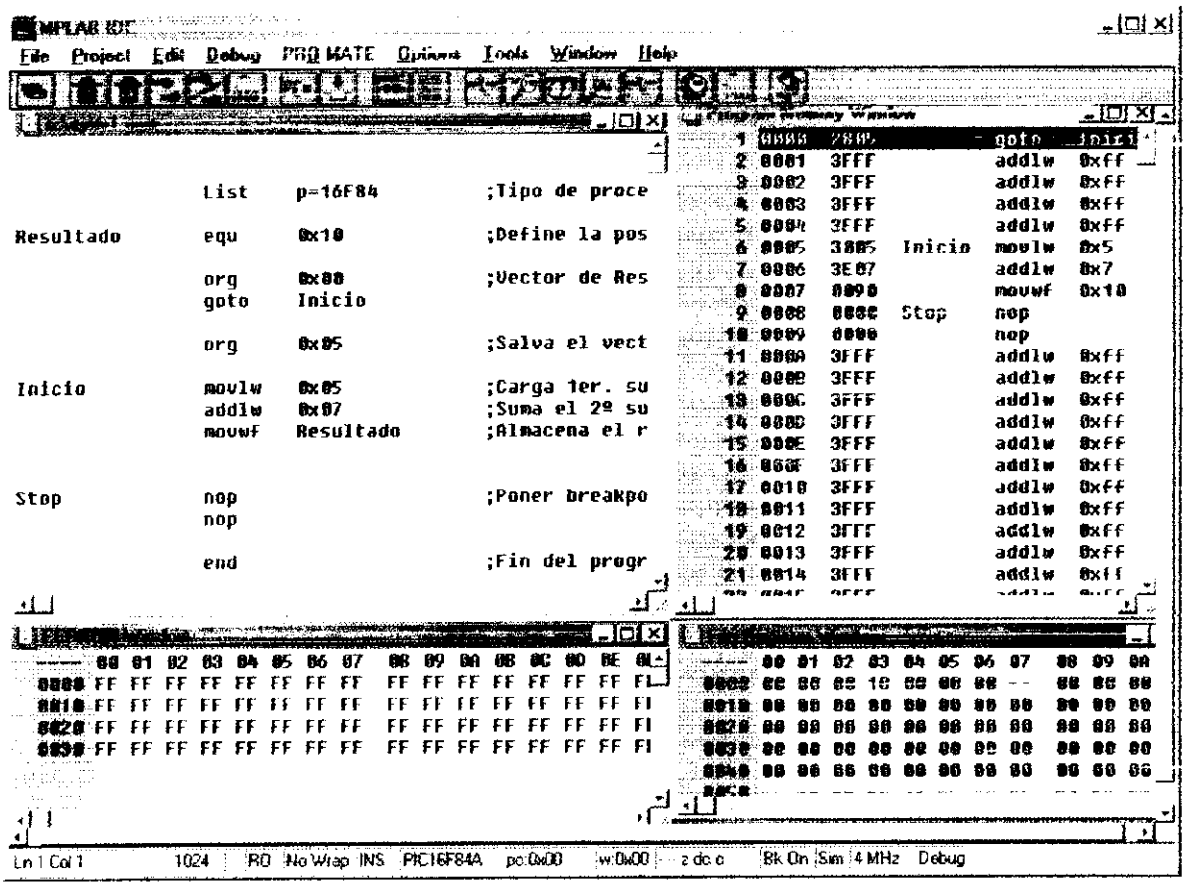


Figura 2.11.34.- Presentación de algunas ventanas de forma simultánea en pantalla

2.12 Tipos de Instrucciones

El Pic utiliza arquitectura Risc con lo cual gana velocidad de trabajo lo que es muy veloz por el motivo que solo usa 35 instrucciones y se combina entre ellas a continuación indicaremos cada una de la instrucciones.

TABLA DE INSTRUCCIONES DE LA FAMILIA PIC

MNEMÓNICO	DESCRIPCIÓN	AFECTADOS
INSTRUCCIONES ORIENTADAS A REGISTROS		

ADDWF f,d	(W)+(f) -> (destino)	C, DC, Z
ANDWF f,d	(W) AND (f) -> (destino)	Z
CLRF f	00 -> (f)	Z
CLRW	00 -> (W)	Z
COMF f,d	Complemento de f [(#f) -> (destino)]	Z
DECF f,d	(f)-1 -> destino	Z
DECFSZ f,d	(f)-1 -> destino y si resultado es 0 salta	Ninguno
INCF f,d	(f)+1 -> destino	Z
INCFSZ f,d	(f)+1 -> destino y si resultado es 0 salta	Ninguno
IORWF f,d	(W) OR (f) -> destino	Z
MOVF f,d	Mueve f -> destino	Z
MOVWF f	(W) -> (f)	Ninguno
NOP	No operación	Ninguno
RLF f,d	Rota f a la izq a través del carry -> destino	C
RRF f,d	Rota f a la dcha a través del cary ->destino	C

SUBWF f,d	(f)–(W) -> (destino)	C,DC,Z
SWAPF f,d	Intercambia los nibbles de f -> destino	Ninguno
XORWF f,d	(W) XOR (f) -> (destino)	Z
INSTRUCCIONES ORIENTADAS A BIT		
BCF f,b	Pone a 0 el bit b del registro f	Ninguno
BSF f,b	Pone a 1 el bit b del registro f	Ninguno
BTFSC f,b	Skip si el bit b del reg. f es 0	Ninguno
BTFSS f,b	Skip si el bit b del reg. f es 1	Ninguno
INSTRUCCIONES CON LITERALES Y DE CONTROL		
ADDLW K	(W)+ K -> (W)	C,DC,Z
ANDLW K	(W) AND K -> (W)	Z
CALL K	Llamada a subrutina	
CLRWDT	Clear del temporizador del WD	Ninguno
GOTO K	Go To dirección	Z
IORLW K	(W) OR K -> (W)	Ninguno
MOVLW K	K -> (W)	Ninguno
RETFIE	Retorno de una interrupción	Ninguno
RETLW K	Retorno con un literal en W	Ninguno

RETURN	Retorno de una subrutina	C,DC,Z
SLEEP	Modo Standby	Z
SUBLW K	K – (W) -> W	
XORLW K	(W) XOR K -> (W)	

ADDLW

Suma el contenido del registro W al literal k, y almacena el resultado en W.Si se produce acerreo el flag C se pone a "1"

ADDLW		**ADDLW**
ADD Literal to W		
Operación	(W) + k ----> (W)	
Sintaxis	ADDLW k	

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	X	X

C Se pone a 1 si se produce un Acarreo desde el bit de mayor peso

DC Se pone a 1 si se genera un Acarreo del bit 3 al bit 4.

Z Se pone a 1 si el resultado de la operación es cero

EJEMPLO: ADDWF FSR,0

Si antes de la instrucción. W = 17 h y FSR = C2 h como d=0

Al ejecutarse: W = 17 h + C2 h = D9 h

FSR = C2 h

ANDLW

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

ANDLW		**ANDLW**
AND Literal and W		
Operación	(W).AND. (k) ----> (W)	
Sintaxis	[Etiqueta] ANDLW k	

EJEMPLO: ANDLW 0x5F

Si antes de la instrucción. W = A3 h

Al ejecutarse: W = 0101 1111 b AND 1010 0011 b = 0000 0011 B = 03 h

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Nota. Z Se pone a 1 si el resultado de la operación es cero

ANDWF

Efectúa la operación AND lógico entre el contenido del registro W y el contenido del registro f, y almacena el resultado en W si d = 0, y en f si d = 1

ANDWF		**ANDWF**
AND W wind F		
Operación	(W) AND (f) ----> (destino)	
Sintaxis	[Etiqueta] ANDWF f,d	

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Z Se pone a 1 si el resultado de la operación es cero

BCF

Pone a cero el bit número b del registro f

BCF		**BCF**
Bit Clear F		
Operación	0 --> (f<b>)	
Sintaxis	[Etiqueta] BCF f,b	

Ejemplo Bcf valor,7

Antes de la instrucción valor=c7

Después de la instrucción valor=47

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

BSF

Pone a 1 el bit b del registro f

BSF		**BSF**
Bit Set F		
Operación	1 --> (f<b>)	
Sintaxis	[Etiqueta] BSF f,b	

Ejemplo Bsf valor,7

Antes de la instrucción valor=0A

Después de la instrucción valor=8A

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

BTFSC

Si el bit número b del registro f es cero, la instrucción que sigue a ésta se ignora y se trata como un NOP (skip). En este caso, y sólo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.

BTFSC		**BTFSC**
Bit Test, Skip if Clear		
Operación	skip if (f<b>) = 0	
Sintaxis	[Etiqueta] BTFSC f,b	

Ejemplo Aquí Btfsc valor,0

Falso Goto inicio

Verdad .....

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

BTFSS

Si el bit número b del registro f está a 1, la instrucción que sigue a ésta se ignora y se trata como un NOP (skip). En este caso, y sólo en este caso, la instrucción BTFSS precisa dos ciclos para ejecutarse.

BTFSS		**BTFSS**
Bit Test, Skip if Set		
Operación	skip if (f<b>) = 1	
Sintaxis	[Etiqueta] BTFSS f,b	

Ejemplo Aquí Btfss valor,0

Falso Goto inicio

Verdad .....

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

CALL

Salvaguarda la dirección de vuelta en la Pila y después llama a la subrutina situada en la dirección cargada en el PC.

El modo de cálculo de la dirección efectiva difiere según la familia PIC utilizada. También hay que posicionar PA2, PA1 y PA0 (PIC 16C5X) o el registro PCLATCH (En los demás PIC) antes de ejecutarse la instrucción.

CALL		**CALL**
Subrutine Call		
Sintaxis	[Etiqueta] CALL k	
Operación	(PC)+1 ---> Top of Stack	
	k ---> PC <10:0>;	
	PCLATCH (<4:3>) ---> PC (<12,11>)	

Ejemplo Aquí Call Rutina

Registro de STATUS

PA2	**PA1**	**PA0**	**TO#**	**PD#**	**Z**	**DC**	**C**
-	-	-	-	-	-	-	-

CLRF

Se borra el contenido del registro f y el flag Z se activa

CLRF		**CLRF**
Sintaxis	[Etiqueta] CLRF f	

Ejemplo Clrf f

Antes de la instrucción f=08

Después de la instrucción f=00

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	1	-	-

Z Se pone a 1 si el resultado de la operación es cero

CLRW

El registro de trabajo W se carga con 00h. El flag Z se pone a 1

CLRW		**CLRW**
	Clear W	
Sintaxis	[Etiqueta] CLRW	

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	1	-	-

Z Se pone a 1 si el resultado de la operación es cero

Ejemplo Clrw

Si antes de la instrucción. W= 5Ah

Al ejecutarse: W = 00

CLRWDT

Se borra tanto el registro WDT (Watchdog) como su preescaler. Los bits T0# y PD# del registro de estado se ponen a "1".

CLRWDT		**CLRWDT**
Clear Watchdog Timer		
Sintaxis	[Etiqueta] CLRWDT	

Ejemplo Clrwdt

Antes de la instrucción Contador WDT=?

Después de la instrucción Contador WDT=00

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	1	1	-	-	-

T0# Se pone a 1 cuando se ejecuta la instrucción CLRWDT o SLEEP. Se pone a 0 si el temporizador Watchdog se desborda

PD# Se pone a 1 cuando se ejecuta la instrucción CLRWDT o SLEEP

COMF

Hace el complemento del contenido del registro f bit a bit. El resultado se almacena en el registro f si d=1 y en el registro W si d=0, en este caso f no varía.

COMF		**COMF**

Complement f	
Sintaxis	[Etiqueta] COMF f,d

Ejemplo Comf valor,0

Antes dc la instrucción valor=13, w=?

Después de la instrucción valor=13, w=EC

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Z Se pone a 1 si el resultado de la operación es cero

DECF

Se decrementa el contenido del registro f en una unidad. El resultado se almacena en f si d=1 y en W si d=0, en este caso f no varía.

DECF		**DECF**
Decrement f		
Operación	(f)-1 --> (dest)	
Sintaxis	[Etiqueta] DECF f,d	

Ejemplo Decf valor,1

Antes de la instrucción valor=13

Después de la instrucción valor=12

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Z Se pone a 1 si el resultado de la operación es cero

DECFSZ

Decrementa el contenido del registro f en una unidad, el resultado se almacena en f si d=1 y en W si d=0, en este caso, f no varía. Si el resultado es cero, se ignora la siguiente instrucción y, en ese caso la instrucción tiene una duración de dos ciclos.

DECFSZ		**DECFSZ**
Decrement f , Skip if 0		
Sintaxis	[Etiqueta] DECFSZ f,d	

Ejemplo Aquí Decfsz valor,1

Goto Ciclo

Continúa.....

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

GOTO

Salto incondicional, normalmente se utiliza para llamar a la subrutina situada en la dirección que se carga en PC.

El modo de cálculo de la instrucción caga de bit 0 al 10 de la constante k en el PC y los bits 3 y 4 del registro PCLATH en los 11 y 12 del PC

GOTO		**GOTO**
Unconditional Branch		
Sintaxis	[Etiqueta] GOTO k	

Ejemplo Goto Ciclo

Antes el contador del Programa =?

Después el contador del Programa =Ciclo

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

INCF

Se incrementa en una unidad el contenido del registro f, si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W, en este caso el resultado de f no varía.

INCF		**INCF**
Increment f		
Operación	(f) + 1 --> (dest)	
Sintaxis	[Etiqueta] INCF f,d	

Ejemplo Incf valor,1

Antes de la instrucción valor=24

Después de la instrucción valor=25

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Z Se pone a 1 si el resultado de la operación es cero al haber desbordamiento

INCFSZ

Incrementa el contenido del registro f en una unidad, el resultado se almacena de nuevo en f si d=1, y en W si d=0, en este caso, f no varía. Si el resultado es cero, se ignora la siguiente instrucción y, en ese caso la instrucción tiene una duración de dos ciclos.

INCFSZ		**INCFSZ**
Increment f, SKIP if 0		
Operación	(f) +1 --> (dest) , skip if result = 0	
Sintaxis	[Etiqueta] <INCFSZ f,d	

Ejemplo Aquí Decfsz valor,1

Goto Ciclo

Continúa.....

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

IORLW

Se realiza la operación lógica OR entre el registro W y el literal k. El resultado se almacena en el registro W.

IORLW		IORLW
Inclusive OR Literal with W		
Operación	(W).OR.k ---> (W)	
Sintaxis	[Etiqueta] IORLW k	

Ejemplo Iorlw 35

Antes de la instrucción w=9ª

Después de la instrucción w=BF

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Z Se pone a 1 si el resultado de la operación es cero

IORWF

Efectúa la operación lógica OR entre el contenido del registro W y el contenido del registro f, y almacena el resultado en f si d=1 y en W si d=0.

IORWF		IORWF
Inclusive OR W with f		
Operación	(W) .OR.(f)--> (dest)	
Sintaxis	[Etiqueta] IORWF f,d	

Ejemplo Iorlwf valor,0

Antes de la instrucción valor=13 , w=91

Después de la instrucción valor=13 , w=93

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Z Se pone a 1 si el resultado de la operación es cero

MOVLW

El registro W se carga con el valor de 8 bits del literal k

MOVLW		**MOVLW**
Move literal to W		
Operación	K --> (W)	
Sintaxis	[Etiqueta] MOVLW k	

Ejemplo Movlw valor

Antes de la instrucción w=4, valor=00
Después de la instrucción w= 4, valor=4

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

MOVF

El contenido del registro f se carga en el registro destino dependiendo del valor de d. Si d=0 el destino es el registro W, si d=1 el destino es el propio registro f . Esta instrucción permite verificar dicho registro ya que el flag Z queda afectado.

MOVF		**MOVF**
Move f		
Operación	(f) --> (dest)	
Sintaxis	[Etiqueta] MOVF f,d	

Ejemplo Movf valor,0

Antes de la instrucción w=?

Después de la instrucción w= el dato guardado en regis.

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

Z Se pone a 1 si el resultado de la operación es cero

MOVWF

Mueve el contenido del registro W al registro f

MOVWF		**MOVWF**
Move W to f		
Operación	(W)--> (f)	
Sintaxis	[Etiqueta] MOVWF f	

Ejemplo Movwf valor

Antes de la instrucción w=24, valor=?

Después de la instrucción w= 24, valor=4

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

NOP

No realiza operación alguna. En realidad, se consume un ciclo de instrucción sin hacer nada.

NOP		**NOP**
No operation		
Operación	no operación	
Sintaxis	[Etiqueta] NOP	

Ejemplo NOP
NOP
NOP

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

EJEMPLO: NOP

RETFIE

Carga el PC con el valor que se encuentra en la parte alta de la Pila, asegurando así la vuelta de la interrupción. Pone a 1 el bit GIE, con el fin de autorizar de nuevo que se tengan en cuenta las interrupciones.

RETFIE		**RETFIE**
Return from Interrupt		
Operación	TOS --> PC	
	1 --> GIE	
Sintaxis	[Etiqueta] RETFIE	

Ejemplo Retfie

Antes de la instrucción =?

Después de la instrucción =Pila

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

RETLW

RETLW		**RETLW**
Retur with Literal in W		
Operación	k --> (W);	
	TOS ---> PC	
Sintaxis	[Etiqueta] RETLW k	

Ejemplo

```
Call    Tabla
  .
  .
Tabla   Addwf   Pc
        Retlw   K1
        Retlw   K2
          .
```

.
.
Retlw Kn

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

RETURN

Carga el PC con el valor que se encuentra en la parte superior de la PILA, efectuando así un retorno de subrutina

RETURN		**RETURN**
Return from Subroutine		
Operación	TOS ---> PC	
Sintaxis	[Etiqueta] RETURN	

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

Ejemplo Return

Después de la interrupción, contador de programa=Pila

RLF

Rotación de un bit a la izquierda del contenido del registro f, pasando por el bit de acarreo C. Si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W.

RLF		**RLF**
Rotate Left f through Carry		
Sintaxis	[Etiqueta] RLF f,d	

Ejemplo Rlf valor,0

Antes de la instrucción C=0, W=?, valor=11100110

Después de la instrucción C=1, W = 11001100, valor=11100110

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

RRF

Rotación de un bit a la derecha del contenido del registro f, pasando por el bit de acarreo C. Si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W

RRF		**RRF**
Rotate Right f through Carry		
Sintaxis	[Etiqueta] RRF f,d	

Ejemplo Rrf valor,0

Antes de la instrucción C=0, W=?, valor=11100110

Después de la instrucción C=1, W = 01110011, valor=11100110

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

SLEEP

Pone al circuito en modo Sleep (bajo consumo) con parada del oscilador. Pone a 0 el flag PD# (Power Down) y el flag TO# (Timer Out) se pone a 1. Se puede salir de este estado por:

1. Activación de MCLR para provocar un Reset
2. Desbordamiento del Watchdog si quedó operativo en el modo reposo
3. Generación de una interrupción que no sea TMR0 ya que ésta se desactiva con la instrucción SLEEP

SLEEP		**SLEEP**
	Sleep	
Operación	00h ---> WDT 0 ---> WDT prescaler 1 ---> TO# 0 --> PD#	
Sintaxis	[Etiqueta] SLEEP	

Ejemplo SLEEP

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	1	0	-	-	-

TO Se pone a 1 al ejecutar la instrucción SLEEP o CLRWDT

PD Se pone a 0 al ejecutar la instrucción SLEEP

SUBLW

Resta en complemento a dos del contenido del literal k el contenido del registro W, y almacena el resultado en W.

SUBLW		**SUBLW**
Subtract W from Literal		
Operación	k - (W) ---> (W)	
Sintaxis	[Etiqueta] SUBLW k	

Ejemplo Sublw 02

Antes de la instrucción C=?, W=1

Después de la instrucción C=1, W =1

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	X	X

Z Se pone a 1 si el resultado de la operación es cero

DC Se pone a 1 si se genera un acarreo del bit 3 al grupo de 4 bits superior

C Se pone a 1 si se genera un acarreo del bit de mayor peso

SUBWF

Resta en complemento a dos el contenido del registro f menos el contenido del registro W almacena el resultado en W si d=0 y en f si d=1.

SUBWF		SUBWF
Subtract W from f		
Operación	(f) - (W) ---> (dest)	
Sintaxis	[Etiqueta] SUBW f,d	

Ejemplo Subwf valor,1

Antes de la instrucción C=?, W=2, valor=3

Después de la instrucción C=1, W =2, valor=1

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	X	X

Z Se pone a 1 si el resultado de la operación es cero

DC Se pone a 1 si se genera un acarreo del bit 3 al grupo de 4 bits superior

C Se pone a 1 si se genera un acarreo del bit de mayor peso

SWAPF

Los cuatro bits de más peso del registro f se intercambian con los 4 bits de menos peso del mismo registro. Si d=0 el resultado se almacena en W, si d=1 el resultado se almacena en f.

SWAPF		SWAPF
Swap Nibbles in f		
Sintaxis	[Etiqueta] SWAPF f,d	

Ejemplo Swapf valor,0

Antes de la instrucción valor=A5, W=?

Después de la instrucción valor=A5, W =5A

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	-	-	-

XORLW

Realiza la función OR-Exclusiva entre el contenido del registro W y la constante k de 8 bits. El resultado se almacena en W

XORLW		**XORLW**
Exclusive OR Literal With k		
Operación	(W).XOR.k ---> (W)	
Sintaxis	[Etiqueta] XORLW k	

Ejemplo Xorlw AF

Antes de la instrucción W=B5

Después de la instrucción W=1A

Registro de STATUS

PA2	PA1	PA0	TO#	PD#	Z	DC	C
-	-	-	-	-	X	-	-

Z Se pone a 1 si el resultado de la última operación es cero

ej.: inicio movlw 0 ;carga el registro w con 0

Las tabulaciones se utilizan en el editor de texto para separar el contenido del programa en columnas. El programa ensamblador entiende este formato y entonces puede proceder a ensamblarlo. La longitud del tabulador (8, 9 ó 10 espacios) no es critica, el ensamblador reconoce cuando se pasa de una columna a otra.

Etiquetas

Son nombres simbólicos que se asignan a una dirección de La memoria. Por ejemplo, cuando se escribe La palabra inicio, se está asignando el símbolo inicio a un valor propio del microcontrolador, que es la dirección de memoria correspondiente donde quedará grabada esa instrucción.

Esto es de gran utilidad porque no es necesario trabajar con los números reales de las direcciones físicas, sino que se puede trabajar con algunos símbolos o palabras que son de muy fácil recordación, sin importar en donde esté ubicado. Las reglas para definir etiquetas son muy simples:

- Deben empezar en la primera posición de la columna uno.
- Pueden incluir caracteres alfanuméricos, línea de subrayado (-) e interrogación.
- Su longitud no debe exceder de 31 caracteres.

Operación

Es La instrucción del microcontrolador que se ejecuta. Todo el conjunto de instrucciones del PIC16C84 se vera en el siguiente capitulo.

Operandos

Son los registros o cantidades sobre los que se realizan las operaciones o instrucciones. Puede ser un registro de la memoria de datos o un valor constante que comúnmente se conoce como <<literal>>. Cuando se utilizan dos operandos, el primero es el operando fuente y el segundo el operando destino.

Literal

Es una constante, usualmente un número hexadecimal. Ejemplo:

```
start        movlw        00
```

En este caso, el Literal (número) es el 00. Los literales son valores que se definen usando la instrucción movlw y otras que se verán más adelante.

Comentario

El ensamblador ignora esta línea en el momento de generar el código objeto, pero para la persona que hace el programa es muy importante ya que ahí puede escribir la idea o la explicación de lo que está haciendo en esa línea del programa.

Punto y coma (;).

Es un delimitador; hace que el ensamblador ignore la línea de texto que se encuentra a la derecha de él. Se usa para escribir comentarios acerca de la instrucción particular que se tiene de la acción que está ejecutando el microcontrolador. Ejemplo: En la línea que empieza con inicio, se explica lo que hace la instrucción movlw.

CAPITULO III

DESARROLLO DEL PROYECTO

CAPITULO III

DESARROLLO DEL PROYECTO

3.1 Tipos de Hardware para quemar el Chip

Hemos encontrado una cantidad de programadores importante dispersos por la web y la bibliografía, con diversas complejidades y prestaciones diferentes. Por ello, hemos elegido el programador o hardware más sencillo posible pero que mantuviera un nivel de prestaciones alto; con un puñado de componentes poco costosos y muy sencillo también tiene que ser capaz de manejar la mayoría de los microcontroladores PIC actuales con una gran facilidad.

Nuestro programador necesita, además, una alimentación que puede ser continua o alterna, comprendida entre 12 y 30V y que no es preciso que esté estabilizada. Como los programadores profesionales, nuestro montaje es capaz, naturalmente, de leer, verificar, programar y comparar los PIC sin ninguna restricción, lo mismo que puede leer y programar sus fusibles de configuración.

Aunque las memorias de PIC se programan en serie, nuestro programador se conecta al puerto paralelo del PC. En efecto, por una parte este puerto se puede controlar muy fácilmente por software y, por otra parte, suministra niveles TTL directamente utilizables. Además, debemos disponer de algunas líneas de control para conmutar las diversas alimentaciones del microcontrolador en el curso de la programación, lo cual es mucho más fácil de realizar en un puerto paralelo que en un puerto serie.

Clases de Programadores o Hardware

Aunque la mayoría de PICs utilizan esencialmente el mismo sistema de programación, existen entre ellos sutiles diferencias, no sólo en cuanto a la disposición de las patillas utilizadas para la programación, sino también en cuanto a las señales eléctricas necesarias, lo que hece que algunos programadores sólo puedan programar ciertos tipos

de PICs. Aquí presentaré cuatro programadores con distintas posibilidades y limitaciones, para que el lector elija el que se adapte mejor a sus necesidades.

Se pueden encontrar en Internet cientos de programadores para PICs distintos. La mayoría no funcionan o, al menos, no hacen todo lo que se espera de ellos y también son muy costos. La ventaja principal de los quemadores o programadores es que son compatibles con gran parte del software de programación existente.

Los cuatro programadores que explicaremos han sido probados en distintos ordenadores y con distintos PICs, lo que me permitirá, en cada caso, enumerar sus posibilidades y limitaciones con bastante exactitud.

La razón de publicar los dos primeros es que, si bien son limitados, su precio no pasa de 500 (3 euros) y permitirán programar los modelos de PICs que utilizaré en montajes posteriores, por lo que son ideales para quien necesite programar un PIC de forma muy esporádica, quedando el tercero para los aficionados a la Electrónica que quieran hacer sus propios diseños y necesiten una herramienta flexible, segura y casi universal.

El primer programador es el que yo he llamado JDM2, por ser una versión modificada de uno de los muchos diseños de JDM. Tan sólo es capaz de programar los PICs 12C508 y 12C509, pero lo hace sin fallos, contrariamente a muchos programadores.

Utiliza un puerto serie para comunicarse con el PC, y de él obtiene las tensiones necesarias, por lo que no necesita fuente de alimentación. La desventaja de este sistema es que si las señales del puerto serie no cumplen con la norma RS232, como ocurre en muchos portátiles, no funcionará correctamente. Sin embargo en la mayoría de ordenadores de sobremesa no habrá problema. Utiliza un software específico muy básico, pero que permite realizar las operaciones necesarias (leer y grabar el PIC).

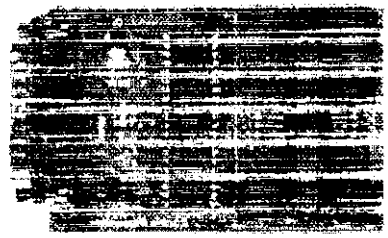


Figura 3.1-1 Programador JDM2

El segundo se ha llamado PIPO2, ya que es una modificación del Ludipipo adaptada a la programación de diversos PICs. Al igual que el anterior, para comunicarse con el PC utiliza el puerto serie, y de él toma las tensiones necesarias. Tiene, por tanto, el mismo problema que el JDM2 al utilizarlo con ordenadores portátiles. Es compatible con el software ICPROG 1.4 y permite programar un montón de modelos de PICs, incluidos 16F8x, 16F62x y 16F87x.

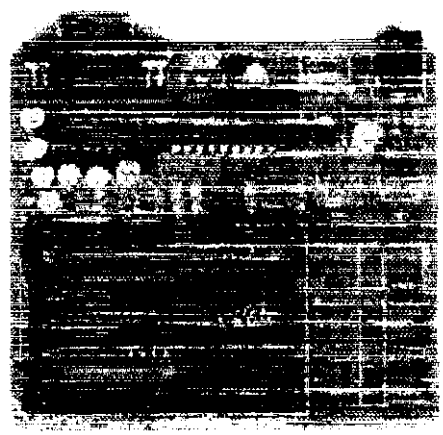


Figura 3.1-2 Programador PIPO2

El tercero programador se ha llamado PP2, y está basado en el ProPic 2. Integra su propia fuente de alimentación y utiliza un puerto paralelo (de impresora) para comunicarse con el PC, por lo que funciona correctamente en cualquier clase de PC, incluídos portátiles. Es compatible con ICPROG 1.4 permite programar prácticamente todos los PICs de las series 12C, 16C y 16F, tanto en su zócalo como a través de ICSP (*In Circuit Serial Programming* – programación serie en el propio circuito). También pude programar memorias 24Cxxx y 24LCxxx sobre su zócalo y muchos otros dispositivos mediante adaptadores.

Figura 3.1-3 Programador RRG8X

El programador PRG8x aqui presentado posee las siguientes caracteristicas:

- Conexión al puerto paralelo, Soporte para los modelos de PIC's 16F83, F84 y C84.

- Un programador de estas características sin gabinete ni fuente de alimentación ronda aproximadamente los 120 dólares en Argentina. El PRG8X se comercializa con un CD-ROM conteniendo las hojas de datos de todos los modelos de PIC's (Ideal para los aprendices).

Utilidades de programación (MPLAB, Compilador C, etc.) y hojas de aplicación, además del manual de uso del programador a un costo final en Argentina de $ 80 (Pesos ochenta) u 80 dólares.

Este es el ultimo programador CK-175N de microcontrolador que puede grabar los Pic de 18 pines que se programan seriamente, estos incluye los PIC 16C8X, 16F8X, 16C62X, 16C71X, y los de 8 pines de la familia 12C50X. Con este sistema no se pueden grabar los Pic de programación en paralelo, ellos son los PIC 16C54, 16C55, 16C56, 16C57, 16C58. Este programador se conecta al puerto paralelo de cualquier computadora Pc o compatible.

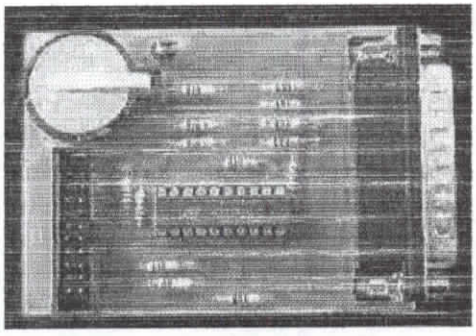


Figura 3.1-4 Programador CK-175N

Este quemador consta con su propio software que nos permite grabar los Pics y utiliza pocos componentes y es fácil de utilizar.

Análisis de los Programadores.

Después de haber analizado todos los programadores hemos elegido el Ck-175N por ser gratis y consta con su propio software para quemar el chip también pose pocos componentes y es de fácil manejo.

3.2 Características del quemador seleccionado

Este aparato se debe conectar al puerto paralelo de una computadora PC, utilizando el cable suministrado, el cual tiene un conector macho tipo <<D>> de 9 pines, que irá al programador.

El programador se alimenta mediante un adatador de corriente que debe entregar en su salida un voltaje entre 16 y 22VDC. La entrada del mismo se debe conectar a un tomacorriente adecuado (220VAC). Se debe prestar atención a la posición en que se encuentra el selector de voltaje de dicho adaptador, mediante el cual se puede seleccionar si la línea de alimentación es de 110 ó 220VCA.

El programador Ck-175 posee un conector (marcado J3), en el cual se encuentran todas las señales necesarias para grabar un microcontralor. Esto permite, si el usuario lo requiere, construri una adaptador con un socket o base para circuito integrado de 28 ó

40 pines, en la cual se puede colocar un microcontrolador con ese número de pines para ser programador. El software del Ch-175N permite grabar los Pic de 28 pines 16C62, 16C63, 16C72, 16C73 y los de 40 pines 165C64, 16C65, 16C74.

3.3 Diseño y Construcción del Programador

En la figura 3.3-2 se muestra la guía de ensamble de los componentes en el circuito impreso, los valores de los mismo se encuentran en la lista de materiales. Se debe poner especial cuidado cuando se hace la soldadura para no causar cortos entre puntos cercanos y también a la correcta ubicación de los componentes en el sitio indicado.

En la figura.3.1 tenemos la pista impresa del Hardware o Quemador.

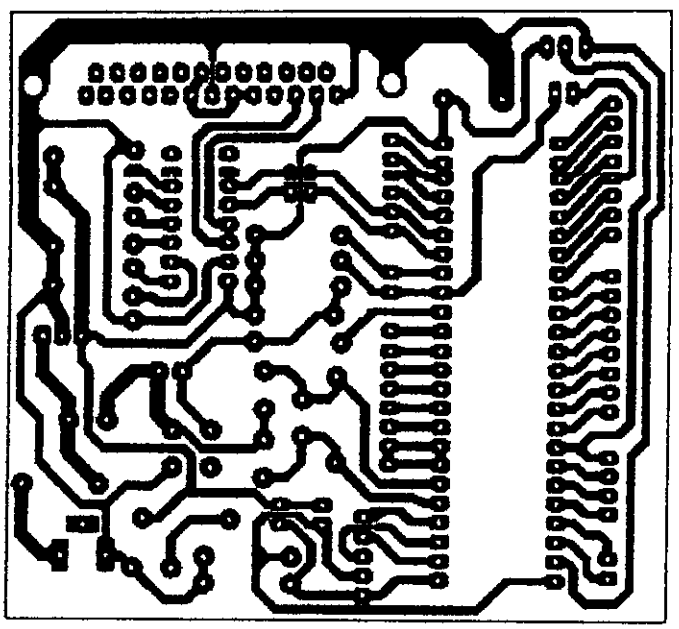


Figura 3.3-1 Diagrama del La Pista del Programador Ck-175

Pasos del ensamble

Antes de iniciar ensamble se debe revisar la lista de materiales para estar seguro de tener todos los elementos necesarios.

Los primeros componentes que se monta en la tarjeta son los de menor altura, como los puertos de alambre, las resistencias y los diodos. Debemos poner especial atención a la guía de ensamble para ubicar los en el sitio y la forma correcta. Asegúrese de que los componentes sean del valor adecuado revisando la lista de materiales. En el momento de hacer la soldadura verifique que no haya corto integrado y los condensadores pequeños.

El siguiente paso es ubicar los conectores y los componentes que falta. Una vez terminado el emsable del circuito impreso se debe limpiar la plaqueta para retirar los residuos del fúndente de la soldadura, para ello utilice un cepillo de dientes y alcohol.

Componentes para Ensamblar la Tarjeta

Los más importante el programador Ck-175 es que consta de pocos componentes para su creación como podemos ver en la figura 3.2-6.

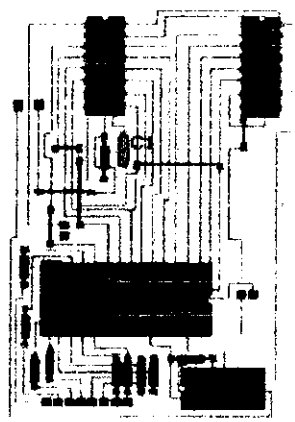


Figura 3.3-2 Diseño del Programador Ck-175

MATERIAL PARA ARMAR EL PROGRAMADOR	
Resistencias de 1/4W	
470	R10
1K	R6,R7,R11
10K	R3,R4,R5,R9,R8,R12

Condensadores cerámicos	
0.1	C2
1000	C1
Diodos	
1N4004	D1,D2,D3,D4,D5,D6
LED Rojo	Vpp
LED Verde	Vdd
Transistores	
2N3904	Q1,Q2
Circuito integrados	
7406	U1
lm7803	U4
lm7810	U3
Conectores y Accesorios	
Conector para el adaptador	J1
Conector DB9 hembra	J2
Conector en línea de 5 pines	J3
Base para circuito integrado	8 pines
Base para circuito integrado	14 pines
BaseZip para circuito integrado	18 pines
Alambre telefónico para puentes	10 cm
Soldadura	1 mt

Otros Elementos
Cable ribbon de 6 hilos
Conector DB9 macho con carcaza
Conector DB25 macho con carcaza
Tornillo milimétrico 3*15 con tuerca
Separador plástico de 8 mm
Topes o patas de caucho
Chasis CK-175N

Diseño del Cable Paralelo

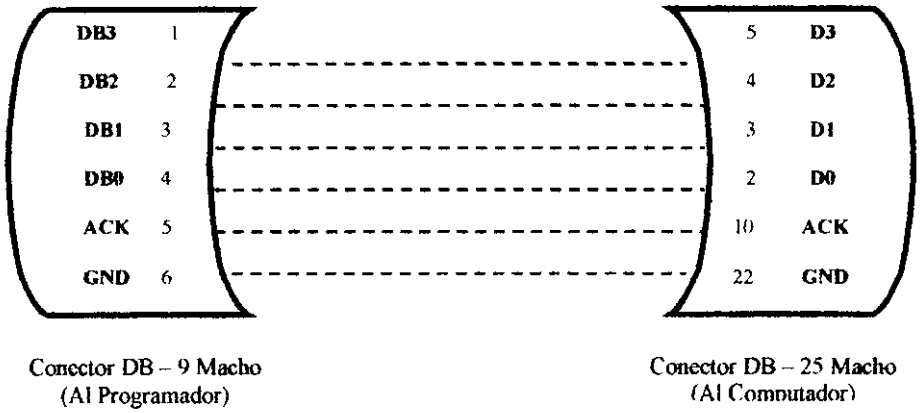


Figura 3.3-3 Diseño del Cable Paralelo Ck-175

Como el programador se conecta al puerto paralelo de cualquier computadora PC, se debe construir un cable con las conexiones que se muestran en la figura 3.3-3. Una vez se ha terminado el ensamble de la tarjeta y se haya comprobado el funcionamiento del circuito, esta se debe montar sobre el chasis.

Para ello se utilizan tornillos con tuerca y separadores plásticos de al menos 8 mm, eso garantiza que los puntos de soldadura no toquen la lámina metálica.

Operación

El procedimiento para grabar los microcontroladores es el siguiente:

- Una vez se tiene el programa ensamblado y con extensión. Hex, esto se debe pasar a la memoria del microcontrolador por medio del programador Ck-175N.

- Antes de iniciar el proceso se debe revisar que el cable que va entre el puerto paralelo de la computadora y el programador se encuentre bien conectado.

- El adaptador de corriente que sirve para alimentar el circuito debe estar conectado y entregando el voltaje adecuado. Uno de los LED o ambos

pueden encenderse. No inserte ni remueva un microcontrolador cuando cualquiera de los LED esté encendido.

- Los socket o base para circuito integrado debe estar vacío para evitar que el micro sufra posibles daños en caso de que exista un error de ensamble.

- Escriba el comando prog y oprima la tecla Enter.

- En ese momento debe aparecer el pantallazo del programa que graba los microcontroladores Pic.

- Los primero que se debe hacer es coger el tipo de microcontrolar que se está

Diseño de la Aplicación

En este capitulo hablaremos acerca del diseño, configuración y programación del microcontrolador.

3.4 Diagrama y Diseño del Circuito

El teclado que utilizaremos es de 4*3 es decir consta de 4 filas y 3 columnas en la fig. 3.4-1 se muestra un teclado de 11 elementos, precisamente solo utiliza 7 líneas del microcontrolador correspondientes a las filas se han configurado como salidas y las columnas como entradas.

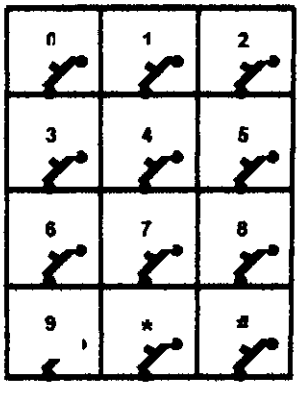


Fig. 3.4-1 Teclado matricial de 4 filas * 3 columnas

Como se puede observar en la fig. 3.4-2 normalmente, las líneas de entrada permanecen en un nivel lógico alto, gracias a los elementos *pull-up*.

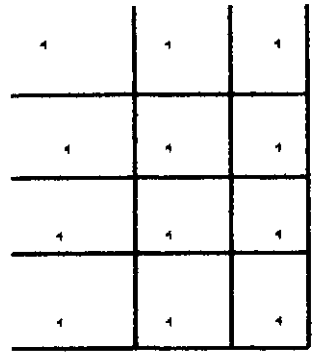


Fig. 3.4-2 columnas como entradas

La clave para manejar este tipo de teclados consiste en enviar por las líneas de salidas sólo un cero por vez por ejemplo si enviamos un cero por la línea RA1, cuando oprimimos una tecla de la segunda fila (el 3,4,5), un nivel lógico bajo se reflejará en el *Pin* correspondiente de las líneas de entrada RBO,RB1,RB2 respectivamente así, si se encuentra un nivel lógico bajo en la RB2 podemos concluir que la tecla presionada fue el 2.

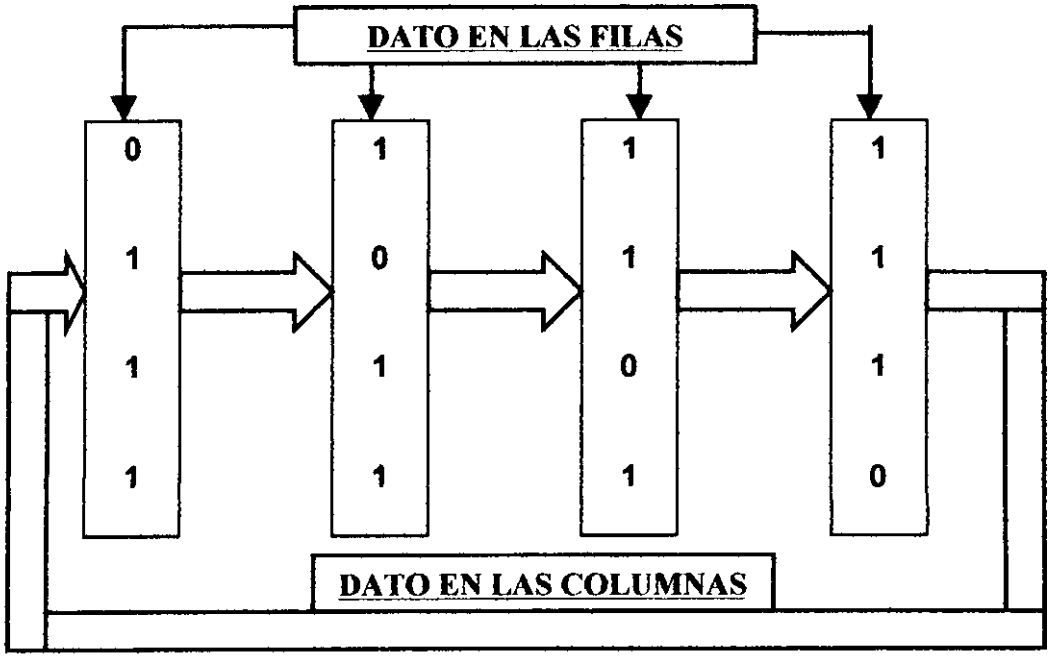


Fig. 3.4-3 Secuencia para la lectura de un teclado matricial

Si queremos explorar todo este teclado, bastará con rotar el cero circularmente, de tal manera que solamente un cero se encuentre en las filas del teclado, cuando se realiza las lectura de las líneas de entradas como se muestra en la fig. 3.4-3 Cuando el cero llegue a la fila más significativa del teclado, debe reingresar en la próxima ocasión por la menos significativa, reincidiendo la exploración del teclado.

El proceso se realiza a una gran velocidad, por lo que se tiene la sensación que todo el teclado se está censando permanente. En el programa realizado, por ejemplo, la exploración total del teclado tarda menos de 60μs, si consideramos que el oscilador es de 4MHZ.

Otros aspecto que no se puede olvidar, son los rebotes causados por la pulsación de una tecla. Cuando una tecla se oprime, sus contactos actúan como resortes, y la unión eléctrica no es estable; se generan una serie de uniones y desuniones mecánicas durante un intervalo significativo de tiempo.

Estos rebotes pueden dar lugar a que, en una aplicación real, el programa los interprete como si se hubieran generado muchas pulsaciones, si es que no se toman los correctivos del caso. Para ello existen soluciones de hardware y software, consideramos más interesantes las segundas ya que simplifican el diseño. Allí, la solución más obvia es que después de la detección de la tecla pulsada se genera un retardo en la lectura del teclado, de tal manera que se ignoren los contactos subsiguientes debidos a los rebotes.

Después haber analizado el funcionamiento de teclado matricial 4x3 mostraremos en la Fig. 3.4-4 el diseño y configuración las líneas del microcontrolador conectados al teclado.

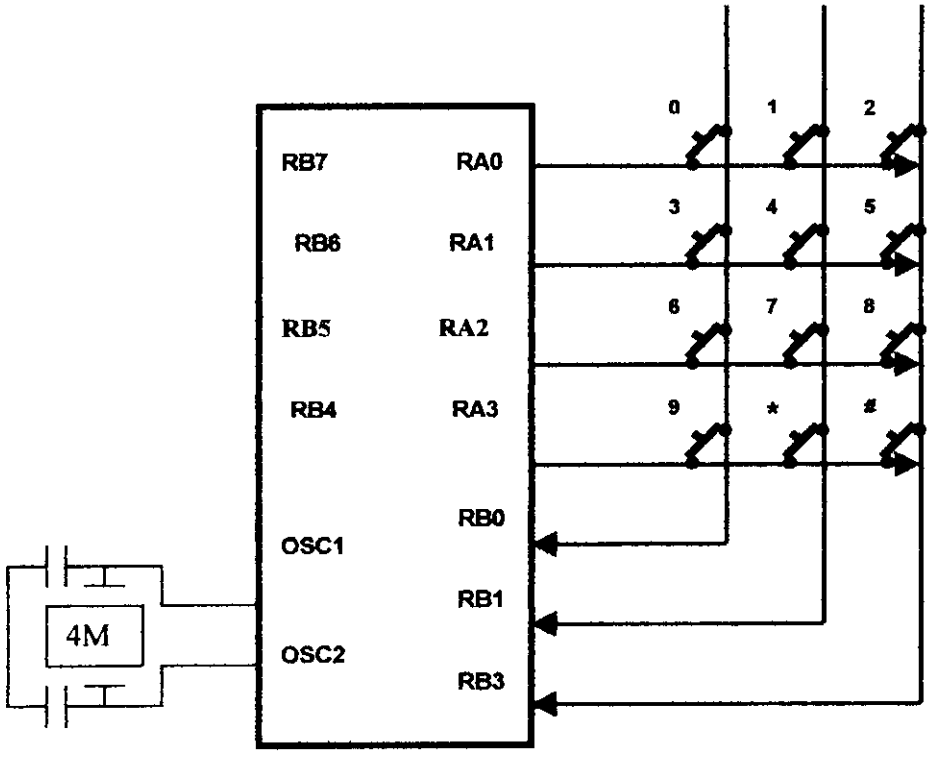


Fig. 3.4-4 Diagrama y Configuración del Pic 16c84

En la fig.3.4-4 nos muestra la configuración del microcontrolador como RA0,RA1,RA2,RA3 son utilizados como salidas, RB0,RB1,RB2 son configurados como entradas y para la alarma se utiliza el pin RB7 para la chapa el pin RB6, tenemos como cambio de clave RB5 y por ultimo RB7 como error en todos esto pines están conectados unos led que me indica las decisiones o resultado del microcontroladores.

En la siguiente fig.3.4-5 tenemos la configuración de los led a utilizar como salidas o resultados del microcontrolador.

DISEÑO DE LOS LED	
●	**Rojo se prende cuando se activa la Alarma**
●	**Verde se ilumina cuando se abre la Puerta**
●	**Amarillo activa cuando se cambia la clave**
○	**Transparente se enciende cuando existe un error**

Fig. 3.4-5 Configuración del los Led

En la figura 3.4-6 tenemos la pista impresa de nuestro sistema de seguridad de control de accesos de entradas y salidas.

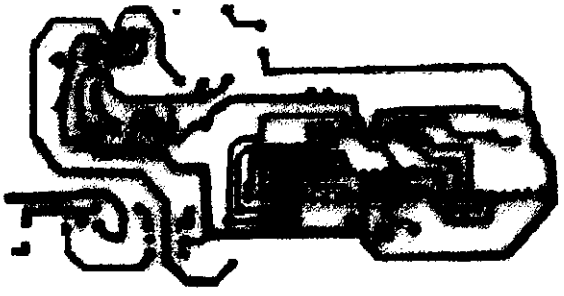
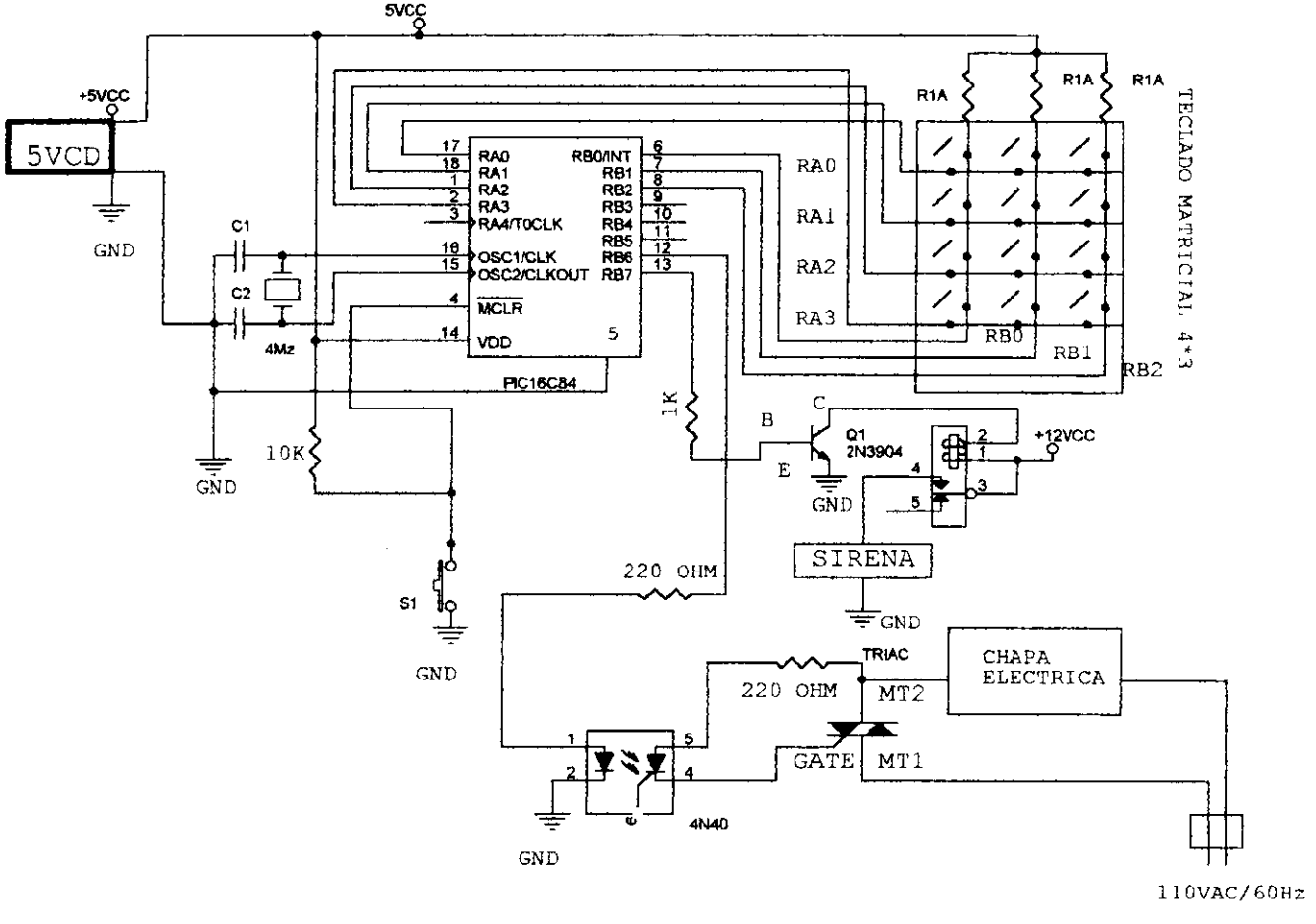


Fig 3.4-6 Pista del sistema de seguridad

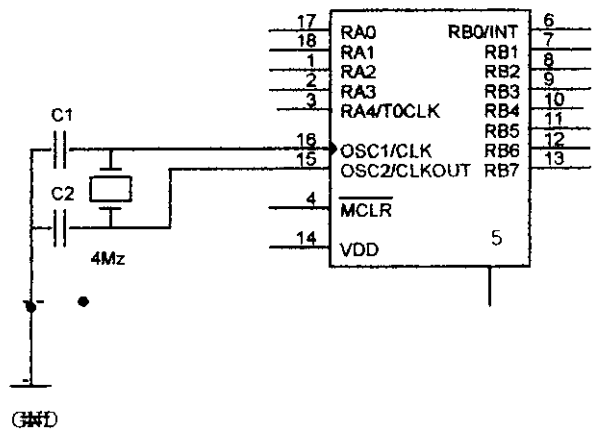
3.5 Montaje del Circuito Integrado



3.5.1 Conexión de un oscilador a cristal.

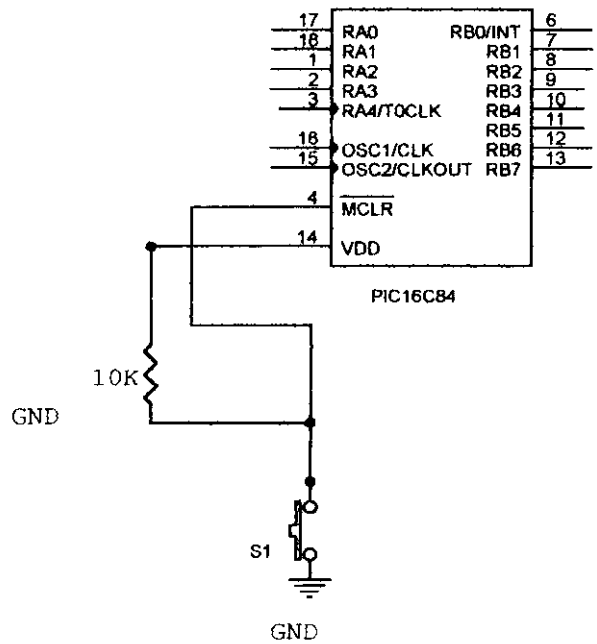
En este proyecto se esta usando un oscilador de cristal de 4 Mhz por que este oscilador me permite crear bases de tiempo de un segundo muy precisas.

Este reloj o oscilador esta conectada a los pines 15 y 16 del microcontrolador también utiliza dos condensadores de 20 pico faradios para tener una mejor frecuencia.

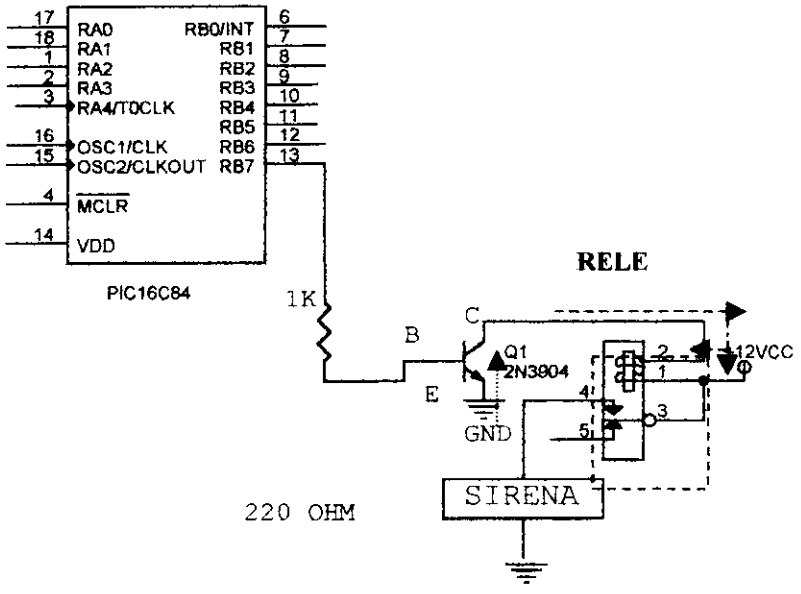


3.5.2 Conexión del Pin Reset

Se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando se necesario y utilizamos el pin 4 del microcontrolador.



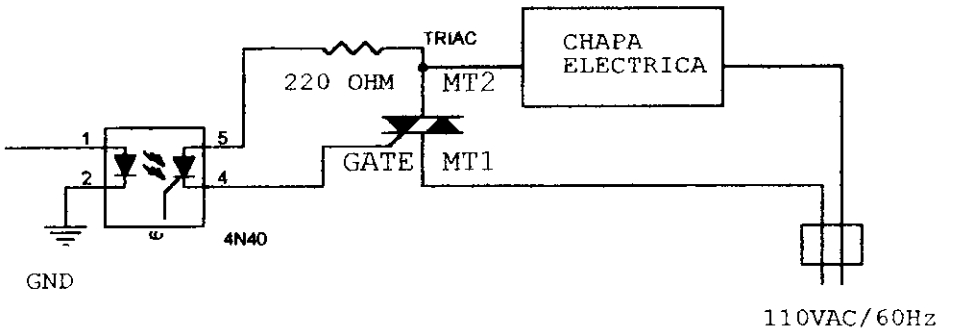
3.5.3 Conexión de una alarma con una sirena



La alarma como podemos observar en el diagrama se activa en el momento que el pic detecto que los dígitos ingresados por el teclado no fueron los correctos entonces envía un alto por el pin 13(RB7) para lo cual utilizamos una interface para el control de la sirena: la bobina del relé(pin1 y 2) necesita que se energize, para que los contactos se cierren(pin 3 y pin4) y circule la corriente hacia la sirena y hacia quede activada hasta el momento que en RB7 tenga un nivel lógico bajo(0).

Al momento que tengo un nivel lógico alto en RB7 este polariza a la base del transistor y la corriente fluye a través del emisor hacia el colector(circula gnd) hacia el un extremo de la bobina y como el otro extremo de la bobina esta conectada a +12vcd la bobina se energiza y crea un campo magnético al igual que un imán logrando hacia que los contactos metálicos que tiene internamente dentro del mismo encapsulado se cierren cuando hay presencia de voltaje y viceversa haciendo posible el flujo de corriente a través del contacto(pin3 y pin 4 se unen)circulando la corriente necesaria para que la sirena se active.

3.5.4 Instalación de la Chapa



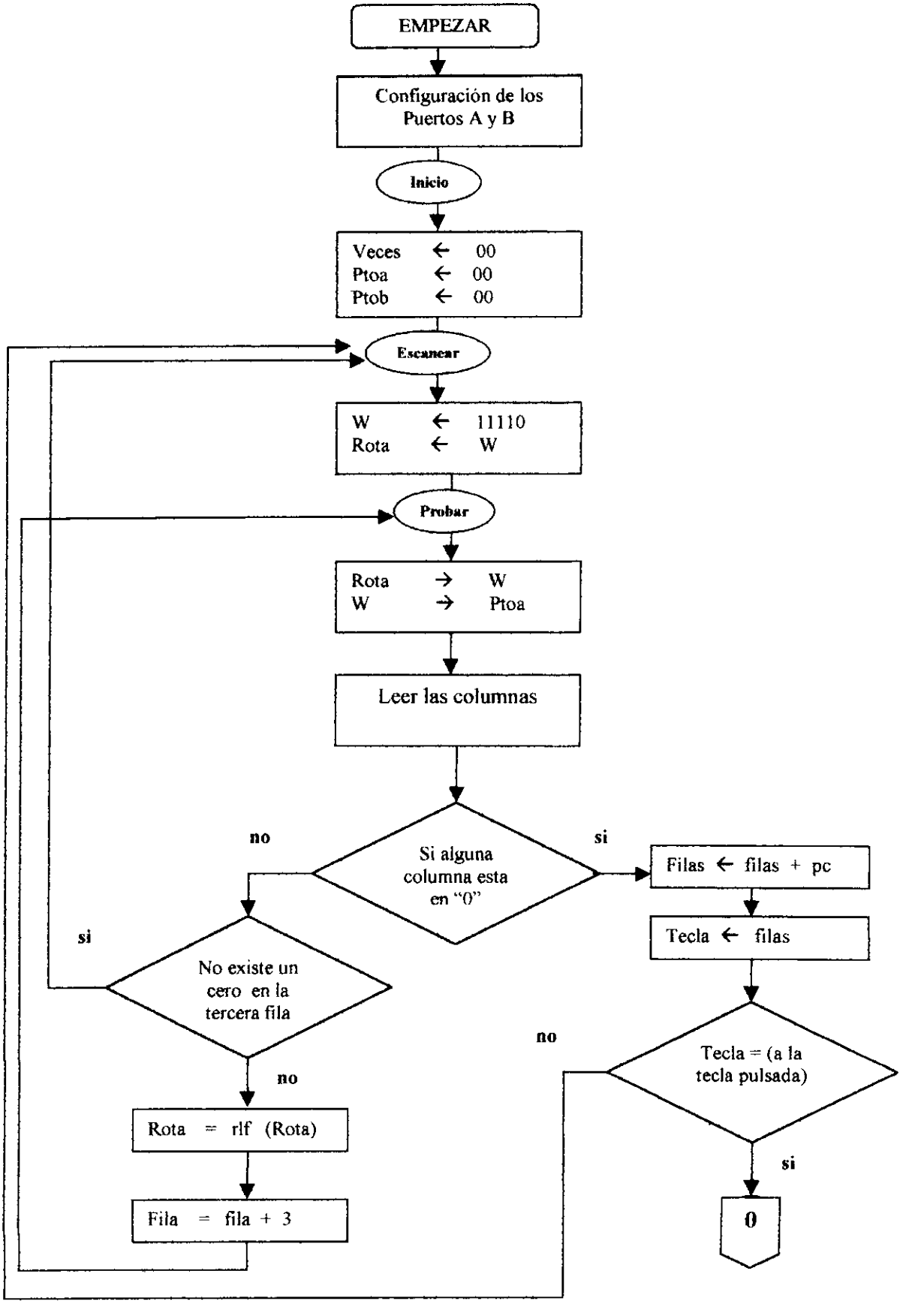
El Pic nos envía un nivel lógico alto por el pin RB6 pero este nivel lógico alto tan solo activaría un led y para que se accione la cerradura eléctrica necesitamos 110VAC, para la cuál utilizamos una interface que sirve como acoplamiento independiente entre ambos Voltajes.

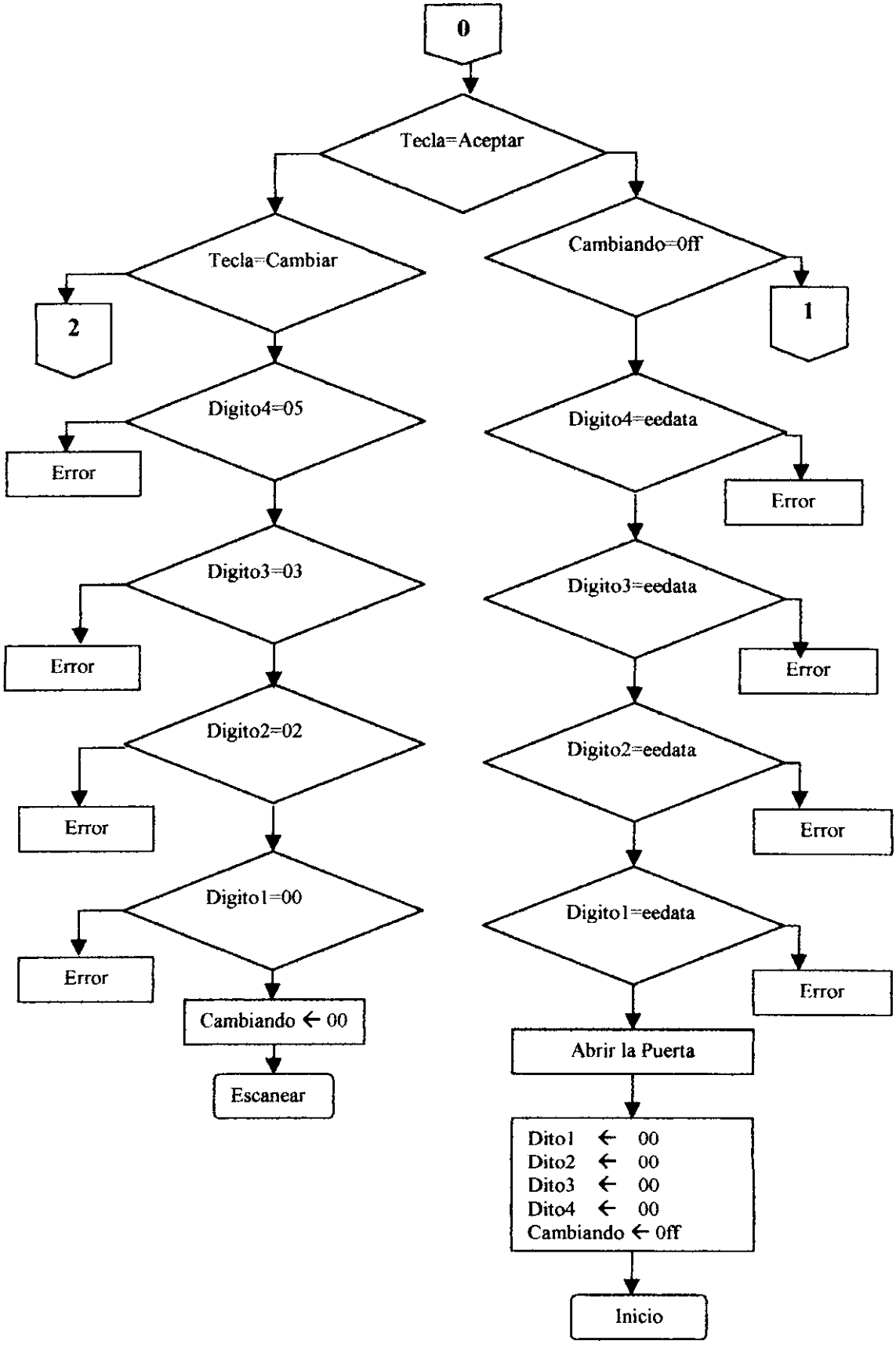
El nivel lógico alto proveniente del pin 12(RB6) como muestra el gráfico hace posible que se encienda el led infrarrojo que esta internamente en encapsulado enviando así una seña al receptor de este(pin4 y pin5) este que a la ves nos envía un pulso necesario de corriente a la compuerta del Triac, haciendo circular la corriente a través de MT1 y MT2, (se comportan como un solo alambre) hacia la cerradura eléctrica haciendo posible que esta se abra.

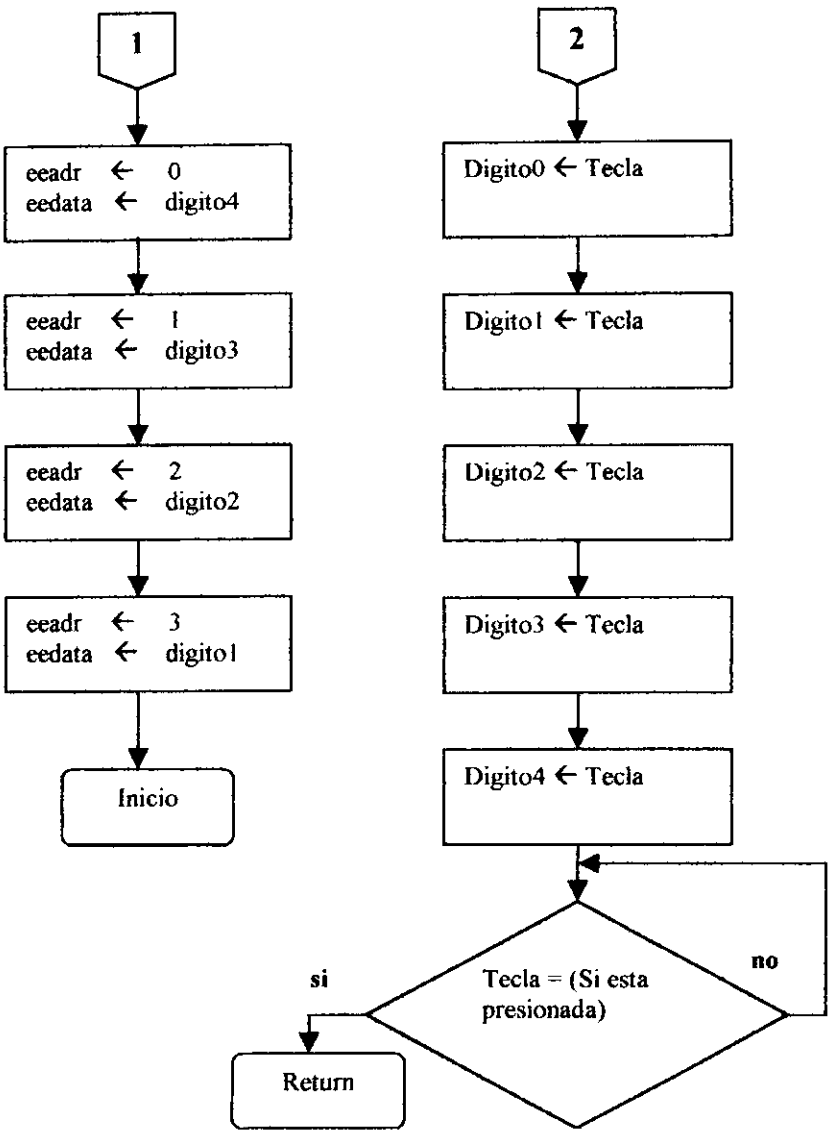
3.6 Programación del Microcontrolador.

Este programa controla una chapa eléctrica accionada por la introducción correcta de una clave de acceso, la cual esta guardada en la EEPROM de datos del microcontrolador.

3.6.1 Diagrama de flujo sobre el sistema de seguridad




0
Tecla=Aceptar
Tecla=Cambiar
2
Digito4=05
Error
Digito3=03
Error
Digito2=02
Error
Digito1=00
Error
Cambiando ← 00
Escanear
Cambiando=0ff
1
Digito4=eedata
Error
Digito3=eedata
Error
Digito2=eedata
Error
Digito1=eedata
Error
Abrir la Puerta
Dito1 ← 00
Dito2 ← 00
Dito3 ← 00
Dito4 ← 00
Cambiando ← 0ff
Inicio



3.6.2 Programa del sistema de Chapa Eléctrica

Se está asignando el nombre pc al valor 02, que correspondo a la dirección de la memoria como es el contador de programa se incrementa en uno con cada instrucción.

```
pc      equ     02h
```

El nombre status esta con la dirección 03 se refiere al registro de estado nos permite seleccionar la pagina para el direccionamiento de la memoria también nos permite saber los acarreo que existan en las operaciones aritméticas.

```
status  equ     03h
```

Los puertos del microcontrolador se los asignan a los nombres ptoa y ptob con la dirección 05h y 06h de la memoria ram.

```
ptoa    equ    05h
ptob    equ    06h
```

Registro de datos de la memoria EEPROM nos permite realizar operaciones de escritura, lectura y direccionamiento de la EEPROM de datos.

```
eedata  equ    08h
eeadr   equ    09h
```

Son registros que almacenaran los datos que ingresaran por el teclado hacia el microcontrolador.

```
digito0   equ    0ch
digito 1  equ    0dh
digito 2  equ    0eh
digito 3  equ    0fh
digito 4  equ    10h
```

Este registro se incrementa ó cuenta hasta 4 veces para activar el sistema de alarma si existiera error al ingresar la clave

```
veces   equ    11h
```

Me permite crear lazos para el retardo de tiempo por medio de estos registro que serán utilizados como variables constantes.

```
loops   equ    12h
loops2  equ    13h
seg1    equ    14h
seg10   equ    15h
```

Este registro es utilizada para almacenar momentáneamente el tecla presionada para ser comparada.

```
tecla    equ     16h
```

Es un registro que se almacenara todas las operación de rotamiento.

```
rota     equ     17h
```

En este registro se utilizara para incrementar las filas.

```
filas    equ     18h
```

Con este registro puedo saber cuando puedo cambiar la clave de 4 digitos.

```
cambiando equ    19h
```

Sirve como auxiliar del registro tecla es decir se almacena el valor de la tecla presionada.

```
tecla1   equ     1ah
```

Registro de configuración y control me permite poner como salida o entrada a los puertos A y B.

```
trisa    equ     85h
trisb    equ     86h
```

registro eecon1 es el que controla la memoria EEPROM de datos mientras eecon2 es el registro auxiliar para control de la memoria.

```
eecon1   equ     88h
eecon2   equ     89h
```

bandera del registro de estados se coloca en 1 cuando el resultado de una operación lógica o aritmética es cero.

```
z       equ     02h
```

Se registro se utiliza como auxiliar y se almacena o indica que el resultado se guarda en W.

```
w       equ     00h     ;
```

bandera de carry esta instrucción aritmética se activa cuando se presenta acarreo desde el bit más significativo del resultado tiene direccionamiento 00 del registro estatus.

```
c       equ     00h
```

Bandera de finalización de la escritura con el direccionamiento 04 del registro EECON1.

```
eeif    equ     04h
```

Esta bandera de error de escritura con la dirección 03 del registro EECON1.

```
wrerr   equ     03h
```

Es utilizada para habilitar la escritura en la memoria EEPROM con la dirección 02 del registro EECON1.

```
wren    equ     02h
```

Me permite saber cuando se inicia la escritura en la memoria

```
wr      equ     01h
```

Para finalizar la escritura de la memoria.

```
rd      equ     00h
```

Programa principal

Especifica el vector de reset del microcontrolador, para escribir la instrucción que le indica al micro que debe ir al inicio .

```
reset   org     0
goto    inicio
```

Se usa para definir el sitio o dirección en donde el programa empieza en la memoria.

```
        org     5
```

Se muestra la rutina de retardo es de 100 milisegundo, esta es utiliza por la gran velocidad con la cual se ejecutan las instrucciones en el microcontrolador.

```
retardo
        movlw   D'100'
        movwf   loops
top2    movlw   D'110'
        movwf   loops2

top     nop
        nop
        nop
        nop
        nop
        nop
```

Esta instrucción decrementa un registro y consulta si el contenido de éste ha llegado a cero; si es así, omite la siguiente instrucción; si no lo es, la ejecuta.

```
        decfsz  loops2
        goto    top
        decfsz  loops
```

```
goto    top2
retlw   0
```

Esta rutina nos permite leer o recuperar el dato guardado en la memoria EEPROM y luego realizar una comparación para ver si es correcto del dato.

clave

Limpia el registro eeadr para que comience en la dirección 00.

```
clrf    eeadr
```

Se ubica en la pagina 1 del registro estatus de la Memoria RAM

```
bsf     status,5
```

Apunta a la dirección 00 del registro EECON1 para iniciar la lectura

```
bsf     eecon1,rd
```

Vuelve al primer banco de memoria 0

```
bcf     status,5
```

Este registro contiene el dato que se escribió en la memoria y lo pasa al registro W

```
movf    eedata,w
```

Realiza una operación OR para ver si el registro W es igual a digito4

```
xorwf   digito4,w
```

Si el dato de Z es igual a "0" va a la rutina de Error caso contrario salta una línea.

```
btfss   status,z
goto    error
```

Incrementa un valor al registro eeadr a 01 indicando que se ubique en dicha posición.

```
incf    eeadr,1
```

Otra vez se ubica en la pagina 1 del registro STATUS de la Memoria RAM

```
bsf     status,5
```

Se direcciona en 00 del registro EECON1 para iniciar la lectura.

```
bsf     eecon1,rd
```

Regresa al primer banco o pagina de memoria 00

```
bcf     status,5
```

Contiene el segundo dato que se escribió en la memoria y lo pasa al registro W

```
movf    eedata,w
```

Realiza una operación OR para ver si el registro W es igual a digito3

```
xorwf   digito3,w
```

Si el dato de Z es igual a "0" va a la rutina de Error caso contrario salta una línea.

```
btfss   status,z
goto    error
```

Incrementa un valor al registro eeadr a 02 indicando que se ubique en dicha posición.

```
incf    eeadr
```

Se ubica en la pagina 1 del registro STATUS de la Memoria RAM

```
bsf     status,5
```

Apunta a la dirección 00 del registro EECON1 para iniciar la lectura del otro dato.

```
bsf     eecon1,rd
```

Vuelve al primer banco de memoria 0

```
bcf     status,5
```

Contiene el tercer dato que se escribió en la memoria y lo pasa al registro W

```
movf    eedata,w
```

Realiza una operación OR para ver si el registro W es igual a digito2

```
xorwf   digito2,w
```

Si el dato de Z es igual a "0" va a la rutina de Error caso contrario salta una línea.

```
btfss   status,z
goto    error
```

Incrementa un valor al registro eeadr a 03 indicando que se ubique en dicha posición.

```
incf    eeadr
bsf     status,5
```

Regresa a la dirección 00 del registro EECON1 para iniciar la lectura del otro dato.

```
bsf     eecon1,rd
```

Regresa al primer banco o pagina de memoria 00

```
bcf     status,5
```

Contiene el cuarto dato que se escribió en la memoria y lo pasa al registro W

```
movf    eedata,w
```

hace una operación OR para ver si el registro W es igual a digito1

```
xorwf   digito1,w
```

Si el dato de Z es igual a "0" va a la rutina de Error caso contrario salta una línea.

```
btfss   status,z
goto    error
```

Si los dígitos son correctos le manda un pulso por el puerto RB6 indicando que abra la chapa eléctrica.

```
bsf     ptob,6
```

Esta instrucción llama a la rutina de retardo para que realice un tiempo de espera

```
call    retardo
```

Limpia el registro veces, ptoa y lo pone en 00 por medio de la instrucción clrf

```
clrf    veces
clrf    ptoa
```

Apaga la chapa eléctrica mandando un "0" por el puerto RB6

```
bcf     ptob,6
```

Igual que el otro limpia los registros digito1,digito2,digito3,digito4, ptoa y lo pone en 00 por medio de la instrucción clrf

```
clrf    digito1
```

```
clrf    digito2
clrf    digito3
clrf    digito4
return
```

Error

Esta rutina se ejecuta cuando existe un error al ingreso de la clave o ingreso del código activando la alarma.

Envía un "1 " por el puerto RB4 indicado que existió un error

```
bsf     ptob,4
```

Esta instrucción incrementa el numero de veces

```
incf    veces,1
```

Mueve el valor o contenido del registro Veces al registro W

```
movf    veces,w
```

Realiza una operación de comparación para ver sí fue el cuarto intento

```
xorlw   04h
```

Pregunta si el resultado fue verdadero activa la alarma caso contrario llama a un retardo de tiempo.

```
btfsc   status,z
goto    alarma
call    retardo1
```

Regresa al programa principal es el escanear otro valor

```
goto    escaniar
```

Escribir

Esta rutina ESCRIBIR me permite escribir o ingresar valores a la memoria EEPROM para luego ser almacenadas.

Pone un "0" es decir manda un bit bajo al puerto RB3 indicando que se esta escribiendo la clave.

```
bcf     ptob,3
```

Limpia el registro eeadr para que comience en la dirección 00.

```
clrf    eeadr
```

Mueve el valor digito4 al registro W por medio de la instrucción Movf

```
movf    digito4,w
```

Lo que esta en registro w lo escribe en eedata.

```
movwf   eedata
```

Regresa a la pagina1 y se ubica en el banco de la memoria Ram

```
bsf     status,5
```

Habilita escritura en memoria EEPROM ubicándose en el registro Eecon1

```
bsf     eecon1,wren ;
```

Se asegura que la bandera este en cero

```
bcf     eecon1,eeif
```

Para escribir en la memoria de datos EEPROM

```
movwf   eecon2
movlw   0aah
movwf   eecon2
```

Escribir el dato que se cargo previamente en el registro eedata en la posición de memoria dirreccionada por eeadr

```
bsf     eecon1,wr
```

Llama a la rutina de espera.

```
call    espera
```

Incrementa la posición de la memoria EEPROM y se ubica allí.

```
incf    eeadr,1
```

Mueve lo que tenga el digito3 al registro W por medio de la instrucción movf.

```
movf    digito3,w
```

El dato que tiene el registro W lo pasa al registro eedata para ser escrita en la memoria EEPROM.

```
movwf   eedata
```

Ubica en el segundo banco de RAM es decir el la pagina 1

```
bsf     status,5
```

Habilita la escritura en memoria EEPROM.

```
bsf     eecon1,wren
```

Se asegura que la bandera este en cero

```
bcf     eecon1,eeif
```

Lo que esta en el registro W se lo pasa al registro eecon2

```
movwf   eecon2
```

Se vuelve a cargar al registro W con el valor de 0aah

```
movlw   0aah
```

Y lo esta en el registro W se lo retorna al registro eecon2

```
movwf   eecon2
```

Escribir el dato que se cargo previamente en el registro eedata en la posición de memoria dirreccionada por eeadr

```
bsf     eecon1,wr
```

Llama a la rutina de espera.

```
call    espera
```

Incrementa la posición 01 de la memoria EEPROM y se ubica allí.

```
incf    eeadr,1
```

Mueve lo que tenga el digito2 al registro W por medio de la instrucción movf.

```
movf    digito2,w
```

El dato que tiene el registro W lo pasa al registro eedata para ser escrita en la memoria EEPROM.

```
movwf   eedata
```

Ubica en el segundo banco de RAM es decir el la pagina 1

```
bsf     status,5
```

Habilita la escritura en memoria EEPROM.

```
bsf     eecon1,wren
```

Se asegura que la bandera este en cero

```
bcf     eecon1,eeif
```

Lo que esta en el registro W se lo pasa al registro eecon2

```
movwf   eecon2
```

Se vuelve a cargar al registro W con el valor de 0aah

```
movlw   0aah
```

Y lo esta en el registro W se lo retorna al registro eecon2

```
movwf   eecon2
```

Escribir el dato que se cargo previamente en el registro eedata en la posición de memoria direccionada por eeadr

```
bsf     eecon1,wr
```

Llama a la rutina de espera.

```
call    espera
```

Incrementa la posición 02 de la memoria EEPROM y se ubica allí.

```
incf    eeadr,1
```

Mueve lo que tenga el digito1 al registro W por medio de la instrucción movf.

```
movf    digito1,w
```

El dato que tiene el registro W lo pasa al registro eedata para ser escrita en la memoria EEPROM.

```
movwf   eedata
```

Ubica en el segundo banco de RAM es decir el la pagina 1

```
bsf     status,5
```

Habilita la escritura en memoria EEPROM.

```
bsf     eecon1,wren
```

Se asegura que la bandera este en cero

```
bcf     eecon1,eeif
```

Lo que esta en el registro W se lo pasa al registro eecon2

```
movwf   eecon2
```

Se vuelve a cargar al registro W con el valor de 0aah

```
movlw   0aah
```

Y lo esta en el registro W se lo retorna al registro eecon2

```
movwf   eecon2
```

Escribir el dato que se cargo previamente en el registro eedata en la posición de memoria direccionada por eeadr

```
bsf     eecon1,wr
```

Llama a la rutina de espera.

```
call    espera
```

Pone un bit bajo al puerto RB5 para indicar que ya termino de escribir

```
bcf     ptob,5
```

Regresa al inicio del programa

```
goto inicio
```

La rutina de espera me permite habilitar o desabilitar la escritura hacia la memoria EEPROM.

Espera

Consulta para ver si finalizo la escritura en la memoria EEPROM Si termino salta una linea.

```
btfss   eecon1,eeif ;
goto    espera
```

Pone un bit bajo al registro eeif es decir finaliza la escritura.

```
bcf     eecon1,eeif
```

Desabilita la escritura en la memoria ubicando un "0" en el registro Wren

```
bcf     eecon1,wren
```

Ubica un cero en el registro Status para ir a la pagina 0 de la memoria Ram.

```
bcf     status,5
```

Retorna el Valor a donde lo llamaron

```
Return
```

Esta rutina es la principal porque nos permite configurar a los puertos es donde comienza el programa nuestro.

Inicio

Se ubica un "1" en el registro Status para ir a la pagina 1 y poder configurar los puertos como entrada y salida.

```
bsf     status,5
```

Se guarda en el registro W con el valor 00

```
movlw   00h
```

Se programa o configura al puerto A como salidas cargándolo con el registro W que tiene el valor 00

```
movwf   trisa
```

Se carga en el registro W con el valor 07

```
movlw   07h
```

Los Pines del puerto B se configuran como entradas cargándolo con el registro W que tiene el valor 07.

```
movwf   trisb
```

Se regresa al registro Status para ir a la pagina 0 de la memoria RAM.

```
bcf     status,5
```

Se carga al registro W con el valor 0ffh

```
movlw   0ffh
```

Se almacena este valor al registro cambiando

```
movwf   cambiando
```

Se borrar todo lo que tenga los registros veces, ptoa y ptob.

```
clrf    veces
clrf    ptoa
clrf    ptob
```

Escaniar

Pone al registro filas en 0

```
clrf    filas
```

Se prepara para enviar ceros a las filas

```
movlw   b'11110'
```

Lo que tiene el registro W lo pasa al registro Rota

```
movwf   rota
```

Probar

Almacena el dato que tiene el registro rota a W

```
movf    rota,w
```

Envía el dato que tiene el registro W al puerto A

```
movwf   ptoa
```

Esta instrucción de hace nada solo da un salto de tiempo

```
nop
nop
nop
nop
```

Leer

Lee las columnas conectadas al puerto B y lo almacena en el registro W

```
movf    ptob,w
```

Elimina la parte alta del byte leído por medio de una operación And

```
andlw   07h
```

Invierte el dato para ver si hay algún cero.

```
xorlw   07h
```

Pregunta si el resultado es cero entonces existe alguna tecla presionada

```
btfss   status,z
goto    dato
```

Consulta si ya van 3 rotaciones en las filas si existiera regresara al escaneo caso contrario sigue rotando

```
btfss   rota,3
goto    escaniar
```

Coloca bit de carry en 1 para rotar un bit

```
bsf     status,c
```

Para rotar el 0 que va a ir hacia las filas

```
rlf     rota
```

Carga W con 3 para sumarlo al valor de filas

```
movlw   3
```

Suma el valor cargado en W a filas

```
addwf   filas,1
```

Va hacia la rutina probar

```
goto    probar
```

Para obtener valor de la columna

```
call    tabla
```

Suma el valor de la columna a registro filas

```
addwf   filas,w
```

Almacena el valor pulsado en el registro Tecla

```
andlw   0fh
movwf   tecla
```

El valor tecla regresa al registro W

```
movf    tecla,w
movwf   tecla1
```

Utiliza un retardo de tiempo

```
call    retardo
```

Utiliza una auxiliar para guardar el valor de tecla

```
movf    tecla1,w
```

Consulta para saber si el valor es el mismo que pulsaron si lo es salta una línea

```
xorwf   tecla,w
btfss   status,z
goto    escaniar
```

Principal

Consulta para ver si la tecla pulsada es la tecla Aceptar

```
movf    tecla,w
xorlw   00bh
btfsc   status,z
goto    veinte
```

Me permite esta rutina ir a la rutina cambiar o ingresar el dato

Veinte1

```
movf    tecla,w
xorlw   00ah
btfsc   status,z
call    cambiar
call ingreso
goto escaniar
```

Esta rutina me permite consultar si la tecla Aceptar es para comparar o escribir la clave

Veinte

```
movf    cambiando,w
xorlw   0ffh
btfss   status,z
 goto escribir

call clave
goto veinte1
```

Se encarga esta rutina de almacenar los datos ingresados por el teclado

Ingreso

```
bcf   ptob,5
movf    tecla,w
movwf   tecla1
call    retardo
movf    tecla1,w
xorwf   tecla,w
btfss   status,z
```

```
goto    escaniar
movf    tecla,w
andlw   0fh
movwf   digito0
movf    digito3,w
movwf   digito4
movf    digito2,w
movwf   digito3
movf    digito1,w
movwf   digito2
movf    digito0,w
movwf   digito1
```

Si la tecla esta presionada se mantiene en este bucle

Siguepre

```
movf    ptob,w
xorlw   07h
btfsc   status,z
return
goto    siguepre
```

Esta rutina se encarga de activar la alarma se los intentos llego 4 manga una señal al microcontrolador indicando que active la alarma por RB7

Alarma

```
bsf    ptob,7
call   retardo2
bcf    ptob,7
clrf   veces
goto   inicio
```

```
retardo1 movlw  D'10'
         movwf  seg1
otro     call   retardo
         decfsz seg1
         goto   otro
         return
retardo2 movlw  D'10'
         movwf  seg10

otravez  call   retardo1
         decfsz seg10
         goto   otravez
         return
```

Me permite cambiar el nuevo código ingresado por el teclado

Cambiar

Se comparan los datos ingresados con el código de cambiar la clave si no es correcto se activa la rutina de error

```
         movf   digito4,w
         xorlw  05h
         btfss  status,z
         goto   error
         movf   digito3,w
         xorlw  03h
         btfss  status,z
         goto   error
         movf   digito2,w
         xorlw  02h
         btfss  status,z
         goto   error
         movf   digito1,w
```

```
xorlw  00h
btfss  status,z
goto   error
```

Se carga un "1" al puerto RB5 indicando el cambio de clave

```
bsf    ptob,5
```

Limpia el registro cambiando poniendo en 00

```
clrf   cambiando
```

Regresa a la rutina de escaneo para coger el valor nuevo para la clave

```
goto   escaniar
```

Esta rutina Tabla es donde permite saber cual tecla precio

Tabla

```
addwf   pc      ;sumar W al PC
nop
retlw   0       ;primera columna
retlw   1       ;segunda columna
nop
retlw   2       ;tercera columna
end             ;fin del programa
```

VALIDACIÓN Y COMPROBACIÓN

Ambato 18 de junio de 2003

Ing. Telmo Viteri

Director de la Escuela de Sistemas

De mis consideraciones:

Luego de expresarle un cordial saludo y desearle éxito en su dirección. Por medio de la presente, me complace informar a usted que luego de la demostración de funcionamiento del proyecto de tesis "Sistema de Seguridad para acceso de Puertas", presentado por el Sr. Fabián Esparza Silva con carné 647, encuentro, a mi criterio, que el mencionado proyecto merece aprobación y una especial felicitación por el trabajo de investigación.

Atentamente.

Ing. Santiago Acurio M.

DOCENTE

P.U.C.E
PROGRAMA AMBATO
RECIBIDO
SECRETARIA
FECHA 18-Junio-2003

Ambato junio 20, 2003

Ingeniero
Telmo Viteri
DIRECTOR ESCUELA DE SISTEMAS
Presente

De mi consideración:

En días anteriores se procedió a revisar la Disertación de Grado realizada por el señor Fabián Esparza sobre la implementación de una cerradura eléctrica, con sus seguridades de entrada/salida, este trabajo ha sido desarrollado minuciosamente y se nota que ha trabajado e investigado sobre electrónica y software para manejar este tipo de dispositivos.

Estimo que es un buen trabajo, el mismo que se encuentra probado y terminado para su presentación.

Por la atención que se sirva dar a la presente, le anticipo mis agradecimientos.

Atentamente,

Ing. Natasha Bayas
DOCENTE ESCUELA DE SISTEMAS

Ambato,12 de Junio del 2003

Ingeniero
TELMO VITERI
DIRECTOR DE LA ESCUELA DE INGENIERIA
EN SISTEMAS DE LA PUCESA.
Presente.-

De mi consideración:

Luego de la exposición del trabajo relacionado con sistemas de seguridad de puertas, de la tesis de grado presentada por el Sr. Fabián Esparza, comunico a Ud. Que el trabajo amerita ser avalizado y felicitado por su originalidad.

Es todo cuanto puedo certificar.

Atentamente

Ing. Washington Medina M:Sc.
DOCENTE FIS

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Los microcontroladores son una tecnología que ha revolucionado la vida cotidiana del hombre por su gran utilidad para el control y automatización de los procesos.

Una gran ventaja del trabajo con los microcontroladores vs. los microprocesadores es que gracias a su arquitectura; se reduce de forma drástica el diseño del circuito final, sin desmejorar el resultado final.

Los microcontroladores se suelen comparar con una minicomputadora porque posee todos sus componentes básicos; haciendo más fácil el manejo y diseño de circuito, y disminuyendo el costo.

El Mplab en comparación con las otras herramientas en el mercado es gratuito y funciona bajo plataforma Windows; permitiendo una programación gráfica y mas amigable.

Es posible realizar proyectos utilizando microncontroladores con un costo muy inferior al costo del mercado.

Es necesario adquirir un hardware que permita la quemada del chip y en nuestro caso fue construido como parte de la tesis.

El resultado obtenido durante el tiempo de esta investigación a llenado mi inquietud con respecto a la informática.

RECOMENDACIONES

Utilizar los microcontroladores para el control de procesos como alternativas más económica y rápida que con la utilización de microprocesadores.

Utilizar el Mplab como herramienta por la versatilidad de su interfaz.

Tomar precauciones con las patas del microcontrolador a la hora de conectar éste a la fuente de poder o al quemador.

Para el desarrollo de proyectos utilizando los microcontroladores se recomienda profundizar en los conocimientos de electrónica y programación a bajo nivel.

Incentivar a los estudiantes al estudio de los microcontroladores como otra alternativa para la solución problemas y para el control y automatización de los procesos.

Implementar este proyecto masivamente con la posibilidad de poner en el mercado un producto tan bueno como el que existe pero a mucho menor costo

BIBLIOGRAFIA

LIBROS:

Curso Básico de Microcontroladores PIC. (CEKIT)

Todo Sobre Pics (EDICION ELECTRONICA)

Manual de referencia del microcontrolador (NET .08 Digital DNA.)

INTERNET

http://www.microchip.com

http://cherokee.iespana.es/cherokee/Microbotica_MC_PIC.htm

http://www.micropic.arrakis.es/marcos.htm

http://www.cybernomo.com/scm/micros/pbasic.htm#Pro

http://lorca.umh.es/isa/es/temas/micros/doc/doc.html

http://wwww.microchip.com/10/Tools

http://www.microchip.com/10/lit/picmicro/index.html