Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA

C. B. I.

Proyecto De Ingeniería Electrónica

El PIC 16F84 en el desarrollo de un robot publicitario.

CATEDRÁTICO:

Maestro. Omar Lucio Cabrera Jiménez.

ALUMNO:

Zambrano Gómez Raymundo Alan.

Indice

Objetivos---------- 1
Alcances---------- 1
Introducción---------- 2
Conceptos de Robótica---------- 5
Breve historia del desarrollo de los robots---------- 7
Clasificación ---------- 9
Contexto actual de la ROBOTICA.---------- 11
La construcción de un ROBOT.---------- 13
Arquitectura de los ROBOTS.---------- 13
La Automatización---------- 16
Vida Artificial ---------- 18
Tarjeta controladora.---------- 20
Microcontrolador PIC---------- 22
El PIC 16F84---------- 22
Simuladores y Ensambladores:---------- 47
Programas---------- 55
Conclusiones---------- 66
Bibliografía---------- 67

Referencias WEB---------- 67

Objetivos:

Uno delos objetivos principales es el de mostrar lo que es un Robot Su funcionamiento, características y estructura, arquitectura.

Clasificándolos por su funcionamiento. Definiendo cada parte de sus componentes.

También tomamos como objetivo el desarrollo de un sistema de control mediante el microcontrolador PIC 16f84 de Microchip mostrando sus ventajas y características.

Alcances:

El alcance de este proyecto es mostrar lo que es un Robot y sus características, aplicaciones como en la industria, hogar, en la medicina, etc. Así como el funcionamiento de el PIC 16F84, mediante el diseño y armado de una tarjeta controladora.

El microcontrolador que en este caso trabajamos, nos permite realizar y diseñar diversas tarea o aplicaciones útiles y practicas, con un mínimo de espacio y a un bajo costo. Teniendo como resultado un mayor desarrollo en base a este microcontrolador.

Introducción

Introducción:

La aspiración del hombre por comprenderse a sí mismo y al mundo que lo rodea lo ha llevado a desarrollar un conjunto de disciplinas que tradicionalmente se denominan naturales, como son la biología, la química, las ciencias sociales por mencionar algunas. Al mismo tiempo y a medida que el conocimiento y el dominio sobre la naturaleza se han ido logrando, se han desarrollado herramientas y tecnologías que contribuyen a mejorar la concepción y comprensión del mundo, con ellas van generándose nuevas disciplinas que podemos denominar artificiales, como son la computación, la inteligencia artificial, la robótica, la vida artificial, entre otras.

El término artificial se asocia con las cosas hechas o producidas por el hombre en imitación de algo natural, de manera que en biología, en particular en el área de genética es común el uso de ese nombre para denominar la obtención, a través de síntesis, sustancias que emulan a las naturales, al grado que se producen desde sustancias vitales como la sangre. Sin embargo, al unirse el término artificial a las palabras inteligencia o vida generamos una actitud de rechazo, de escepticismo.

Dentro de la mitología griega se puede encontrar varios relatos sobre la creación de vida artificial, por ejemplo, Prometeo creó el primer hombre y la primer mujer con barro y animados con el fuego de los cielos. De esta manera nos damos cuenta de que la humanidad tiene la obsesión de crear vida artificial desde el principio de los tiempos. El hombre ha creado autómatas como un pasatiempo, con la finalidad de entretener a su dueño. Los materiales que se utilizaron, se encontraban al alcance de todo el mundo, esto es, utilizaban maderas resistentes, metales como el cobre y cualquier otro material moldeable, esto es, que no necesitara o requiriera de algún tipo de transformación para poder ser utilizado en la creación de los autómatas.

Muchos fueron los intentos por lograrlos, como ejemplo tenemos los siguientes:

- Consideran que el primer autómata en toda la historia fue Adán creado por Dios. De acuerdo a esto, Adán y Eva son los primero autómatas inteligentes creados, y Dios fue quien los programó y les dio sus primeras instrucciones que debieran seguir.
- El príncipe Kaya, hijo del Emperador Kannu, construye en el año 840 una muñeca que derrama agua.
- Reloj con forma de gallo que canta en la catedral de Strasbourg, que funcionó desde 1352 hasta 1789.
- Leonardo Da Vinci construye en el año 1500 un león automático en honor de Luis XII que actúa en la entrada del Rey de Milán.
- En 1640, René Descartes inventó un autómata al que se refiere como "mi hijo Francine".
- Robert Houdini construye una muñeca que escribe. También realiza un pastelero, un acróbata, una bailarina en la cuerda floja, un hombre que apunta con una escopeta y una artista del trapecio.
- Thomas Alva Edison construyó en el año 1891 una muñeca que habla.

A manera de experimento, si se pregunta, cómo reconoce si algo está vivo o tiene vida, podría apostarle que daría una lista de características para reconocerlo como tal pero no estaría en posición de dar una definición de vida que resultara aceptable por todas las personas que conoce. Aún más, haciendo un recuento de características, podría encontrar lo que siempre se conoce como la "excepción que confirma la regla".

Lista de acciones propuesta de forma pragmática por J. Doyne Farmer y Alletta D'a Belin, a través de las cuales podemos reconocer si un sistema está vivo.

- Existir en tiempo y en el espacio.
- Reproducirse por si mismo o dentro de un organismo relacionado.
- Almacenar información sobre sí mismo.
- Metabolizar (cambiar materia en energía).
- Actuar sobre su ambiente.
- Estar compuesto de partes interactivas independientes.
- Mantener la estabilidad durante condiciones ambientales cambiantes.

- Evolucionar.
- Crear o expandirse

Otro motivo del rechazo a la inteligencia y vida artificial se debe a que no hay una definición común aceptada por los diversos enfoques y disciplinas que de alguna manera tienen como objetivo la comprensión de la inteligencia o de la vida.

Conceptos de Robótica

Conceptos de Robótica.

Robótica es: El conjunto de conocimientos teóricos y prácticos que permiten concebir, realizar y automatizar sistemas basados en estructuras mecánicas poliarticuladas, dotados de un determinado grado de "inteligencia" y destinados a la producción industrial o como herramientas del hombre en determinadas tareas.

De forma general, la robótica se define como: El conjunto de conocimientos teóricos y prácticos que permiten concebir, realizar y automatizar sistemas basados en estructuras mecánicas poliarticuladas, dotados de un determinado grado de "inteligencia" y destinados a la producción industrial o a la sustitución del hombre en muy diversas tareas. Un sistema robótico puede describirse, como "Aquel que es capaz de recibir información, de comprender su entorno a través del empleo de modelos, de formular y de ejecutar planes, y de controlar o supervisar su operación". La robótica es esencialmente pluridisciplinaria y se apoya en gran medida en los progresos de la microelectrónica y de la informática, así como en los de nuevas disciplinas tales como el reconocimiento de patrones y de inteligencia artificial.

La definición de la Asociación de Industrias Robóticas (RIA) de en robot industrial es la siguiente:

> *Un robot industrial es un manipulador programable multifuncional, diseñado para mover piezas, herramientas, dispositivos especiales, programados para la ejecución de diversas tareas*

La definición de la Organización Internacional de Normas (ISO) es:

> *Un robot industrial es un manipulador automático reprogramable y multifuncional, que posee ejes capaces de agarrar materiales, objetos, herramientas mecanismos*

especializados a través de operaciones programadas para la ejecución de una variedad de tareas. Como se puede apreciar, estas definiciones se ajustan a la mayoría de las aplicaciones industriales de robots salvo para las aplicaciones de inspección y para los robots móviles (autónomos) o robots personales. Para Firebaugh un robot es una computadora con el propósito y la capacidad de movimiento.

Breve historia del desarrollo de los robots.

Breve historia del desarrollo de los robots.

La robótica abre una nueva y decisiva etapa en el actual proceso de mecanización y automatización creciente de los procesos de producción. Consiste esencialmente en la sustitución de máquinas o sistemas automáticos que realizan operaciones concretas, por dispositivos mecánicos que realizan operaciones concretas, de uso general, dotados de varios grados de libertad en sus movimientos y capaces de adaptarse a la automatización de un número muy variado de procesos y operaciones.

La robótica se ha caracterizado por el desarrollo de sistemas cada vez más flexibles, versátiles y polivalentes, mediante la utilización de nuevas estructuras mecánicas y de nuevos métodos de control y percepción.

La palabra robot surge con la obra RUR, los "*Robots Universales de Rossum*" de Karel Capek, es una palabra checoslovaca que significa trabajador, sirviente. Sin embargo podemos encontrar en casi todos los mitos de las diversas culturas una referencia a la posibilidad de crear un ente con inteligencia, desde el Popol-Vuh de nuestros antepasados mayas hasta el Golem del judaísmo. Desde la época de los griegos se intentó crear dispositivos que tuvieran un movimiento sin fin, que no fuera controlado ni supervisado por personas.

En los siglos XVII y XVIII la construcción de autómatas humanoides fabricados con mecanismos de relojería por Jacques de Vaucanson, Pierre Henri-Louis, Jaquet-Droz, como el escribiente, the Draughtsman, el músico Henri Maillar det (1800), Olimpia de la ópera de Offenback de Hoffman, fortalecieron la búsqueda de mecanismos que auxiliaran a los hombres en sus tareas. Estos autómatas desataron controversias alrededor de la posible inteligencia que pudieran tener estos dispositivos.

Los fraudes surgieron como en el caso del ajedrecista, en el que un muñeco mecánico daba respuesta a jugadas de ajedrez, comprobándose más tarde que era un enano encerrado en la caja del muñeco el que daba las respuestas y movía el muñeco.

Todos estos mitos anteceden a la obra de Karel Capek, en la que se plantea la construcción de robots para liberar a las personas de la carga pesada de trabajo. Sin embargo, esta ficción y la creada por Asimov, junto con las desarrollos mecánicos de máquinas como el telar de Thaillard, motiva a George Devol a crear el origen de los robots industriales, un manipulador que sería parte de una célula de trabajo, tomando como célula al conjunto de dispositivos, herramientas, manipuladores y operadores los cuales interactuan entre si para poder realizar una determinada tarea o tareas.

En las historias de robots escrita por Isaac Asimov, éste prevé un mundo futuro en que existían reglas de seguridad para que los robots no puedan ser dañinos para los seres humanos, por tal razón Isaac Asimov propuso las siguientes tres leyes para la robótica:

1. Un robot no puede dañar a un ser humano o, a través de la inacción, permitir que se dañe a un ser humano.

2. Un robot debe obedecer las órdenes dadas por los seres humanos, excepto cuando tales órdenes estén en contra de la primera ley.

3. Un robot debe proteger su propia existencia siempre y cuando esta protección no entre en conflicto con la primera y segunda ley.

Clasificación

Clasificación

La clasificación de los robots se establece de diversas maneras, temporalmente, por su funcionalidad, por su geometría, por la inteligencia, de ahi que hablar de generaciones de robots se puede plantear desde esos diversos puntos de vista.

Las características con las que se clasifican principalmente los robots son:

Propósito o función, Sistema de coordenadas empleado, Número de grados de libertad del efecto formal, Generación del sistema control.

1) Clasificación basada en su propósito o función:

a) Industriales

b) Persónale/ Educativos

La robótica es una gran ayuda en el área de investigación; con ayuda de robots especiales, los científicos pueden experimentar con robots de prueba antes de implantar algún nuevo programa de control.

La medicina también esta siendo apoyada por la robótica. Aunque todavía se esta investigando, se tienen resultados muy satisfactorios, de los cuales a largo plazo se podrán disfrutar.

La aplicación más antigua es en el hogar. Los electrodomésticos, como hoy los conocemos, forman parte del mundo de la robótica, y aunque parezca increíble, éstos son robots domésticos o personales. No se requiere de una gran programación previa, ni de mecanismos muy complejos para poder caracterizar a un robot doméstico, puesto que este es su fin: facilitar las labores domésticos, y por consiguiente ocupar el menor espacio posible para poder realizar las tareas.

En esta clasificación, también entran lo que se llaman mascotas virtuales un ejemplo: es el Furby.

c) Militares--vehículos autónomos.

Una de las aplicaciones muchos más aprovechadas de la robótica, y que el hombre se ha seguido maravillando, es la telerobótica en el espacio extraterrestre. La organización más importante dentro de este aspecto, y que ha marcado un rumbo muy avanzado en cuanto a tecnologías e investigaciones, es la NASA (National Aeronautics and Space Administration).

2) Clasificación de los robots basados en las generaciones de sistemas de control.

La primera generación: El sistema de control usado en la primera generación de robots esta basado en las "paradas fijas" mecánicamente. Esta estrategia es conocida como control de lazo abierto o control "bang bang". Podemos considerar como ejemplo esta primera etapa aquellos mecanismos de relojería que permiten mover a las cajas musicales o a los juguetes de cuerda. Este tipo de control es muy similar al ciclo de control que tienen algunos lavadores de ciclo fijo Son útiles para las aplicaciones industriales de tomar y colocar pero están limitados a un número pequeño de movimientos.

La segunda generación utiliza una estructura de control de ciclo abierto, pero en lugar de utilizar interruptores y botones mecánicos utiliza una secuencia numérica de control de movimientos almacenados en un disco o cinta magnética. El programa de control entra mediante la elección de secuencias de movimiento en una caja de botones o a través de palancas de control con los que se "camina", la secuencia deseada de movimientos. El mayor número de aplicaciones en los que se utilizan los robots de esta generación es de la industria automotriz, en tareas como soldadura, pintado con "spray". Este tipo de robots constituye la clase más grande de robots industriales en E.U., incluso algunos autores sugieren que cerca del 90 % de los robots industriales en EU pertenece a esta 2ª generación de control

La tercera generación de robots utiliza las computadoras para su estrategia de control y tiene algún conocimiento del ambiente local a través del uso de censores, los cuales miden el ambiente y modifican su estrategia de control, con esta generación se inicia la era de los robots inteligentes y aparecen los lenguajes de programación para escribir los programas de control. La estrategia de control utilizada se denomina de "ciclo cerrado"

La cuarta generación de robots, ya los califica de inteligentes con más y mejores extensiones sensoriales, para comprender sus acciones y el mundo que los rodea. Incorpora un concepto de "modelo del mundo" de su propia conducta y del ambiente en el que operan. Utilizan conocimiento difuso y procesamiento dirigido por expectativas que mejoran el desempeño del sistema de manera que la tarea de los censores se extiende a la supervisión del ambiente global, registrando los efectos de sus acciones en un modelo del mundo y auxiliar en la determinación de tareas y metas.

La quinta generación, actualmente está en desarrollo esta nueva generación de robots, que pretende que el control emerja de la adecuada organización y distribución de módulos conductuales, esta nueva arquitectura es denominada arquitectura de subsumción, cuyo promotor es Rodney Brooks

Contexto actual de la ROBOTICA.

En el contexto actual la noción de robótica implica una cierta idea preconcebida de una estructura mecánica universal capaz de adaptarse, como el hombre, a muy diversos tipos de acciones y en las que concurren, en mayor o menor grado según los casos, las características de movilidad, programación, autonomía y multifuncionalidad. Pero en sentido actual, abarca una amplia gama de dispositivos con muy diversos trazos físicos y funcionales asociados a la particular estructura mecánica de aquellos, a sus características operativas y al campo de aplicación para el que se han concebido. Es además evidente que

todos estos factores están íntimamente relacionados, de tal forma que la configuración y el comportamiento de un robot condicionan su adecuación para un campo determinado de aplicaciones y viceversa, y ello a pesar de la versatilidad inherente al propio concepto de robot.

La construcción de un ROBOT.

La construcción de un ROBOT.

La construcción de un robot, ya sea una máquina que camine de forma parecida a como lo hace el ser humano, o un manipulador sin rostro para una línea de producción, es fundamentalmente un problema de control. Existen dos aspectos principales: mantener un movimiento preciso en condiciones que varían y conseguir que el robot ejecute una secuencia de operaciones previamente determinadas. Los avances en estos dos campos son, esencialmente un problema matemático, y el segundo de tecnología, los cuales suministran la más grande contribución al desarrollo del robot moderno.

Los manipuladores propiamente dichos representan, en efecto, el primer paso en la evolución de la robótica y se emplean preferentemente para la carga - descarga de máquinas - herramientas, así como para manutención de prensas, cintas transportadores y otros dispositivos.

Actualmente los manipuladores son brazos articulados con un número de grados de libertad que oscila entre dos y cinco; cuyos movimientos, de tipo secuencial, se programan mecánicamente o a través de una computadora. Los manipuladores no permiten la combinación simultánea de movimientos ni el posicionamiento continuo de sus terminales. A pesar de su concepción básicamente sencilla, se han desarrollado manipuladores complejos para adaptarlos a aplicaciones concretas en las que se dan condiciones de trabaja especialmente duras o especificaciones de seguridad muy exigente

Arquitectura de los ROBOTS.

La arquitectura, definida por el tipo de configuración general del robot, puede sér metamórfico. El concepto de metamorfismo, de reciente aparición, se ha introducido para incrementar la flexibilidad funcional de un robot a través del cambio de su configuración por el propio robot. El metamorfismo admite diversos niveles, desde los más elementales -

cambio de herramienta o de efector terminal-, hasta los más complejos como el cambio o alteración de algunos de sus elementos o subsistemas estructurales.

Los dispositivos y mecanismos que pueden agruparse bajo la denominación genérica del robot, tal como se ha indicado, son muy diversos y es por tanto difícil establecer una clasificación coherente de los mismos que resista un análisis crítico y riguroso. La subdivisión de los robots, con base en su arquitectura, se hace en los siguientes grupos: Poliarticulados, Móviles, Androides, Zoomórficos e Híbridos.

Poliarticulados.-

Bajo este grupo están los robots de muy diversa forma y configuración cuya característica común es la de ser básicamente sedentarios aún que excepcionalmente pueden ser guiados para efectuar desplazamientos limitados- y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas y con un número limitado de grados de libertad". En este grupo se encuentran los manipuladores, los robots industriales, los robots cartesianos y se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o deducir el espacio ocupado en el suelo.

Móviles.-

Son robots con gran capacidad de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus censores.

Las tortugas motorizadas diseñadas en los años cincuenta, fueron las precursoras y sirvieron de base a los estudios sobre inteligencia artificial desarrollados entre 1965 y 1973 en la Universidad de Stanford.

Estos robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

Androides.-

Son robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Actualmente los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, fundamentalmente, al estudio y experimentación.

Uno de los aspectos más complejos de estos robots, y sobre el que se centra la mayoría de los trabajos, es el de la locomoción bípeda. En este caso, el principal problema es controlar dinámica y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del robot.

Zoomórficos.-

Los robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos.

A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los robots zoomórficos no caminadores está muy poco evolucionado. Cabe destacar, entre otros, los experimentados efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. En cambio, los robots zoomórficos

caminadores multípedos son muy numerosos y están siendo experimentados en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, manipulados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

Híbridos.-

Estos robots corresponden a aquellos de difícil clasificación cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo uno de los atributos de los robots móviles y de los robots zoomórficos. De igual forma puede considerarse híbridos algunos robots formados por la yuxtaposición de un cuerpo formado por un carro móvil y de un brazo semejante al de los robots industriales. En parecida situación se encuentran algunos robots antropomorfos y que no pueden clasificarse ni como móviles ni como androides, tal es el caso de los robots personales.

La Automatización

La automatización y la robótica son dos tecnologías estrechamente relacionadas. En un contexto industrial podemos definir la automatización como una tecnología que está relacionada con el empleo de sistemas mecánicos, electrónicos y basados en computadoras en la operación y control de la producción. Ejemplos de esta tecnología son: líneas de transferencias, máquinas de montaje mecanizado, sistemas de control de retroalimentación, máquinas - herramientas con control numérico y robots. En consecuencia, la robótica es una forma de automatización industrial.

Hay tres clases amplias de automatización industriales:

Automatización fija.

Automatización programable.

Automatización flexible.

La automatización fija se utiliza cuando el volumen de producción es muy alto, y por tanto es adecuada para diseñar equipos especializados para procesar el producto (o un componente de producto) con alto rendimiento y con elevadas tasas de producción, esto claro es en la producción de motores y transmisiones.

La automatización programable se emplea cuando el volumen de producción es relativamente bajo y hay una diversidad de producción a obtener. En este caso el equipo de producción está diseñado para ser adaptable a variaciones en la configuración del producto. Esta característica de adaptabilidad se realiza haciendo funcionar el equipo bajo el control de un programa de instrucciones para el producto dado.

La automatización flexible es una categoría entre automatización fija y automatización programable. Este tipo de automatización se ha visto que es más adecuado para el rango de producción medio. Una de las características que distingue a la automatización programable de la flexible, es que con la primera los productos se obtienen en lote. Cuando se completa un lote, el equipo sé reprograma para procesar el siguiente lote. Con la automatización flexible, diferentes tipos pueden obtenerse al mismo tiempo e n el mismo sistema de fabricación.

De los tres tipos de automatización, la robótica coincide más estrechamente con la automatización programable.

Vida Artificial

Vida Artificial

La Vida Artificial se puede considerar como la parte de la Inteligencia Artificial que pretende reproducir los procesos y comportamientos típicos de los seres vivos con el objetivo de resolver problemas. ¿Para que una entidad sea considerada inteligente basta con que se comporte inteligentemente, o además debe razonar de forma inteligente? Puede que la diferencia entre las dos posibilidades parezca sutil a primera vista, pero tiene más importancia de la que parece. También podemos definirla como el intento de crear vida, o algo parecido a la vida, mediante la combinación de símbolos (datos) y procesos de símbolos (programas) independientemente del soporte físico de estos símbolos y procesos.

Vida Artificial (VA), su surgimiento, sus objetivos, su relación unificante de disciplinas divergentes como la robótica, la psicología, la lingüística, las neurociencias, la ciencia cognitiva, la teología, entre otras. Se explora el impacto mutuo de la robótica y la biología, y dentro de esta exploración la del enfoque denominado "Basado en la Conducta" para la construcción de un robot que conduzca al posible modelado del comportamiento de forrajeo de monos con grupos de estos animats (*animat = animal artificial*).

Por una parte están los intentos "hardware" de emulación de vida. Por ejemplo, es posible construir un pequeño robot con aspecto de ratón capaz de encontrar la salida de un laberinto. Por otra parte están las simulaciones "software". Éstas tienen la ventaja de que permiten construir un gran número de seres vivos y entornos en los que estos existen, de manera que es más fácil estudiar comportamientos sociales.

Se explora el impacto mutuo de la robótica y la biología, en particular el que se consigue con el enfoque de "sistemas basados en la conducta". Estos sistemas permiten la construcción de un robot que modele las conductas de forrajeo del mono1, y la utilización de un grupo de estos "animats" en la modelación del forrajeo grupal de los monos. El interés reside en la observación de la emergencia de conductas grupales en

los animats correspondientes a las de los grupos de monos y la realimentación conceptual que tales observaciones pueden tener para robotistas y etólogos.

Podemos construir los seres artificiales con el objetivo de solucionar los problemas que a nosotros nos interesen, y que aprendan o colaboren entre ellos hasta conseguir el resultado deseado.

Tarjeta controladora

Tarjeta controladora.

El diseño de la tarjeta esta basada en el funcionamiento del microcontrolador PIC 16F84 .

El funcionamiento es el siguiente:

El PIC consta de dos puertos, el puerto A en esta ocasión lo tomamos como entrada al ser bidireccional teniendo cinco posibles entradas de datos. El puerto B lo tomamos como salida, este puerto nos dará la dirección y el dato ya que tomamos a las patas RB7(13) y RB6(12) como direcciones y las patas RB0 a RB3 como datos. Con esta combinación podemos tener dos direcciones de cuatro bits cada uno.

Con esto decimos que nuestra tarjeta costa de dos puertos uno de entrada de cinco bits y uno de dos direcciones de cuatro bits de salida.

La salida esta controlada y protegida mediante un integrado el 74 LS 374, se pone este circuito ya que el PIC manda sus datos en un pequeño lapso de tiempo y 74LS374 retiene ese valor hasta que le llega uno nuevo.

Los puertos tanto de entrada como de salida pueden operar independientemente uno del otro.

La tarjeta esta diseñada para que se pueda expandir ya sea en la entrada o en la salida , utilizando las dos patas restantes RB5(11) y RB4(10) , teniendo como entrada a siete bits da datos y como salida cuatro direcciones de cuatro bits cada uno.

AL sistema se le puede conectar diferentes dispositivos , que controlen a la tarjeta (teclado) o dispositivos que la tarjeta controle (circuitos de potencia).

¿Porque el PIC 16F84 ? La rasos de utilizar este microcontrolador es la siguiente:

Primero su costo, se puede conseguir con gran facilidad a un precio de alrededor de $ 65.00 M.N.

Consta de una memoria EEPROM de 64 bytes .

Una memoria de programación Flash. De 1k.

Una pila(stack) de 8 niveles.

Cuatro diferentes interrupciones.

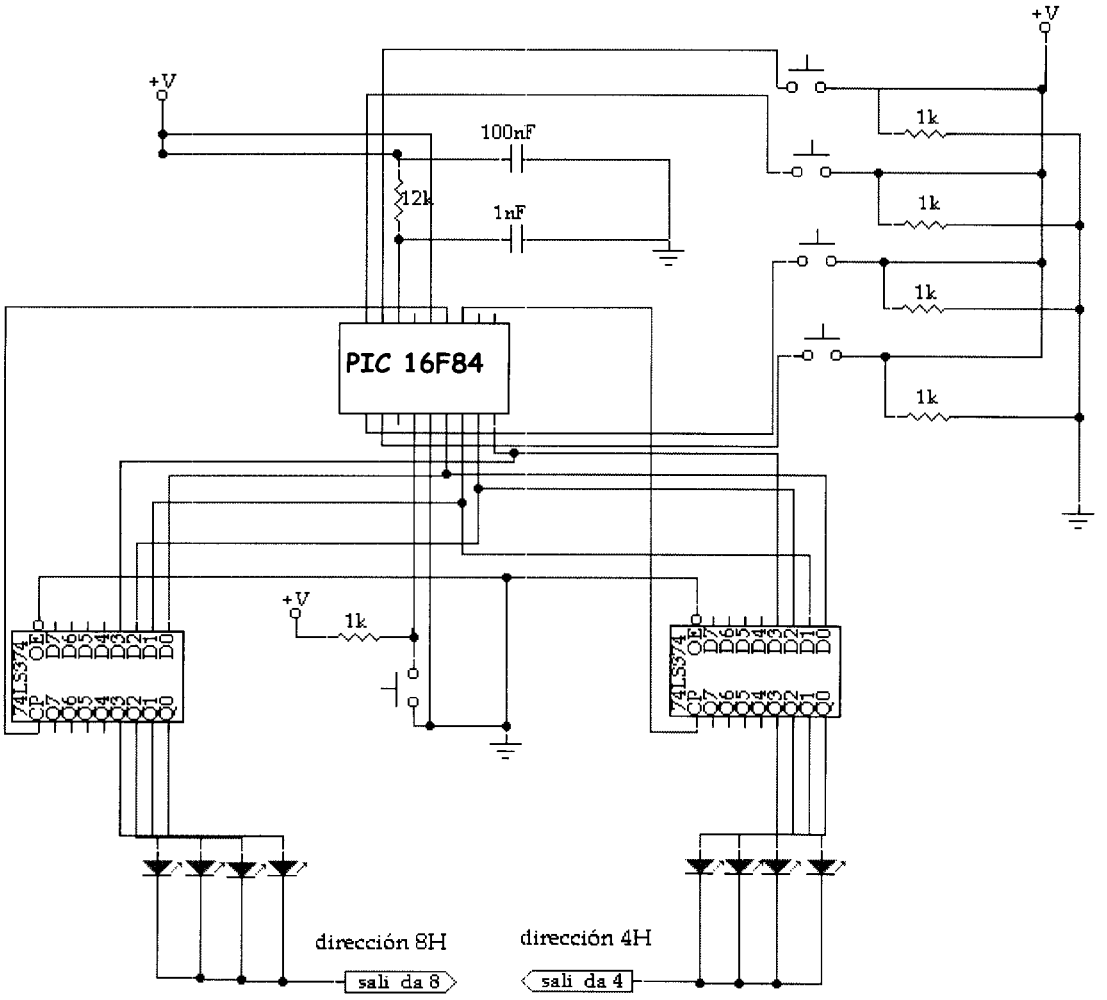
Encapsulado plástico DIP de 18 patas.

Con una frecuencia máxima de trabajo de 10 Mhz.

El software es fácil de conseguir, se encuentra gratis en la pagina de Microchip.

El grabador se puede armar con un bajo costo y se consigue en Internet.

Por esto y las demás características que se explicaran posteriormente, se toma la decisión de trabajar en base al PIC 16F84.



Tarjeta Controladora

Microcontrolador PIC

Microcontrolador PIC

Microchip dispone de cuatro familias de microcontroladores de 8 bits , según sea la necesidad.

Se compone de :

Gama Enana, PIC 12C(F)XX de 8 patas, su alimentación esta entre 2.5V y 5.5V y consume menos de 2mA cuando trabaja a 5V y 4 MHz. El formato de sus instrucciones puede ser de 12 o de 14 bits y con un repertorio de 33 o 35 instrucciones.

Gama Baja o básica, PIC 16c5x, estos pueden tener 18 o 28 paras y se alimenta a partir de una tensión de 2.5V.Tiene un numero de 33 instrucciones con formato de 12 bits.

Gama Media , Abarca modelos con encapsulado de 18 patas hasta 68, el repertorio de instrucciones es de 35 de 14 bits cada una ,siendo compatible con la gama baja.

Dentro de esta gama se encuentra el PIC 16X8X donde se encuentra el PIC 16F84.

Gama Alta , PIC 17CXXX consta de 58 instrucciones de 16 bits , sus modelos disponen de in sistema de gestión de interrupciones vectorizadas. También incluyen variados controladores de periféricos, puertas de comunicación serie y paralelo con elementos externos y un multiplicador hardware.

El PIC 16F84

El " PIC 16F84 " es un microcontrolador con memoria de programa tipo FLASH, lo que representa gran facilidad en el desarrollo de prototipos y en su aprendizaje ya que no se requiere de borrado con luz ultravioleta como las versiones EPROM sino, permite reprogramarlo nuevamente si ser borrado con anterioridad. Por esta razón, lo usaremos en la mayoría de aplicaciones que se desarrollan a lo largo del estudio.

Es compatible con el PIC 16C84. Su principal característica es que posee memoria "EEPROM" en lugar de memoria Flash, pero su manejo es igual. Con respecto al PIC16F84, este presenta dos diferencias:

- La memoria de datos tiene menor tamaño, aquí se tienen 32 registros de propósito general (el mapa de memoria de datos llega hasta 2Fh).
- En el momento de programar ,el fusible de selección del temporizador de arranque (Power Up Timer) trabaja de forma inversa, es decir, si en el PIC 16F84 se selecciona la opción "Low" para activarlo, en el PIC 16C84 se debe seleccionar "High".

E l PIC 16C84 ha sido reemplazado de forma gradual por el PIC 16F84, por lo tanto, los diseños que lo utilicen como elemento de control deben ser actualizados. Aunque, como se ve, es un proceso casi transparente.

Se basa en la Arquitectura Harvard, en la cual el programa y los datos se pueden trabajar desde memorias separadas, lo que posibilita que las instrucciones y los datos posean longitudes diferentes. Esta misma estructura es la que permite la superposición de los ciclos de búsqueda y ejecución de las instrucciones, lo cual se ve reflejado en una mayor velocidad del microcontrolador.

Memoria de programa:

Es una memoria de 1 K byte de longitud con palabra de 14 bits. Como es del tipo FLASH se puede programar y borrar eléctricamente, en otras palabras, se puede programar o borrar sin necesidad de un borrador de luz ultravioleta, lo que facilita el desarrollo de programas y la experimentación. Como el PIC 16F84 tiene un contador de programa de 13 bits, tiene una capacidad de direccionamiento de 8K x 14, pero solamente tiene implementado el primer 1K x 14 (000h hasta 03FFh). Si se direccionan posiciones de memoria superiores a 3FFh se causará un desborde con el espacio del primer 1K.

Vector de REST:

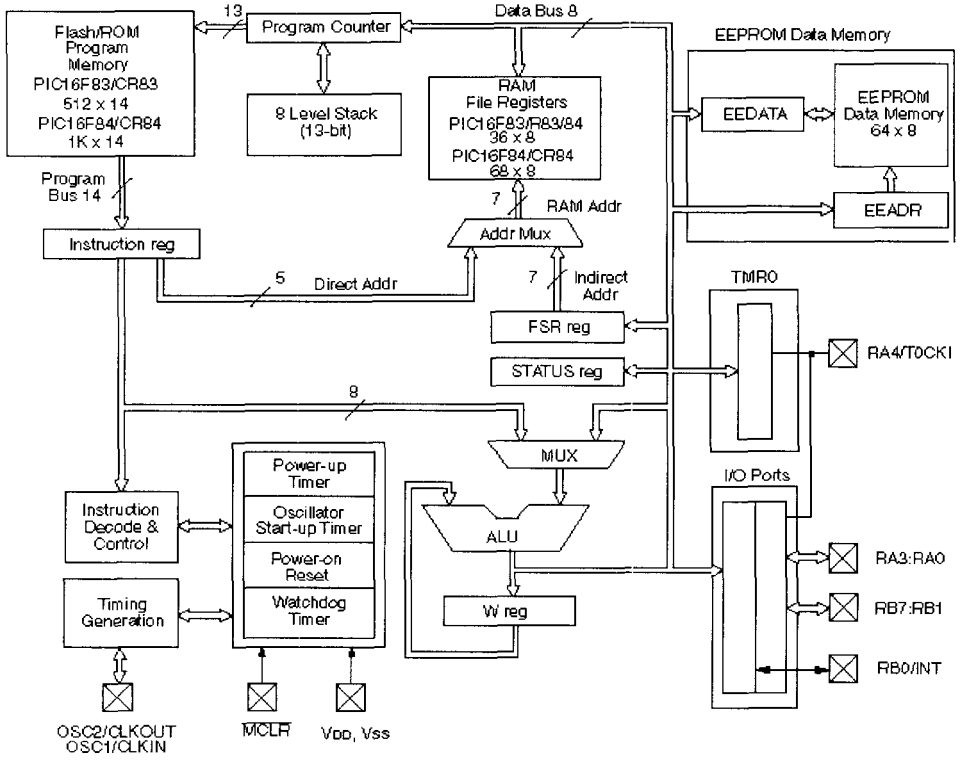
Cuando ocurre un reset o se enciende el microcontrolador, el contador de programa se pone en ceros (000h). Por esta razón, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

Vector de interrupción:

Cuando el microcontrolador recibe una señal de interrupción el contador de programa apunta a la dirección 04h de la memoria de programa, por eso allí se debe escribir toda la programación necesaria para atender dicha interrupción.

Registro (Memoria RAM):

El PIC 16F84 puede direccionar 128 posiciones de memoria RAM, pero solamente tiene implementado físicamente los primeros 80 (0 a 4Fh). De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 68 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se esta ejecutando. Los registros están organizados como dos bancos (paginas) de 128 posiciones de 8 bits cada una (128 x 8); todas las posiciones se pueden accesar directa o indirectamente (estas ultimas a través del registro FSR). Para seleccionar que pagina de registro se trabaja en un momento determinado se utiliza el bit RP0 del registro STATUS.



Pines y funciones:

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 entrada o como salida.

El PIC 16F84 tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8 líneas. Cada pin se puede configurar como entrada o como salida independiente programado por un par de registros diseñados para tal fin. En ese registro un "0" configura el pin del puerto correspondiente como salida y un "1" lo configura como entrada.

Puerto A

RA0 = Pin de Entrada/Salida (TTL).

RA1 = Pin de Entrada/Salida (TTL).

RA2 = Pin de Entrada/Salida (TTL).

RA3 = Pin de Entrada/Salida (TTL).

RA4/TOCKI = Pin de Entrada/Salida o entrada de Reloj Externo para el TMR0, cuando este pin se configura como salida es de tipo Open Drain (ST), cuando funciona como salida se debe conectar a Vcc (+5V) a través de una resistencia.

Puerto B:

RB0/INT = Pin de Entrada/Salida o entrada de interrupción externa. (TTL/ST).

RB1 = Pin de Entrada/Salida (TTL).

RB2 = Pin de Entrada/Salida (TTL).

RB3 = Pin de Entrada/Salida (TTL).

RB4 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL).

RB5 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL).

RB6 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL/ST).

RB7 = Pin de Entrada/Salida con Interrupción por cambio de Flanco (TTL/ST).

Pines adicionales:

MCLR = Pin de Reset del Microcontrolador (Master Clear). Se activa (el pic se resetea) cuando tiene un "0" lógico en su entrada.

Vss = Ground o Tierra

V_{DD} = Fuente Positiva (+5V)

OSC2/CLKOUT= Entrada del Oscilador del Cristal. Se conecta al Cristal o Resonador en modo XT (Oscilador de Cristal). En modo RC (Resistencia - Condensador), este pin actúa como salida el cual tiene 1/4 de la frecuencia que entra por el pin OCS1/CLKIN.

OCS1/CLKIN = Entrada del Oscilador del Cristal / Entrada de reloj de una Fuente Externa.

El Puerto B tiene Internamente unas resistencias de pull-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 pull-up conectan o desconectan a la vez. La resistencia de pull-up es desconectada automáticamente en un pin si este se programa como salida. El pin RB0/INT se puede configurar por software para que funcione como interrupción externa.

El pin RA4/TOCKI del puerto A puede ser configurado como un pin de entrada/salida como se mencionaba anteriormente o como entrada del temporizador/contador. Cuando este pin se programa como entrada digital, funciona como un disparador de Schmitt (Schmitt trigger, ST), esto quiere decir que puede reconocer señales un poco distorsionadas y llevarlas a niveles lógicos (cero y cinco voltios). Cuando se usa como salida digital se comporta como colector abierto, por lo tanto se debe poner una resistencia de pull-up (resistencia externa conectada a un nivel lógico de cinco voltios). Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega en el pin un "1" lógico. Además como salida no puede manejar cargas como fuente, sólo en el modo sumidero.

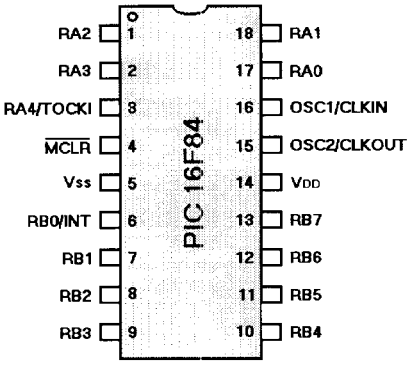
Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectado a alguna parte, nunca dejarlos al aire por que se puede dañar el integrado. Los

pines que no se estén usando se deben conectar la fuente de alimentación +5V con una resistencia de aproximadamente 5 Kilo Ohmio.

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.

El consumo de corriente del microcontrolador para su funcionamiento depende del voltaje de operación, la frecuencia y de las cargas que tengan sus pines.

Por Ejemplo: Para un reloj de 4 MHz el consumo es de aproximadamente de 2mA; aunque este se puede reducir a 40 microamperes cuando está en el modo sleep (en este modo el micro se detiene y disminuye el consumo de potencia). Se sale de este estado cuando se produce alguna condición especial que veremos mas adelante.



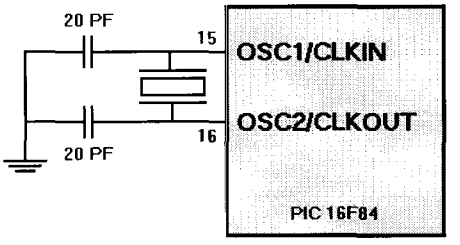
Oscilador Externo:

Todo Microcontrolador requiere un circuito externo que le indique la velocidad a la que debe trabajar. Este circuito, que se conoce con el nombre de oscilador o reloj, es muy simple pero de vital importancia para el buen funcionamiento del sistema. El PIC 16F84 puede utilizar cuatro tipos de oscilador diferentes. Estos tipos son:

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

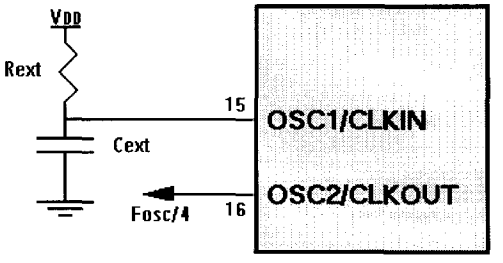
En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados "fusibles de configuración".

En la mayoría de las practicas que realizaremos se sugiere el cristal de 4MHz. por que garantiza una mayor precisión y un buen arranque del microcontrolador. Internamente esta frecuencia esta dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se realiza en un microsegundo (1 μS). El cristal debe ir acompañado de dos condensadores y se conecta como se muestra en la figura siguiente.



Dependiendo de la aplicación, se pueden utilizar cristales de otras frecuencias; por ejemplo se usa el cristal de 3.579545 MHz por que es muy económico, el de 32.768 KHz cuando se necesita crear bases de tiempo de un segundo muy precisas. El límite de velocidad de estos microcontroladores es de 10 MHz.

Si no se requiere mucha presición en el oscilador y se requiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra a continuación:



Los valores recomendados para este tipo de oscilador son: 5 KHz. Rext de 100 KΩ y Cext de 20 pF.

Nota: Cuando el oscilador del dispositivo esta en modo RC, no maneje el pin OSC1 con un reloj externo por que puede dañar el dispositivo.

La frecuencia del oscilador dividida por cuatro está disponible en el pin OSC2/CLKOUT, y puede ser usada para chequear propósitos o para sincronizar otra lógica.

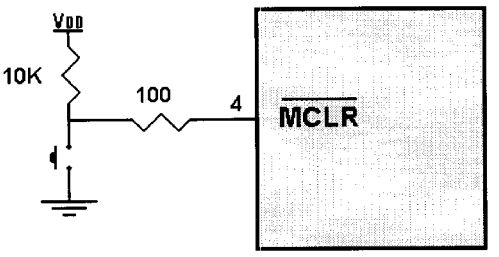
EL RESET:

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando sea necesario, ya sea por una falla que se presente o por que así se halla diseñado el sistema. El pin de reset en los PIC es llamado "Master Clear". El PIC 16F84 admite diferentes tipos de reset:

- Al encendido (Power On Reset)
- Pulso en el pin Master Clear durante operación normal
- Pulso en el pin Master Clear durante el modo de bajo consumo (modo sleep)
- El rebase del conteo del circuito de vigilancia (watchdog) durante operación normal.
- El rebase del conteo del circuito de vigilancia (watchdog) durante el modo de bajo consumo (sleep)

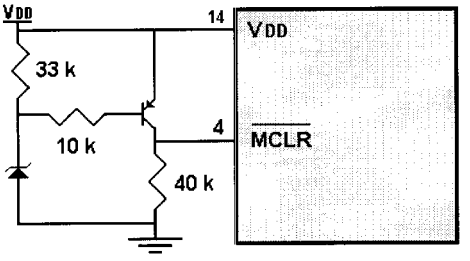
El reset al encendido se consigue gracias a dos temporizadores. El primero de ellos es el OST (Oscillator Star-Up Timer: Temporizador de encendido del oscilador), orientado a mantener el microcontrolador en reset hasta que el oscilador de cristal es estable. El segundo es el PWRT (Power-Up Timer: Temporizador de encendido), que provee un retardo fijo de 72 mS (nominal) en el encendido únicamente, diseñado para mantener el dispositivo en reset mientras la fuente se estabiliza. Para utilizar estos temporizadores, solo basta conectar el pin Master Clear a la fuente de alimentación evitándose utilizar las tradicionales redes RC externas en el pin de reset.

El reset por Master Clear se consigue llevando momentáneamente este pin a un estado lógico bajo, mientras que el watchdog WDT produce un reset cuando su temporizador rebasa la cuenta. Cuando se quiere tener control sobre el reset del sistema se puede conectar un botón como se muestra en la siguiente figura.



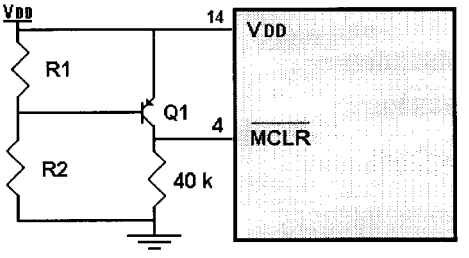
Reset por Brown-Out: Un brown-out es una condición en donde la alimentación del dispositivo (Vdd) baja a un valor mínimo, pero no a cero y luego se normaliza. El dispositivo debe resetearse en caso de presentarse un brown-out. Para resetear un PIC 16F84 cuando un brown-out ocurre se debe construir un circuito de protección externo como el de la siguiente figura:

Circuito de Protección # 1.



Este circuito entrará en un reset activo cuando VDD baja por debajo de Vz + 0.7, en donde Vz = Voltaje del Zener.

Circuito de Protección # 2.



Este circuito es más económico, aunque menos eficaz. El transistor Q1 pasará a un estado de corte cuando VDD está por debajo de un cierto nivel tal que:

VDD * (R1 / (R1 + R2)) = 0.7 V

Registros (Memoria RAM):

El PIC 16F84 puede direccionar 128 posiciones de memoria RAM, pero solamente tiene implementado físicamente los primeros 80 (0 a 4Fh). De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 68 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se esta ejecutando. Los registros están organizados como dos bancos (paginas) de 128 posiciones de 8 bits cada una (128 x 8); todas las posiciones se pueden accesar directa o indirectamente (estas ultimas a través del registro FSR). Para seleccionar que pagina de registro se trabaja en un momento determinado se utiliza el bit RP0 del registro STATUS.

00h o INDO :Registro para el direccionamiento indirecto de datos. Este no es un registro disponible físicamente; utiliza el contenido del FSR y el bit RP0 del registro STATUS para seleccionar indirectamente la memoria de datos o RAM del usuario; la instrucción determinara que se debe señalar con el registro señalado.

01h o TMR0: Temporizador/contador de 8 bits. Este se puede incrementar con una señal externa aplicada al pin RA4/TOCKI o de a cuerdo a una señal interna proveniente del reloj de instrucciones del microcontrolador. La rata o tasa de incremento del registro se puede determinar por medio de un preescalador, localizado en el registro OPTION. Los anteriores microcontroladores no contaban con la generación de una interrupción cuando se rebasaba la cuenta (el paso de 0FFh a 00h).

02h o PCL: CONTADOR DE PROGRAMA. Se utiliza para direccionar las palabras de 14 bits del programa del usuario que se encuentra almacenado en la memoria ROM; este contador tiene un tamaño de 13 bits. Sobre el byte bajo, se puede escribir o leer a voluntad directamente, mientras que en el byte alto, no. El byte alto se maneja mediante el registro

PCLATH (0Ah). A diferencia de los PIC de primera generación el 16F84 ante una condición de reset inicia el contador de programa con todos sus bits en "cero". Durante la ejecución normal del programa, y dado que todas las instrucciones ocupan solo una posición de memoria, el contador se incrementa con cada instrucción, a menos que se trate de alguna instrucción de salto.

03h o STATUS: REGISTRO DE ESTADO. Contiene el estado Aritmético de la ALU, la causa de reset y los bits de preselección de pagina para la memoria de datos. En tabla 1. se muestran los bits correspondientes a este registro. Los bits 5 y 6 (RP0 y RP1) son los bits de selección de pagina (Bank 0 y Bank 1), para el direccionamiento directo de la memoria de datos; solamente RP0 se usa en los PIC 16F84. RP1 se puede utilizar como un bit de propósito general de lectura/escritura. Los bits TO y PD no se pueden modificar por un proceso de escritura; ellos muestran la condición por la cual se ocasiono el ultimo reset.

tabla1

Registro STATUS

IRP bit 7	RP1 bit 6	RP0 bit 5	T0 bit 4	PD bit 3	Z bit 2	DC bit 1	C bit 0
IRP	Selector de página para direccionamiento indirecto. Este bit no se utiliza efectivamente en el PIC 16F84, por lo que se puede utilizar como un bit de propósito general.						
RP1,0	Selectores de página para un seleccionamiento directo. Solamente RP0 se utiliza en el PIC 16F84. RP1 se puede utilizar como un bit de propósito general.						
T0	Time Out o bit de finalización del temporizador. Se coloca en 0 cuando el circuito de vigilancia Watchdog finaliza la temporización						
PD	Power Down o bit de bajo consumo. Se coloca en 0 por la instrucción sleep.						
Z	Zero o bit de cero. Se coloca en 1 cuando el resultado de una operación aritmética o lógica es cero.						
DC	Digit Carry o bit de acarreo de dígito. En operaciones aritméticas se activa cuando hay un acarreo entre el bit 3 y 4, es decir cuando hay acarreo entre el nible de menor y de mayor peso.						
C	Carry o bit de acarreo. En instrucciones aritméticas se activa cuando se presenta acarreo desde el bit más significativo del resultado.						

04h o FSR : REGISTRO SELECTOR DE REGISTROS. En asocio con el registro INDO, se utiliza para seleccionar indirectamente los otros registros disponibles. Mientras que los antecesores del PIC 16F84 solo poseían 5 bits activos, en este microcontrolador se poseen

solo 8 bits. Si en el programa no se utilizan llamadas indirectas, este registro se puede utilizar como un registro de propósito general.

05h o PORTA : PUERTO DE ENTRADA/SALIDA DE 5 BITS (RA0~ RA4). Este puerto al igual que todos sus similares en los PIC, puede leerse o escribirse como si se tratara de un registro cualquiera. El registro que controla el sentido (entrada o salida) de los pines de este puerto esta localizado en la pagina 1 (Banco 1), en la posición 85h y se llama TRISA.

06h o POTRB: PUERTO DE ENTRADA/SALIDA DE 8 BITS (RB0~RB7). Al igual que en todos los PIC, este puede leerse o escribirse como si se tratara de un registro cualquiera; algunos de sus pines tienen funciones alternas en la generación de interrupciones. El registro de control para la configuración de la función de sus pines se localiza en la pagina 1 (Banco 1), en la dirección 86h y se llama TRISB.

08h o EEDATA: REGISTRO DE DATOS DE LA EEPROM. Este registro contiene el dato que se va a escribir en la memoria EEPROM de datos o el que se leyó de ésta.

09h o EEADR: REGISTRO DE DIRECCION DE LA EEPROM. Aquí se mantiene la dirección de la EEPROM de datos que se van a trabajar, bien sea para una operación de lectura o para una de escritura.

0Ah o PCLATH: REGISTRO PARA LA PARTE ALTA DE LA DIRECCION. Este registro contiene la parte alta del contador de programa y no se puede acceder directamente.

0Bh o INTCON :REGISTRO PARA EL CONTROL DE INTERRUPCIONES. Es el encargado del manejo de las interrupciones y contiene los bits que se muestran en tabla 2.

tabla2

Registro INTCON

GIE Bit 7	EEIE bit 6	TOIE bit 5	INTE bit 4	RBIE bit 3	TOIF bit 2	INTF bit 1	RBIF bit 0
GIE	Global Interrup Enable o Habilitador general de interrupciones. 0: Deshabilita todas las interrupciones 1: Habilita las interrupciones						
EEIE	EEPROM Write Interrup Enable o Habilitación de interrupción por escritura de la EEPROM. 0: La deshabilita 1: La habilita						
TOIE	TMR0 Interrup Enable o Habilitación de interrupción del temporizador TMR0. 0: La deshabilita 1: La habilita						
INTE	INT Interrup Enable o Habilitación de la interrupción INT. 0: La deshabilita 1: La habilita						
RBIE	RBIF Interrup Enable o Habilitación de la interrupción RBIF. 0: La deshabilita 1: La habilita						
TOIF	TMR0 Overflow Interrup Flag o Bandera de la interrupción por desbordamiento del TMR0. Se coloca en 1 cuando el TMR0 pasa de 0FFh a 00h; ésta debe ser puesta a 0 por programa.						
INTF	INT Interrup Flag o Bandera de interrupción INT. Se coloca en 1 cuando la interrupción INT ocurre; ésta debe ser puesta cero por programa.						
RBIF	RB Port Change Interrup Flag o Bandera de interrupción por cambio en el puerto B. Se coloca en 1 cuando una de las entradas (RB4 a RB7) cambia; ésta debe ser puesta a 0 por programa						

81h u OPTION: REGISTRO DE CONFIGURACION MULTIPLE. Posee varios bits para configurar el preescalador, la interrupción externa, el timer y las características del Puerto B. Los bits que contiene y las funciones que realiza este registro se muestran en tabla 3. El preescalador es compartido entre el TMR0 y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.

tabla 3.

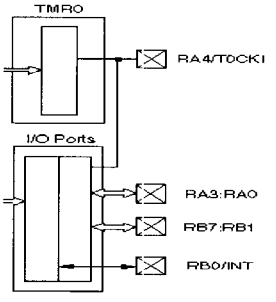
REGISTRO INTCON

GIE Bit 7	EEIE bit 6	TOIE bit 5	INTE bit 4	RBIE bit 3	TOIF bit 2	INTF bit 1	RBIF bit 0

GIE	Global Interrup Enable o Habilitador general de interrupciones. 0: Deshabilita todas las interrupciones 1: Habilita las interrupciones
EEIE	EEPROM Write Interrup Enable o Habilitación de interrupción por escritura de la EEPROM. 0: La deshabilita 1: La habilita
TOIE	TMR0 Interrup Enable o Habilitación de interrupción del temporizador TMR0. 0: La deshabilita 1: La habilita
INTE	INT Interrup Enable o Habilitación de la interrupción INT. 0: La deshabilita 1: La habilita
RBIE	RBIF Interrup Enable o Habilitación de la interrupción RBIF. 0: La deshabilita 1: La habilita
TOIF	TMR0 Overflow Interrup Flag o Bandera de la interrupción por desbordamiento del TMR0. Se coloca en 1 cuando el TMR0 pasa de 0FFh a 00h; ésta debe ser puesta a 0 por programa.
INTF	INT Interrup Flag o Bandera de interrupción INT. Se coloca en 1 cuando la interrupción INT ocurre; ésta debe ser puesta cero por programa.
RBIF	RB Port Change Interrup Flag o Bandera de interrupción por cambio en el puerto B. Se coloca en 1 cuando una de las entradas (RB4 a RB7) cambia; ésta debe ser puesta a 0 por programa

85h o TRISA: REGISTRO DE CONFIGURACION DEL PUERTO A. Es el registro de control para el puerto A. Un "cero" en el bit correspondiente al pin lo configura como salida, mientras que un "uno" lo hace como entrada.

86h o TRISB: REGISTRO DE CONFIGURACION DEL PUERTO B. Es el registro de control para el puerto B. Un "cero" en el bit correspondiente al pin lo configura como salida, mientras que un "uno" lo hace como entrada.



88h o EECON1 : REGISTRO DE PARA EL CONTROL DE LA MEMORIA EEPROM DE DATOS. Este es el registro de control para la memoria de datos y solo destina cinco bits para ello, los más bajos; los tres bits superiores permanecen sin implementar. A continuación se muestran las funciones de estos bits.

REGISTRO EECON1

U bit 7	U bit 6	U bit 5	EEIF bit 4	WRERR bit 3	WREN bit 2	WR bit 1	RD bit 0

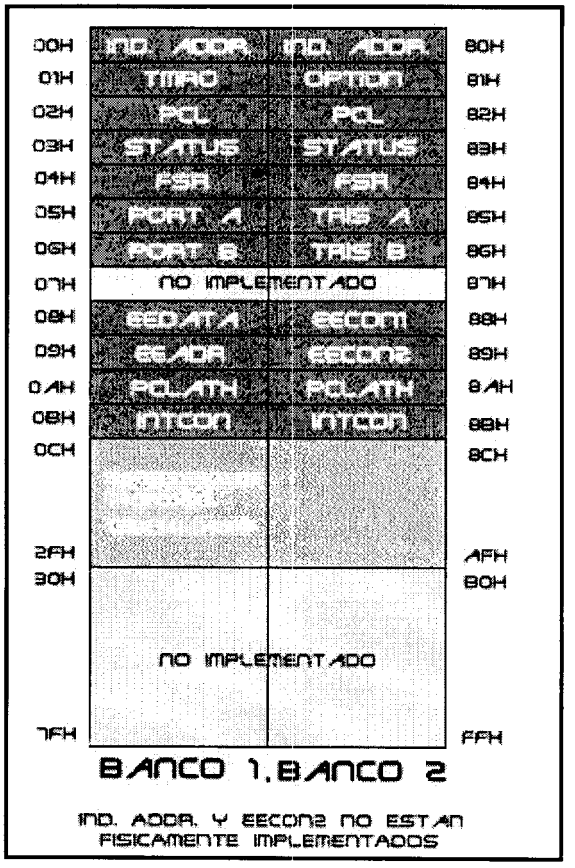
U	Unimplemented. No implementados
EEIF	EEPROM Write Completion Interrup Flag o Bandera de finalización de la escritura. Se coloca en "1" cuando finaliza con éxito la escritura de la EEPROM de datos; se debe colocar en "0" por programa. El bit de habilitación correspondiente es el EEIE, localizado en el registro INTCON.
WRERR	Write Error Flag o Bandera de error de escritura. Si se coloca en "1" cuando la operación de escritura termina prematuramente, debido a cualquier condición de reset.
WREN	Write Enable o habilitación de escritura. Si se coloca en "0" no permite las operaciones de escritura; en "1" las habilita.
WR	Write Control o Control de escritura. Al colocarse en "1" inicia un ciclo de escritura. Este bit sólo es puesto a "0" por hardware, una vez la escritura termina.
RD	Read Control o Control de lectura. Al colocarse en "1" se inicia una lectura de la EEPROM de datos, la cual toma un ciclo de reloj de instrucciones. Este bit sólo se limpia (colocar en "0") por hardware, al finalizar la lectura de la posición de la EEPROM.

89h o EECON2 : EGISTRO AUXILIAR PARA EL CONTROL DE LA MEMORIA EEPROM DE DATOS. Este registro no es implementado físicamente por el microcontrolador, pero que es necesario en las operaciones de escritura en la EEPROM de datos; ante cualquier intento de lectura se tendrán "ceros".

0Ch a 4Fh: EGISTRO DE PROPOSITO GENERAL. Estas 68 posiciones están implementadas en la memoria RAM estática, la cual conforma el área de trabajo del usuario; a ellas también se accede cuando en la pagina 1 (Banco 1) se direccionan las posiciones 8Ch a CFh. Esto se ha diseñado así para evitar un excesivo cambio de paginas en el manejo de la RAM del usuario, agilizando los procesos que se estén llevando a cabo y facilitar la labor del programador.

REGISTRO DE TRABAJO W. Este es el registro de trabajo principal, se comporta de manera similar al acumulador en los microprocesadores. Este registro participa en casi todo el programa y por consiguiente en la mayoría de las instrucciones.

PILA (STACK). Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina (CALL), 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 PIC 16F84 tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problema alguno.



Características Principales:

Algunos elementos que forman parte de los PIC no se encuentran en microcontroladores de otros fabricantes, o simplemente representan alguna ventaja o facilidad a la hora de hacer un diseño. A continuación una corta descripción de las más significativas.

Circuito de Vigilancia (Watchdog TIMER o perro guardián):

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 electromagnético. Esta conformado por un oscilador RC que se encuentra dentro del microprocesador.

Este oscilador corre de manera independiente al oscilador principal. Cuando se habilita su funcionamiento, dicho circuito hace que el microcontrolador sufra un reset cada determinado tiempo (que se puede programar entre 18 µS y 2 segundos). Este reset lo puede evitar el usuario mediante una instrucción especial del microcontrolador (CLRWT: Borra el contenido del watchdog), la cual se debe ejecutar antes de que termine el periodo nominal de dicho temporizador. De esta manera si el programa se ha salido de su flujo normal, por algún ruido o interferencia externa, el sistema se reiniciará (cuando se acabe el tiempo programado y no se haya borrado el contador) y el programa puede restablecerse para continuar con su funcionamiento normal.

En las primeras practicas no se utiliza el circuito de vigilancia para facilitar el trabajo; por eso, en el momento de programar el microcontrolador se debe seleccionar en los fusibles de configuración "watchdog timer OFF". Mas adelante veremos algunos ejemplos que ilustran su funcionamiento y la manera de utilizarlos

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 del sistema. En el momento de grabar el microcontrolador se debe habilitar el fusible de configuración "Power-up Timer", para ello se debe seleccionar "ON". Su tiempo de retardo es de 72 milisegundos.

Modo de bajo consumo (SLEEP):

Esta característica permite que el microcontrolador entre en un estado pasivo donde consume muy poca potencia. Cuando se entra en este modo el oscilador principal se detiene, pero el temporizador del circuito de vigilancia (watchdog) se reinicia y empieza su conteo nuevamente. Se entra en ese estado por la ejecución de una instrucción especial (llamada SLEEP) y se sale de él cuando el microcontrolador sufre un reset por un pulso en el pin MCLR, por que el watchdog hace que se reinicie el sistema o por que ocurre una interrupción al sistema.

Interrupciones:

Este microcontrolador incluye el manejo de interrupciones, lo cual representa grandes ventajas. El PIC16F84 posee cuatro formas de interrupción que son:

- Interrupción externa en el pin RB0/INT
- Finalización del temporizador/contador TMR0
- Finalización de escritura en la EEPROM de datos
- Cambio de estado en los pines RB4 a RB7

El registro 0Bh o INCON1 contiene las banderas de las interrupciones INT, cambio en el puerto B y finalización del conteo del TMR0, al igual que el control para habilitar o deshabilitar cada una de las fuentes de interrupción, incluida la de escritura de la memoria EEPROM. Sólo la bandera de finalización de la escritura reside en el registro 88h o EECON1.

Si el bit GIE (Global Interrup Enable) se coloca en 0, deshabilita todas las interrupciones. Cuando una interrupción es atendida, el bit GIE se coloca en 0 automáticamente pare evitar interferencias con otras interrupciones que se pudieran presentar, la dirección de retorno se coloca en la pila y el PIC se carga con la dirección 04h. Una vez en la rutina de servicio, la fuente de interrupción se puede determinar examinando las banderas de interrupción. La bandera respectiva se debe colocar, por software, en cero antes de regresar de la interrupción, para evitar que se vuelva a detectar nuevamente la misma interrupción.

La instrucción RETFIE permite al usuario retornar de la interrupción, a la vez que habilita de nuevo las interrupciones, al colocar el bit GIE en uno. Debe tenerse presente que solamente el contador de programa es puesto en la pila al atenderse la interrupción; por lo tanto, es conveniente que el programador tenga cuidado con el registro de estados y el de trabajo, ya que se pueden introducir resultados inesperados si dentro de ella se modifican.

Interrupción Externa. Actúa sobre el pin RB0/INT y se puede configurar para activarse con el flanco de subida o el de bajada, de acuerdo al bit INTEDG (Interrup Edge Select Bit, localizado en el registro OPTION). Cuando se presenta un flanco valido en el pin INT, la bandera INTF INTCON) se coloca en uno. La interrupción se puede deshabilitar colocando el bit de control INTE (INTCON) en cero. Cuando se atiende la interrupción, a través de la rutina de servicio, INTF se debe colocar en cero antes de regresar al programa principal. La interrupción puede reactivar al microcontrolador después de la instrucción SLEEP, si previamente el bit INTE fue habilitado

Interrupción por finalización de la temporización. La superación del conteo máximo (0FFh) en el TMR0 colocara el bit TOIF (INTCON) en uno. El bit de control respectivo es TOIE (INTCON).

Interrupción por cambio en el puerto RB. Un cambio en los pines del puerto B (RB4 a RB7) colocará en uno el bit RBIF (INTCON). El bit de control respectivo es RBIE (INTCON).

Interrupción por finalización de escritura. Cuando la escritura de un dato en la EEPROM finaliza, se coloca en 1 el bit EEIF (EECON1). El bit de control respectivo es EEIE (INTCON).

Memoria de datos de la EEPROM:

El PIC 16F84 tiene una memoria EEPROM de datos de 64 posiciones (00h a 3Fh), de 8 bits cada una. Este bloque de memoria no se encuentra mapeado en ningún banco, el acceso a esas posiciones se consigue a través de dos registros de la RAM:

- El registro EEADR (posición 09), que debe contener la dirección de la posición de la EEPROM a ser accesada.
- El registro EEDATA (posición 08), que contiene el dato de 8 bits que se va a escribir o el que se obtuvo de la ultima lectura.

Adicionalmente, existen dos registros de control: el EECON1 (88h), que posee cinco bits que manejan las operaciones de lectura/escritura y el EECON2 (89h), que aunque no es un registro físico, es necesario para realizar las operaciones de escritura.

La lectura toma un ciclo de reloj de instrucciones, mientras que las escritura, por ser controlada por un temporizador incorporado, tiene un tiempo nominal de 10 milisegundos, este tiempo puede variar con la temperatura y el voltaje. Cuando se va a realizar una operación de escritura, automáticamente se hace primero la operación de borrado. El numero típico de ciclos de borrado/escritura de la EEPROM de datos es de 1.000.000.

Fusibles de configuración:

El PIC 16F84 posee cinco fusibles, cada uno de los cuales es un bit. Estos fusibles se pueden programas para seleccionar varias configuraciones del dispositivo: tipo de oscilador, protección de código, habilitación del circuito de vigilancia y el temporizador al encendido. Los bits se localizan en la posición de memoria 2007h, posición a la cual el usuario sólo tiene acceso durante la programación del microcontrolador. Cuando se programa la protección del código, el contenido de cada posición de la memoria no se puede leer completamente, de tal manera que el código del programa no se puede reconstruir. Adicionalmente, todas las posiciones de memoria del programa se protegen contra la reprogramación.

Una vez protegido el código, el fusible de protección solo puede ser borrado (puesto a 1) si se borra toda la memoria del programa y la de datos.

Las PULL - UPS internas:

Cada uno de los pines del puerto B tiene un elemento débil pull-up interno (250 uA típico); este elemento es automáticamente desconectado cuando el pin se configura como salida. Adicionalmente, el bit RBPU (OPTION) controla todos estos elementos, los cuales están deshabilitados frente a una condición de reset. Estos elementos pull-up son especialmente útiles cuando el microcontrolador va a colocarse en el modo de bajo consumo, ya que ayudan a no tener las entradas flotantes, significado una reducción en el consumo de corriente.

Conjunto de Instrucciones:

Todos los modelos de microcontroladores PIC responden a la arquitectura RISC. No solo implica que el numero de instrucciones que se capaz de interpretar y ejecutar sea pequeño (en el caso del Pic 16F84 que consta de 35instrucciones), sino también que consta de las siguientes características :

1. Las instrucciones son simples y rápidas
2. Las instrucciones son ortogonales

   Apenas tiene restricciones en el uso de operadores
3. La longitud de las instrucciones y los datos es constante.

   Todas las instrucciones tienen la misma longitud, 14 bits en los PIC 16X8X, y todos los datos son de un byte. La arquitectura Harvard aísla la memoria de instrucciones de la de datos.

Los diferentes formatos que admiten las instrucciones, se clasifican en cinco grandes grupos, atendiendo al tipo de operación que desarrollan.

1. **Operaciones orientadas a manejar registros de tamaño byte.**

   Se divide en tres campos:

   - Campo de código OP de 6 bits.
   - Campo de la dirección del operando fuente (f) de 7bits.
   - Campo que define el operando destino (d) de 1 bit.
2. **Operaciones orientadas a manejar bits.**

   Este formato consta de tres campos:

   - Campo de código OP de 4 bits
   - Campo de la dirección del registro fuente de 7 bits.
   - Campo de la posición del bit en el registro y es de 3bits.

13	10 9	7 6	0
Código Op	b (Posición)	f(Dirección del Registro.	

3. **Operaciones que manejan un valor inmediato o literal**

Constas de solo dos campos:

- *Campo del código OP con 6 bits.*
- *Campo del valor inmediato (k) con 8 bits.*

4. **Operaciones incondicionales de control del flojo del programa.**

Este tipo instrucciones efectúan al contenido del contador de programa y sirven para romper la secuencia ordenada de las instrucciones del programa. Consta de dos campos:

- *campo del código OP de 3 bits*
- Campo de la dirección del salto que se carga en el contador de programa de 11bits.

5. ***Operaciones de salto condicional***

Se dispone de pocas instrucciones que cuando se cumplen una condición dan un brinco(skip). Un brinco es un salto muy pequeño, es decir que solo se salta una instrucción , la que hay detrás de la condicional.

Las instrucciones de dividen en:

- Instrucciones que manejan registros.

  Responden a la sintaxis mnemónico f, d, siendo f y d los dos operadores fuente y destino.

  El registro f viene reverenciado por la dirección de 7bits que ocupa, mientras que el destino solo por 1, cuando vale 0 es el registro W y si vale 1 es el fuente.

SINTAXIS	OPERACIÓN	CICLOS	FORMATO 14bits	SEÑALIZADORES
ADDWF f,d	suma W y f	1	00 0111 dfff ffff	C,DC,Z
ANDWF f,d	AND W con f	1	00 0101 dfff ffff	Z
CLRF f f	borra f (pone los bits en cero)	1	00 0001 1fff ffff	z
CLRW ---	Borra W	1	00 0001 0xxx xxxx	Z
COMF f,d	Complementa f (invierte)	1	00 1001 dfff ffff	Z
DECF f,d	Decrementa f	1	00 0011 dfff ffff	Z
INCF f,d	Incrementa f	1	00 1010 dfff ffff	Z
IORWF f,d	OR entre W y f	1	00 0100 dfff ffff	Z
MOVF f,d	Mueve f	1	00 1000 dfff ffff	Z
MOVWF f	Mueve W y f	1	00 0000 1fff ffff	---
NOP ---	No opera	1	00 0000 0xx0 0000	---
RLF f,d	Rota f a la izquierda a través del cero	1	00 1101 dfff ffff	C
RRF f,d	Rota f a la derecha a través del cero	1	00 1100 dfff ffff	C
SUBWF f,d	Resta W y f	1	00 0010 dfff ffff	C, DC, Z
SWAPF f,d	Intercambia nobbles	1	00 1110 dfff ffff	---
XORWF f,d	XOR de W con f	1	00 0110 dfff ffff	Z

- **Instrucciones que manejan Bits.**

Solo hay dos instrucciones en este grupo . Una de ellas pone a 1 (bsf) cualquier bit de u registro, mientras que la otra pone a 0 (bcf).

SINTAXIS	OPERACIÓN	CICLOS	FORMATO 14bits	SEÑALIZADORES
BCF f,d	Borra bit de f	1	01 00bb bfff ffff	---
BSF f,d	Pone a 1 el bit de f	1	01 01bb bfff ffff	---

- **Instrucciones de brinco.**

En los PIC de gama media, solo existen cuatro instrucciones de salto condicional. Dos de ellas según su valor (1 o 0) brincan o no , sólo se saltan la instrucción siguiente a la condición . Las dos instrucciones restantes incrementan o decrementan un registro y la posibilidad del brinco se efectúa si con esa operación el valor del registro a llegado a cero. Tardan 2 ciclos de instrucciones cuan do brinca y un ciclo cuando no se realiza el brinco.

SINTAXIS	OPERACIÓN	CICLOS	FORMATO 14bits	SEÑALIZADORES
BTFSC f,d	Explora un bit de f y brinca si vale 0	1 (2)	01 10bb bfff ffff	---
BTFSS f,d	Explora un bit de f y brinca si vale 1	1 (2)	01 10bb bfff ffff	---
DEFSZ f,d	Decrementa f y si es 0 ,brinca	1 (2)	00 1011 dfff ffff	---
INCFSZ f,d	Decrementa f y si es 1 ,brinca	1 (2)	00 1111 dfff ffff	---

- **Instrucciones que manejan operadores inmediatos.**

Consta de seis instrucciones que realizan una operación con un valor inmediato de 8 bits que proporciona dentro del formato , el cual solo tiene dos campos: el del Código OP (6 bits) y el del operador inmediato (8bits).

SINTAXIS	OPERACIÓN	CICLOS	FORMATO 14bits	SEÑALIZADORES
ADDLW k	suma inmediata con W	1	11 111x kkkk kkkk	C,DC.Z
ANDLW K	AND inmediato con W	1	11 1001 kkkk kkkk	Z
IORLW k	OR inmediato con W	1	11 1000 kkkk kkkk	Z
MOVLW k	Mueve a W un valor inmediato	1	11 00xx kkkk kkkk	---
SUBLW k	Resta W de un inmediato	1	11 11xx kkkk kkkk	C,DC.Z
XORLW k	OR exclusiva con W	1	11 1010 kkkk kkkk	---

- **Instrucciones de control y especiales.**

En este grupo se incluyen las instrucciones que rompen la secuencia normal del programa porque alteran el contenido del PC , y las instrucciones especiales.

Entre las instrucciones de control se encuentran cinco : GOTO, CALL, RETURN, RETLW,RETFIE. Mientras que en las instrucciones especiales se encuentran dos : CLRWDT y SLEEP.

SINTAXIS	OPERACIÓN	CICLOS	FORMATO 14bits	SEÑALIZADORES
CALL k	Llamada a subrutina	2	10 0kkk kkkk kkkk	TO#,PD#
CLRWDT	Borra o refresca el perro guardián	1	00 0000 0110 0100	---
GOTO k	salto incondicional	2	10 1kkk kkkk kkkk	---
RETFIE	Retorno de interrupción (GIE =1)	2	00 0000 0000 (00)	---
RETLW k	Retorno subrutina y carga W=k	2	11 01xx kkkk kkkk	---
RETURN	Retorno de subrutina	2	00 0000 0000 1000	---
SLEEP	Pasa al modo de reposo	1	00 0000 0110 0011	TO#,PD#

Simuladores y Ensambladores:

El ***MPLAB*** es un "Entorno de Desarrollo Integrado" (Integrated Development Environment, IDE) que corre en "Windows", mediante el cual Usted puede desarrollar aplicaciones para los microcontroladores de las familias PIC 16/17. Con el MPLAB usted puede escribir, depurar y optimizar los programas de sus diseños con PIC 16/17. El MPLAB incluye un editor de texto, un simulador y un organizador de proyectos. Por otra parte, también soporta al emulador PICMASTER y a otras herramientas de desarrollo de Microchip como el PICSTAR-Plus.

Con el Mplab Usted puede:

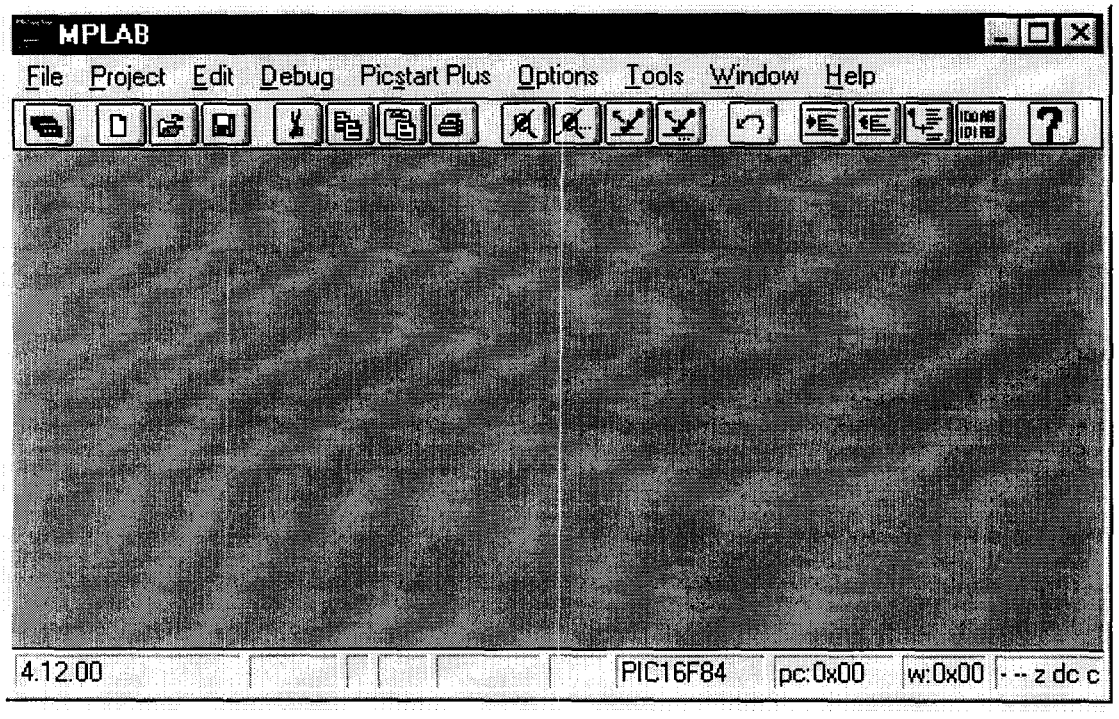
- Depurar sus programas fuentes.
- Detectar errores automáticamente en sus programas fuente para editarlos.
- Depurar los programas utilizando puntos de corte (breakpoints) mediante valores de los registros internos.
- Observar el flujo del programa con el simulador MPLAB-SIM ó seguirlo en tiempo real utilizando el emulador PICMASTER.
- Realizar medidas de tiempo utilizando un cronometro.
- Mirar variables en las ventanas de observación.
- Encontrar respuestas rápidas a sus preguntas, utilizando la ayuda en línea del MPLAB.

Requerimientos Mínimos de HARDWARE y de SOFTWARE:

- PC con procesador 386 o superior.
- 4 MB de memoria RAM
- 8 MB libres de disco duro.
- Monitor VGA.
- Windows 3.11 de Microsoft o una versión posterior.

- Iniciar el Programa: Después de haber bajado e instalado el software, haga doble click en el icono correspondiente a MPLB .

- Definir el tipo de microcontrolador a usar (para el simulador): En el menú "Options", seleccione "Development mode...", ahora despliegue en el menú "MAPLAB-SIM Simulator" y seleccione el tipo de procesador a utilizar (que en nuestro caso es el 16F84), y finalmente oprima "Reset".



Habilitar las herramientas para el simulador: En el menú "Tools", seleccione "Verify PICMASTER..." y a continuación siga las instrucciones correspondientes en los menús emergentes, y ya esta!.

Simulación:

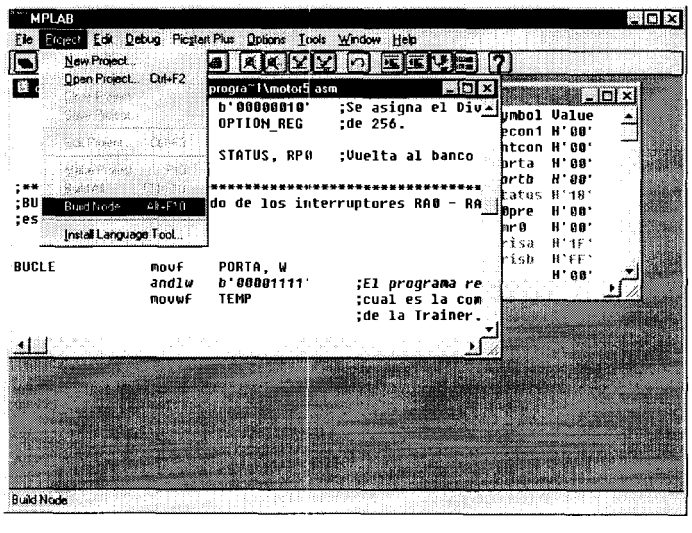
Para realizar la simulación de un programa deben seguirse los siguientes pasos: Partimos de que el usuario ya tenga el programa guardado en el disco duro o disquete el cual desea ensamblar y abierto bajo el MPLAB.

En el menú "Project", seleccione la opción "New Project...", en la ventana emergente de New Project no le modifiques nada, solo presione OK. En la siguiente ventana Edit Project, hacer click en la sección Non-Project Files sobre el nombre del archivo fuente que usted le dio anteriormente. Haga click en el botón <=Add y luego de que éste aparezca en la sección Projects Files: haga click sobre el botón OK.

Salvar el proyecto, en el menú "Project" presione "Save Project".

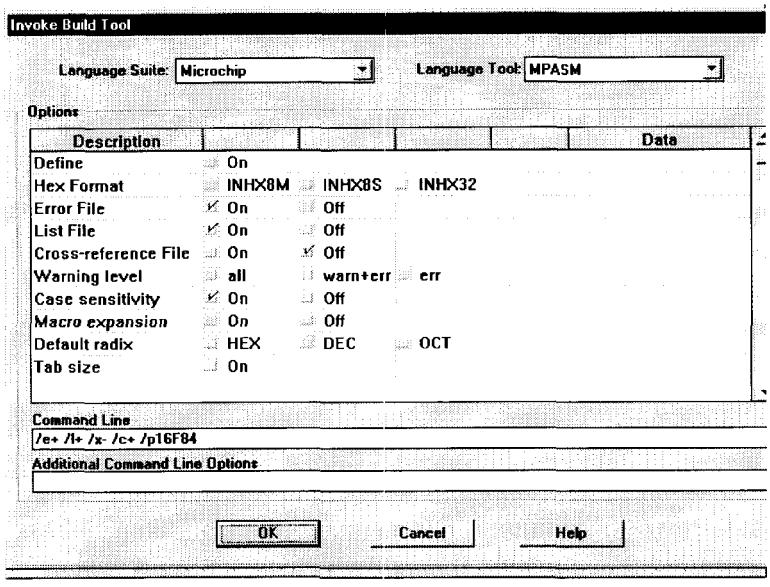
Realizar la construcción de todo el proyecto: En el menú "Project", seleccione "Build All" o si mejor prefiere abreviar con la combinación de teclas

En esta etapa se realiza en forma automática el ensamble de nuestro programa fuente y el vaciado de éste en memoria de simulación. El proceso de ensamble generará un archivo de errores en caso de que estos existan, si es así deben corregirse directamente sobre el archivo fuente, hacer las correcciones necesarias, salvar (guardar el programa) y reconstruir el proyecto. (CTRL + F10)

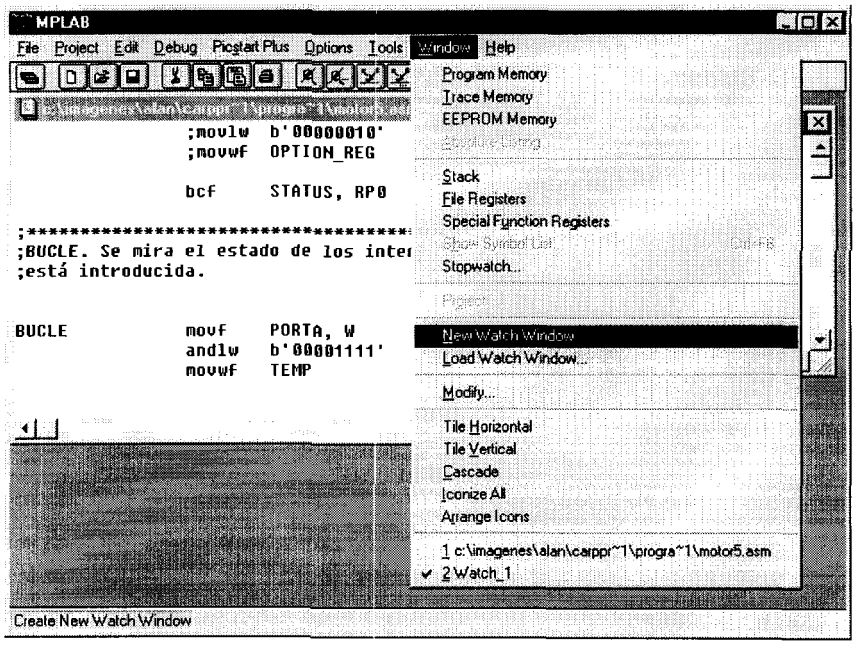


Pasos en la compilación de un programa .

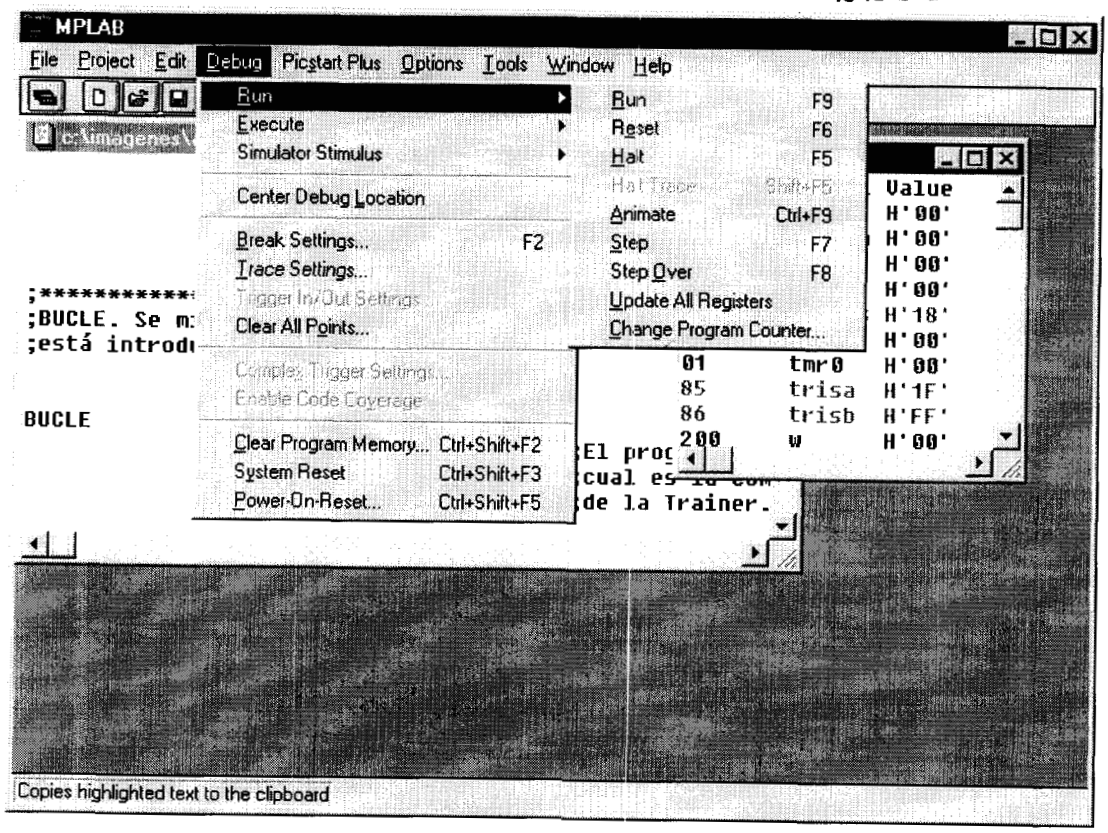
En esta parte se especifica el PIC a utilizar y el modo en que se quiere el archivo HEX.



En esta etapa del proceso ya se tiene el entorno listo para la simulación del programa

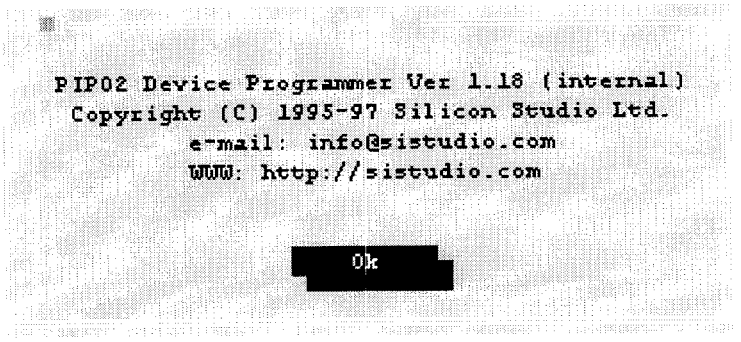


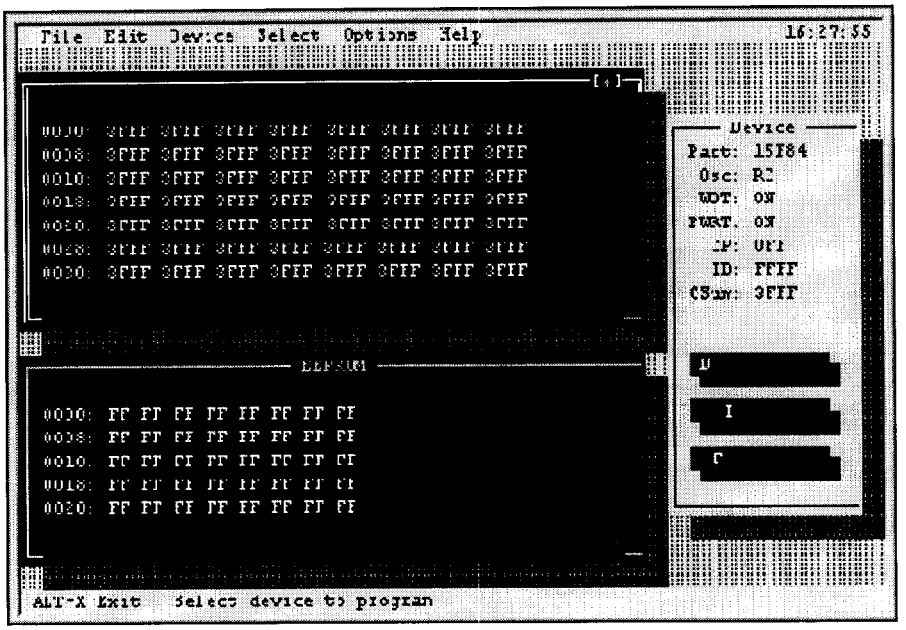
225914


MPLAB
File Project Edit Debug Picstart Plus Options Tools Window Help
Run
Execute
Simulator Stimulus
Center Debug Location
Break Settings... F2
Trace Settings...
Trigger In/Out Settings...
Clear All Points...
Complex Trigger Settings...
Enable Code Coverage
Clear Program Memory... Ctrl+Shift+F2
System Reset Ctrl+Shift+F3
Power-On-Reset... Ctrl+Shift+F5
Run F9
Reset F6
Halt F5
Halt Trace Shift+F5
Animate Ctrl+F9
Step F7
Step Over F8
Update All Registers
Change Program Counter...
Value
H'00'
H'00'
H'00'
H'00'
H'18'
H'00'
01 tmr0 H'00'
85 trisa H'1F'
86 trisb H'FF'
200 w H'00'
;BUCLE. Se m
;está introd
BUCLE
de la Trainer.
Copies highlighted text to the clipboard

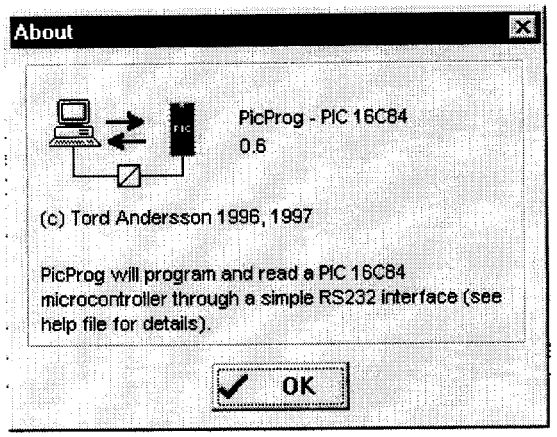
Grabadores de PIC.

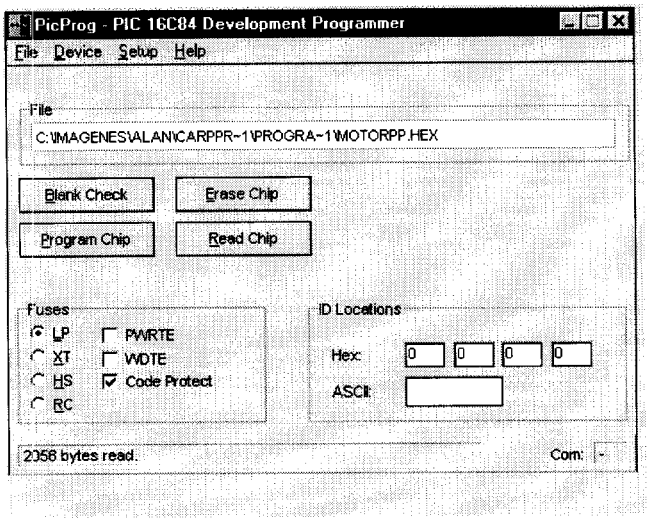
Un grabador de PIC PIPO2 en versión para MS-DOS. El cual no solo graba el PIC 16f84 si no una gran variedad de pic.



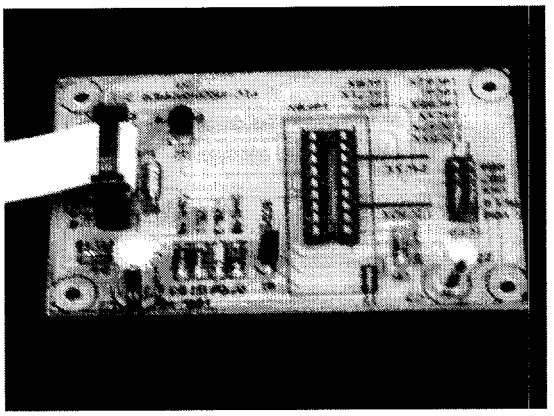


O el PicProg, para Windows, esta versión solo graba o quema el PIC de la familia 16xxx .





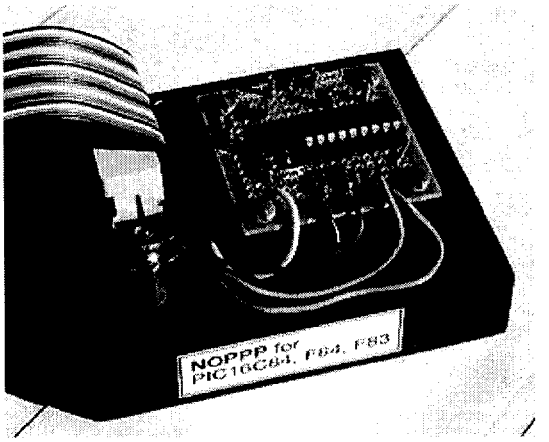
Las imágenes que a continuación se muestran son las del grabador de PIC, los paquetes mencionados anteriormente funcionan con este circuito.



```
--------------------------------
NOPPP - "No-Parts" PIC Programmer
Michael A. Covington
Version of Mar 10 1999 14:40:27
--------------------------------
Using LPT2 on 000H
--------------------------------
--------------------------------

Devices supported:

  C  PIC16C84
  F  PIC16F84
  3  PIC16F83

  T  Test the programmer circuit

Your choice (C,F,3,T): ^C
```

Este es otro grabador para PIC el cual funciona en MS-DOS .

Este es el grabador , el cual funciona conectándolo en el puerto paralelo de la PC y tiene una alimentación externa de 12 y 5 V. es el NOPPP.



Programas

Hemos explicado el funcionamiento de el PIC, pero para entender mejor, mostramos un pequeño ejemplo de como se inicializa los puertos ya sea entrada o salida .

El siguiente programa muestra como utilizar el puerto B como salida.

Define los registros necesario a utilizar.

```
TRISA       EQU   05h
TRISB       EQU   06h

PORTA       EQU   05h
PORTB       EQU   06h
STATUS      EQU   03h
cuenta      EQU   15h
cuenta2     EQU   16h
LUZ         EQU   17h
```

En esta parte se especifica cual es el PIC a programar.

```
list      p=16f84

          ORG  00h    ; vector de reset
          goto INICIO
```

Definimos el puerto B como salida.

```
INICIO    movlw 00
          bsf   STATUS,5     ;registro de configuración
          movwf TRISB        ; puerto b como salidas
          bcf   STATUS,5     ;puerto
```

A continuación mostramos el desarrollo del programa.

```
TRISA       EQU   05h
TRISB       EQU   06h
PORTA       EQU   05h
PORTB       EQU   06h
STATUS      EQU   03h
cuenta      EQU   15h
cuenta2     EQU   16h
LUZ         EQU   17h

list      p=16f84

          ORG  00h    ; vector de reset
          goto INICIO
```

```
INICIO      movlw 00
            bsf   STATUS,5      ;registro de configuración
            movwf TRISB         ; puerto b como salidas
            bcf   STATUS,5      ;puerto

comienza    movlw 01h
            movwf LUZ
aqui        rlf   LUZ,0
            movwf PORTB
            movwf LUZ
            call  retardo
            btfss LUZ,7
            goto  aqui
ahi         rrf   LUZ,0
            movwf PORTB
            movwf LUZ
            call  retardo
            btfss LUZ,0
            goto  ahi
            goto  comienza
retardo     movlw 99h           ;modificando este dato aumenta o
            movwf cuenta        ;disminuye la velocidad...
retar1      decfsz cuenta,1
            goto  retar1
            movlw 99h           ;o modificando este otro
            movwf cuenta2
retard      decfsz cuenta2,1
      goto  retard
            return
            END
```

Para definir un puerto como salida se puede utiliza el siguiente ejemplo, en este caso tomamos al puerto A como entrada.

```
movlw 0x0F
            bsf STATUS,5    ;ponemos el bit 5 del registro STATUS a "1"
            movwf TRISA     ;Puerto A como entrada
            movlw 0x00
```

La siguiente rutina lee el puerto, verificando si hay señal de entrada.

```
mira_fila   movf  RA,w      ;leemos RA para ver si hay alguna tecla pulsada
            addlw 0x0F
            sublw 0x0E
            movwf byte      ;comparamos y si no es 00001111 es que se ha pulsado
            decfsz byte,f
            goto alli
            return          ;si es 00001111 retorna
  alli      movf  RA,w
```

```
                movwf fila
                call retardo     ;antirebotes
                btfsc fila,1     ;miramos si el bit 1 es cero de RA
                goto otro
                movlw 0x00
                addwf colum      ;sumamos para fila 1, "00" y la columna
                movwf byte       ;guardamos en byte
                return
otro        movf  RA,w
                movwf fila
                call retardo     ;antirebotes
                btfsc fila,2     ;miramos si el bit 2 es cero
                goto otro1
                movlw 0x10
                addwf colum
                movwf byte
                return
otro1       movf  RA,w
                movwf fila
                call retardo     ;antirebotes
                btfsc fila,3     ;miramos si el 3 bit es cero
                goto otro2
                movlw 0x20
                addwf colum
                movwf byte
                return
otro2       movlw 0x30
                addwf colum      ;si no es ninguno de los otros 3 será este
                movwf byte
                return
```

El siguiente ejemplo muestra una rutina de retardo, muy útil en nuestros programas.

```
retardo         movlw   0xFF
                movwf   cuenta
retado          decfsz  cuenta,f
                goto    retado
                movlw   0x99
                movwf   cuenta2
rateado                 decfsz  cuenta2,f
                goto    rateado
```

Los programas que se muestran a continuación son los desarrollados en nuestra tarjeta. Que consisten en rutinas para mover un motor ya sea a paso o de corriente directa.

El siguiente programa se encarga de mover un motor a pasos.

```
PCL         EQU    H'0002'
STATUS      EQU    H'0003'
FSR         EQU    H'0004'
PORTA       EQU    H'0005'
PORTB       EQU    H'0006'
EEDATA              EQU       H'0008'
EEADR               EQU       H'0009'
PCLATH      EQU    H'000A'
INTCON      EQU    H'000B'

OPTION_REG    EQU    H'0081'
TRISA       EQU    H'0085'
TRISB       EQU    H'0086'
EECON1              EQU       H'0088'
EECON2              EQU       H'0089'

;Definición de los bits del registro INTCON

GIE         EQU    H'0007'
EEIE                EQU       H'0006'
T0IE        EQU    H'0005'
INTE        EQU    H'0004'
RBIE        EQU    H'0003'
T0IF        EQU    H'0002'
INTF        EQU    H'0001'
RBIF        EQU    H'0000'

;Definición de los bits del registro OPTION

NOT_RBPU      EQU    H'0007'
INTEDG      EQU    H'0006'
T0CS        EQU    H'0005'
T0SE        EQU    H'0004'
PSA         EQU    H'0003'
PS2         EQU    H'0002'
PS1         EQU    H'0001'
PS0         EQU    H'0000'
j                   EQU       H'1f'
k                   EQU       H'1E'
m                   EQU       H'1D'
l                   EQU       H'1C'
mm                  EQU       H'1B'


;Definición de los bits del registro STATUS

IRP         EQU    H'0007'
RP1         EQU    H'0006'
RP0         EQU    H'0005'
NOT_TO      EQU    H'0004'
NOT_PD      EQU    H'0003'
Z           EQU    H'0002'
DC          EQU    H'0001'
C           EQU    H'0000'

;Definición de los bits del registro EECON1

EEIF            EQU H'0004'
WRERR               EQU       H'0003'
WREN                EQU       H'0002'
WR                  EQU       H'0001'
RD                  EQU       H'0000'
TEMP                EQU       0X0E        ;Registros de propósito general.
TEMP2               EQU       0x0F
```

```
                                LIST

                list        p = 16f84   ;Elección del modelo del PIC
                RADIX       HEX         ;Sistema de numeración hexadecimal.

                ORG         0x00        ;El programa comienza en la dirección 0.
                goto        INICIO

                ORG         5           ;Se salta el vector interrupción.

;*****************************************************************************************
;Rutinas de DELAY.
DELAY10
        movlw  0x0f
        movwf  j
jlp1:   movwf  k
        movwf  l
klp2    decfsz l,f
        goto klp2
klp1    decfsz k,f
        goto klp1
        decfsz j,f
        goto   jlp1
        RETURN

DELAY20
        movlw  0x06
        movwf  j
jloop:  movwf  k
kloop   decfsz k,f
        goto kloop
        decfsz j,f
        goto   jloop
                        RETURN
;*************************************************************
; rutina pon 0 en en bufer

rucero

        movlw   0x00
        movwf   PORTB
        movlw   0xC0
        movwf   PORTB
        call    DELAY10

                RETURN

;*****************************************************************************************

;rutina adelante
ADELANTE
        movlw   0x05
        movwf   PORTB
        movlw   0x85
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10

        movlw   0x09
        movwf   PORTB
```

```
        movlw   0x89
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10

        movlw   0x0A
        movwf   PORTB
        movlw   0x8A
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10

        movlw   0x06
        movwf   PORTB
        movlw   0x86
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10

        clrw
        movlw   m
                RETURN
;*****************************************************************************************
;rutina atras
ATRAS
        movlw   0x05
        movwf   PORTB
        movlw   0x85
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10

        movlw   0x06
        movwf   PORTB
        movlw   0x86
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
```

```
        call    DELAY10

        movlw   0x0A
        movwf   PORTB
        movlw   0x8A
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10

        movlw   0x09
        movwf   PORTB
        movlw   0x89
        movwf   PORTB
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10
        call    DELAY10

        clrw
        movlw   m
                RETURN

;*****************************************************************************************

INICIO          bsf     STATUS, RP0  ;Banco 1.
                movlw   B'00000111'  ;Se configura RA0, RA1 Y RA2 como entradas y el resto de la
                movwf   PORTA        ;puerta A como salida.
                clrf    PORTB        ;PuertaB como salida.
                bcf     STATUS, RP0  ;Vuelta al banco 0.

;*****************************************************************************************

BUCLE
        movf    PORTA, W
        andlw   b'00000111'
        movwf   TEMP

        xorlw   b'00000001'
        btfsc   STATUS, Z
        goto    atra
        movf    TEMP, W

        xorlw   b'00000000'
        btfsc   STATUS, Z
        goto    cero
        movf    TEMP, W


        goto    BUCLE
;*****************************************************************************************

atra
        call    rucero
        call  DELAY10
salto2
        call  ATRAS
```

```
        decfsz    m,f
        goto      salto2
        movwf     PORTA
    call  DELAY20
        clrf      PORTA
        call      DELAY20
        movlw     b'00011000'
        movwf     PORTA
    call  DELAY20
        GOTO      BUCLE

;*****************************************************************************************
cero
        call      rucero
        call      DELAY10

        movlw     0x4B
        movwf     m
salto1
        call   ADELANTE
        decfsz    m,f
        goto      salto1
        movwf     PORTA
    call  DELAY20
        clrf      PORTA
        call      DELAY20
        movlw     b'00011000'
        movwf     PORTA
    call  DELAY20

        GOTO      BUCLE

;*****************************************************************
slep

        END
```

El programa que a continuación se muestra se encarga de mover un motor de corriente directa.

```
                        LIST

                        NOLIST

W         EQU    H'0000'
F         EQU    H'0001'

;Definición de los registros generales

INDF              EQU    H'0000'
TMR0        EQU    H'0001'
PCL         EQU    H'0002'
STATUS      EQU    H'0003'
FSR         EQU    H'0004'
PORTA       EQU    H'0005'
PORTB       EQU    H'0006'
EEDATA            EQU      H'0008'
EEADR             EQU      H'0009'
PCLATH      EQU    H'000A'
INTCON      EQU    H'000B'
```

```
OPTION_REG     EQU    H'0081'
TRISA          EQU    H'0085'
TRISB          EQU    H'0086'
EECON1                EQU        H'0088'
EECON2                EQU        H'0089'

;Definición de los bits del registro INTCON

GIE            EQU    H'0007'
EEIE                  EQU        H'0006'
T0IE           EQU    H'0005'
INTE           EQU    H'0004'
RBIE           EQU    H'0003'
T0IF           EQU    H'0002'
INTF           EQU    H'0001'
RBIF           EQU    H'0000'

;Definición de los bits del registro OPTION

NOT_RBPU       EQU    H'0007'
INTEDG         EQU    H'0006'
T0CS           EQU    H'0005'
T0SE           EQU    H'0004'
PSA            EQU    H'0003'
PS2            EQU    H'0002'
PS1            EQU    H'0001'
PS0            EQU    H'0000'
j                     EQU        H'1f'
k                     EQU        H'1E'
;Definición de los bits del registro STATUS

IRP            EQU    H'0007'
RP1            EQU    H'0006'
RP0            EQU    H'0005'
NOT_TO         EQU    H'0004'
NOT_PD         EQU    H'0003'
Z              EQU    H'0002'
DC             EQU    H'0001'
C              EQU    H'0000'

;Definición de los bits del registro EECON1

EEIF           EQU H'0004'
WRERR                 EQU        H'0003'
WREN                  EQU        H'0002'
WR                    EQU        H'0001'
RD                    EQU        H'0000'
TEMP                  EQU        0X0E          ;Registros de propósito general.
TEMP2                 EQU        0x0F

                                    LIST

                      list        p = 16f84   ;Elección del modelo del PIC
                      RADIX       HEX           ;Sistema de numeración hexadecimal.

                      ORG         0x00          ;El programa comienza en la dirección 0.
                      goto        INICIO

                      ORG         5         ;Se salta el vector interrupción.

;*****************************************************************************************
;Rutinas de DELAY.

DELAY10
            movlw  d'60'
            movwf  j
jloop:  movwf  k
```

```
kloop   decfsz k,f
        goto kloop
        decfsz j,f
        goto   jloop
                        RETURN

;*****************************************************************************************

INICIO          bsf     STATUS, RP0   ;Banco 1.
                movlw   B'00001111'   ;Se configura RA0, RA1 Y RA2 como entradas y el resto de la
                movwf   PORTA         ;puerta A como salida.
                clrf    PORTB         ;PuertaB como salida.
                ;movlw  b'00000010'   ;Se asigna el Divisor de Frecuencia al TMR0 con un preescaler
                ;movwf  OPTION_REG    ;de 256.

                bcf     STATUS, RP0   ;Vuelta al banco 0.

;*****************************************************************************************
;BUCLE. Se mira el estado de los interruptores RA0 - RA2 de la Trainer para saber que velocidad
;está introducida.


BUCLE           movf    PORTA, W
                andlw   b'00001111'     ;El programa realiza diversas máscaras para saber
                movwf   TEMP            ;cual es la combinación introducida por los conmutadores
                                        ;de la Trainer.

                xorlw   b'00000001'
                btfsc   STATUS, Z
                goto    ADELANTE
                movf    TEMP, W

                xorlw   b'00000010'
                btfsc   STATUS, Z
                goto    ATRAS
                movf    TEMP, W


                xorlw   b'00000100'
                btfsc   STATUS, Z
                goto    DERECHA
                movf    TEMP, W

                xorlw   b'00001000'
                btfsc   STATUS, Z
                goto    BR_IZQ
                movf    TEMP, W

                xorlw   b'00000000'
                btfsc   STATUS, Z
                goto    muerte
                movf    TEMP, W

                goto    BUCLE

;*****************************************************************************************
ADELANTE                movlw   0x04
                        movwf   PORTB
                        movlw   0x4A
                        movwf   PORTB
                        call    DELAY10
                        movlw   0x07
                        movwf   PORTB
                        movlw   0x87
                        movwf   PORTB
                        clrf    PORTA
                        call    DELAY10
```

```
                movlw   b'00011000'
                movwf   PORTA
        call    DELAY10
                GOTO    BUCLE

ATRAS
                movlw   0x05
                movwf   PORTB
                movlw   0x45
                movwf   PORTB
                call    DELAY10
                movlw   0x07
                movwf   PORTB
                movlw   0x87
                movwf   PORTB
                clrf    PORTA
                call    DELAY10
        movlw   b'00011000'
                movwf   PORTA
        call    DELAY10
                GOTO    BUCLE


DERECHA
                movlw   0x09
                movwf   PORTB
                movlw   0x49
                movwf   PORTB
                clrf    PORTA
                call    DELAY10
                movlw   b'00011000'
                movwf   PORTA
        call    DELAY10
                GOTO    BUCLE

BR_IZQ
                movlw   0x02
                movwf   PORTB
                movlw   0x82
                movwf   PORTB
                CLRF    PORTA
                call    DELAY10
                movlw   b'00011000'
                movwf   PORTA
        call    DELAY10

                GOTO    BUCLE
muerte
                movlw   0x00
                movwf   PORTB
                movlw   0xC0
                movwf   PORTB
                CLRF    PORTA
                call    DELAY10
                movlw   b'00011000'
                movwf   PORTA
        call    DELAY10

                GOTO    BUCLE

;*****************************************************************************************

slep

        END             ;Fin del programa.
```

Conclusiones

Conclusiones:

En este trabajo mostramos que es un Robot, su funcionamiento y características principales , observando la tendencia a seguir , en la de crear autómatas cada ves mas parecidos a la naturaleza o vida artificial.

Mediante el uso de microcontroladores ,en determinadas tareas, estas se pueden realizar con mas facilidad y teniendo un mayor control en los dispositivos .

El PIC es un microcontrolador encargado de controlar estos dispositivos a un bajo costo y en un tamaño reducido de espacio.

El PIC 16F84 de Mirochip, es el microcontrolador que utilizamos para desarrollar nuestra tarjeta controladora , debido a su bajo costo, su memoria de datos EEPROM de 64 bytes, una memoria de programa tipo Flash de 1K, una memoria de datos RAM de 68 bytes, su facilidad de polarización, arquitectura , que costa se 35 instrucciones lo que lo hacen fácil de programar, la amplia información localizada en Internet, etc.

Por lo tanto podemos concluir que el uso y desarrollo de este microcontrolador, nos permite realizar tareas o trabajos específicos, siendo este una gran herramienta en el desarrollo de sistemas de control, con el fin de crear un autómata.

Bibliografía y Referencias WEB

Bibliografía

Robotica practica Tecnología y aplicaciones.

Angulo Usategui, José Ma.

Robot, hombres y mente.

Guadarrama , Madrid.

Manual del PIC 16F84 de Microchip.

Microcontroladores PIC .

Diseño practico de aplicaciones.

José María Angulo Usategui.

Ignacio Angulo Martinez.

MC. Graw Hill.

Saber Electrónica

Numeroso 108,109 y 110.

Articulo de Horacio D. Vallejo.

Editorial Televisa Internacional.

Referencias WEB:

http://skyscraper.fortunecity.com/email/250/pic.htm

http://members.aol.com/_ht_a/Fickpci/pic.htm

http://www.qsl.net/lz2rr/pic.html

http://www.microchip.com/

http://ranier.hq.nasa.gov/telerobotics_page/coolrobots.html

http://members.xoom.com/carlos_76/robot/

http://www.chi.itesm.mx/~cim/robind/robotica.html

http://sunserv.fei.uv.mx/robot/robot.htm

http://www.elen.utah.edu/~osantos/robotica.html

http://ranier.oact.hq.nasa.gov/telerobotics.html. 1997.

http://robotics.eecs.berkeley.edu/~mcenk/medical/index.html.

http://www.aircenter.net/gaia/