UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

INGENIERÍA MECÁNICA ELÉCTRICA

Desarrollo de un caso práctico para obtener el titulo de Ingeniero Mecánico Eléctrico.

“Propuesta de prácticas de microprocesadores y microcontroladores.”

Presenta: Edgar Arciba Guevara.

Asesor: M. en I. Fernando Macedo Chagolla

A mi madre.
Por darme la vida y dirigir mis primeros pasos para ser el hombre que soy ahora. Sin tu apoyo y cariño no estaría aquí.

A mi padre.
Por todo el esfuerzo y amor que invirtió en mi educación para hacerme un hombre responsable y trabajador. Eres un gran hombre.

A mis hermanas.
Para que este trabajo sirva de inspiración para que sobresalgan y se conviertan en personas de provecho para la sociedad y para si mismas.

A Martha mi esposa.
Por que llevas en tu vientre el fruto de nuestro amor y por todo el apoyo, paciencia y amor que me has dado. Recuerda somos uno.

A Canek mi hijo.
Ya que aunque aun no naces me has hecho muy feliz desde el momento en que supe que venias al mundo y has resultado una gran motivación para superarme.

A Fernando Macedo.
Por ser la gran persona que es, por haberme apoyado en la elaboración de este trabajo y simplemente por su amistad. Gracias.

A la UNAM
Por haberme formado como profesionista , como persona y haberme dado el tesoro más grande: la educación.

Por mi raza hablara el espíritu.

Si puedes ver los detalles más
pequeños podrás conocer el universo

ÍNDICE

OBJETIVO GENERAL DEL TRABAJO.

Analizar la teoría básica de funcionamiento de los microprocesadores y microcontroladores para comprender que son, como se programan y poder diseñar sistemas con ellos para resolver problemas de ingeniería.

INTRODUCCIÓN.

Los microprocesadores y microcontroladores son el elemento principal de control en muchos procesos industriales (control de velocidad de motores, control de temperatura de los hornos de fundidoras de acero, etc.) y en la gran mayoría de los aparatos electrónicos actuales (televisiones, computadoras, etc.). Por ejemplo cuando se enciende una televisión, se oprime la tecla de encendido (power) y el TV enciende. En este ejemplo el micro recibe la orden de encendido proveniente de la tecla power, busca en su memoria la parte del programa correspondiente al encendido, la procesa y ejecuta otra orden por medio de un cambio de estado lógico en otra de sus terminales que le indica a la fuente de alimentación que debe de energizar al resto del equipo para que este pueda trabajar. Además de controlar el encendido, también controla el resto de las funciones del equipo (volumen, cambio de canal, etc.).

Con el ejemplo anterior nos podemos dar cuenta de la importancia de estos dispositivos en los equipos electrónicos. Debido a esto un buen ingeniero dedicado al área eléctrica-electrónica debe de conocer los conceptos básicos de estos dispositivos y utilizar estos conocimientos en la implementación y diseño de sistemas con microprocesadores. La mejor forma de comprender de manera adecuada estos conceptos es practicando con ellos por lo que en el presente trabajo hemos diseñado un conjunto de prácticas que abordan los temas básicos de estos dispositivos las cuales se complementan con la información teórica presentada.

La idea de la realización de este manual de prácticas se dio debido a que los planes y programas de estudios de la carrera de Ingeniería Mecánica Eléctrica están en proceso de modificación y modernización para que la preparación de los futuros ingenieros sea de calidad y sean competitivos al momento de ingresar en el campo laboral. Los planes y programas de estudios fueron discutidos y aprobados por el comité de carrera durante sus sesiones de trabajo del mes de Enero del presente año y actualmente se encuentran en el consejo universitario en espera de su aprobación para poder ser implementado a partir del semestre 2009-1.

Debido a esto todas las materias han sufrido modificaciones en sus programas y la materia de microprocesadores no es la excepción. El primer cambio que sufrió esta materia es en el nombre, ya que en el nuevo plan de estudios se llamara "Microprocesadores y microcontroladores" y el segundo cambio que sufrió es que ya no se especifica un dispositivo para estudiarlo si no que ahora es libre, es decir el profesor decidirá a su criterio el dispositivo que utilizara para desarrollar el temario en el cual se especifican de una manera general los temas que se deben de tratar en la materia.

La materia de microprocesadores y microcontroladores es una materia de índole teórico-práctica y el presente trabajo esta elaborado para servir como manual de prácticas para el laboratorio de la materia y comprender de forma práctica los conceptos básicos que se aprenden en la teoría. En esta materia a diferencia de otras que llevan un laboratorio complementario a la teoría, **no existe un manual de prácticas establecido** en el cual se pueda basar el profesor de laboratorio, por lo que actualmente el profesor del laboratorio da una serie de prácticas que muchas veces no tienen una secuencia lógica que permitan la comprensión práctica de los conceptos teóricos fundamentales de la materia. Pensando en esta problemática se elaboro el presente manual de prácticas haciendo notar que existen en el mercado una diversidad de marcas de fabricantes, los cuales a su vez tienen una gran cantidad de estos dispositivos con diferentes características. Independientemente de la marca o del dispositivo utilizado todos tienen en común que la teoría básica de funcionamiento es la misma para todos estos dispositivos y que las diferencias radican en la forma de programarse y en los elementos especifico con los que cuenta cada dispositivo.

El objetivo del presente trabajo no es explicar cada uno de estos dispositivos si no exponer de forma práctica la teoría básica de los microprocesadores y microcontroladores utilizando solo uno de estos dispositivos ya que una vez comprendida la teoría básica de funcionamiento de uno de estos dispositivos se tendrán los conocimientos suficientes para poder utilizar cualquier dispositivo con mayores prestaciones tomando en cuenta sus características (las cuales se encuentran en el manual del fabricante). Para lograr esto se eligió el microcontrolador PIC 16F84A de la empresa Microchip por ser uno de los dispositivos mas sencillos y baratos que se encuentran en el mercado y por estas características es ideal para ser utilizado con fines didácticos

En el proceso de aprendizaje es necesario cimentar los conocimientos empezando por entender lo más básico, ya que las bases cimentadas permiten comprender de una manera más fácil los conocimientos más

avanzados y complejos. Difícil es poder imaginar los complicados procesos que se llevan a cabo dentro de una computadora actual si no somos capaces de entender el funcionamiento de los dispositivos mas simples en los que se basan estas tecnologías y por ello escogimos el microcontrolador PIC16F84A que es un dispositivo muy simple y sencillo de usar.

Razones para usar el PIC 16F84A

- Contiene un microprocesador y los periféricos necesarios que necesita este (memoria ROM, RAM, etc.) en un chip. Por esta razón se reduce el numero de componentes y por tanto el tamaño del circuito.
- Es barato.
- Fácil de usar.
- Fácil de conseguir.
- Los programas ensambladores son gratuitos.
- Existe mucha información de el.
- Permite conocer de manera sencilla el funcionamiento de un sistema con microprocesador.
- Actualmente se prefiere el uso de un sistema con microcontrolador, a menos que se tenga que controlar varios procesos.
- Los sistemas con PIC's son tecnología actual.
- Una vez que el alumno pueda manejar este microcontrolador podrá manejar otros microcontroladores ya que el PIC 16F84A es la puerta de entrada al manejo de dispositivos mas complejos.

En las prácticas propuestas no utilizaremos un microprocesador debido a que el microcontrolador ya tiene uno en su interior además de sus dispositivos de memoria

En este trabajo se plantean siete prácticas cuyos temas fueron tomados del temario de la materia de microprocesadores y microcontroladores que esta en proceso de aprobación. El temario de prácticas propuesto es el siguiente:

Practica 1	Proceso de grabación de un microcontrolador PIC.
Práctica 2	Programación elemental del PIC.
Práctica 3	Conexión del PIC 16F84A con dispositivos periféricos.
Práctica 4	Saltos.
Práctica 5	Subrutinas de tiempo.
Práctica 6	Comunicación serie
Práctica 7	Interrupciones.

Los temas fueron elegidos debido a la importancia que tienen para lograr la comprensión de la teoría básica de estos dispositivos además que desde la primera práctica el alumno arma un programador de PIC's y programa un microcontrolador lo que permite el poder familiarizarse con los procesos que intervienen cuando se trabaja con estos dispositivos (diseñar el código fuente, ensamblar, grabar, etc.) además de que permite la motivación e interés por parte del alumno ya que puede comprobar de manera científica (teórico-práctica) los conceptos de estos dispositivos, permitiendo así cumplir el objetivo de exponer la teoría básica de los microprocesadores y microcontroladores de forma práctica.

Capitulo 1

LA MATERIA DE MICROPROCESADORES Y MICROCONTROLADORES EN LA CARRERA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA EN LA FES ARAGÓN.

1.1 Propuesta de creación de la carrera de Ingeniería Eléctrica Electrónica en la FES Aragón.

Las necesidades del país demandan la competitividad y especialización en todos los campos profesionales, cualidades que han adquirido una vital relevancia. Previendo esta situación, la Licenciatura de Ingeniería Mecánica Eléctrica, con tres áreas terminales Ingeniería Mecánica, Ingeniería Industrial e Ingeniería Eléctrica Electrónica, de la Facultad de Estudios Superiores Aragón solicitó al Consejo Técnico de la Facultad la separación de las áreas en tres licenciaturas nuevas: Ingeniería Mecánica, Ingeniería Industrial e Ingeniería Eléctrica Electrónica (Acuerdo No 10022 del 22 de abril del 2005).

La evolución de dicha profesión orientada a dar respuesta a los planteamientos de la sociedad en sus actividades productivas y necesidad de satisfactores, con el uso de nuevas técnicas y tecnologías, contribuye a la caracterización de cada especialidad de la ingeniería eléctrica electrónica. Los avances científicos que se van logrando en las diferentes áreas del conocimiento y las innovaciones tecnológicas que se desarrollan rápidamente debido a la investigación y a la competencia tecnológica, van señalando rumbos para el ejercicio de la profesión.

Actualmente, la demanda de profesionales en el área eléctrica electrónica se hace con base en un perfil de especialización específico, por lo cual los egresados de la Licenciatura de Ingeniería Mecánica Eléctrica (IME) tienen menor demanda y su nivel de especialización es menor en comparación con los egresados de Ingeniería Eléctrica Electrónica.

En este contexto, la Licenciatura de Ingeniería Eléctrica Electrónica tiene como objetivo formar profesionales que utilicen las ciencias físico-matemáticas y las técnicas de la ingeniería de control, la electrónica y electricidad, además de los principios de las telecomunicaciones en

beneficio de la sociedad, y para mejorar la integración y el funcionamiento de los sistemas tecnológicos.

El campo de desarrollo que abarca el perfil propuesto para la Licenciatura de Ingeniería Eléctrica Electrónica es la generación, la transmisión y la distribución de la energía eléctrica; la planeación, el diseño, la construcción de instalaciones eléctricas residenciales e industriales; la fabricación y diseño de sistemas electrónicos para uso definidos; el diseño, construcción y mantenimiento de sistemas de control e instrumentación de procesos industriales; el establecimiento y diseño de los sistemas de telecomunicaciones, además del ejercicio de la docencia y la realización de investigación.

Cabe mencionar, que la propuesta del perfil planteado por la Facultad de Estudios Superiores Aragón (FES Aragón) para la Licenciatura de Ingeniería Eléctrica Electrónica, es resultado de varios estudios que se ven reflejados en el documento titulado "Diagnóstico de la carrera de Ingeniería Mecánica Eléctrica en la ENEP Aragón 2004". Debido a los resultados de este diagnóstico no sólo se propuso el dividir la carrera de IME en tres carreras, sino también la creación de nuevas asignaturas, la simplificación de otras y la introducción de nuevas áreas del conocimiento de acuerdo a la demanda del mercado laboral.

Actualmente existe demanda de un Ingeniero Eléctrico Electrónico con un valor agregado, es decir, que tenga los conocimientos inherentes a su profesión además de algunas habilidades complementarias que permitan su interacción social, por ello se decidió integrar asignaturas optativas de carácter socio humanísticas y la acreditación del idioma inglés como segunda lengua, con la finalidad de generarle el valor agregado que la sociedad demanda de él.

Como se puede observar, la importancia que tendrá el ingeniero eléctrico electrónico de la FES Aragón en la definición del desarrollo y la satisfacción de las necesidades de la sociedad hace indispensable que éste cuente con una formación educativa sólida, que le permita una evaluación permanente de sus resultados en términos integrales y en relación con la sociedad.

Es por lo anterior que la Licenciatura de Ingeniería Eléctrica Electrónica de la FES Aragón está planeada para formar a los ingenieros eléctricos y electrónicos que demanda el país.

1.2 Objetivo general del plan de estudios propuesto de la carrera de Ingeniería Eléctrica Electrónica

El objetivo de la Licenciatura de Ingeniería Eléctrica Electrónica es formar profesionales de alto nivel, capaces de aplicar y crear tecnología en el campo de la electricidad, telecomunicaciones, control y de la electrónica mediante la aplicación de los conocimientos adquiridos durante su formación académica para resolver y evaluar, con eficiencia y calidad, los problemas y necesidades que la industria y la sociedad le soliciten en el área de la ingeniería eléctrica y electrónica.

1.3 Perfiles

1.3.1 Perfil de ingreso

El aspirante, además de haber cursado el bachillerato, deberá contar con sólidos conocimientos de física, química y matemáticas.
Asimismo, es importante que posea:

- Aptitud para detectar, definir y aplicar el razonamiento científico al estudio y la solución de problemas prácticos, en el campo de la física y en especial de los fenómenos eléctricos.
- Capacidad para dirigir el trabajo en equipo.
- Habilidad para el manejo de diferentes paquetes de biblioteca y lenguajes de computación.
- Inventiva y creatividad.
- Actitud responsable, positiva y emprendedora, a fin de realizar con seguridad y confianza en sí mismo, las tareas que le implicará el ejercicio de su profesión.
- Manejo del idioma inglés.

1.3.2 Perfil de egreso

Conocimientos

El egresado de la Licenciatura de Ingeniería Eléctrica Electrónica de la FES Aragón, utilizará los conocimientos de las ciencias exactas, matemáticas, administrativas y de ingeniería para desarrollar su actividad profesional en aspectos tales como diseño y rediseño de sistemas electrónicos, manejo e implantación de sistemas de control industrial, manejo y administración de sistemas de telecomunicaciones, además de la generación y transmisión de energía eléctrica. Esta formación le permitirá participar activamente en las diversas ramas de la

ingeniería y responder a las necesidades productivas y de servicios que requiere la sociedad.

Conocimientos específicos

- Conceptos matemáticos que le permitan un ejercicio profesional de calidad.
- Conceptos de física que le permitan sentar las bases de su formación.
- Conocimientos sólidos de ciencias de la ingeniería e ingeniería aplicada que le permitan alcanzar el perfil profesional del Ingeniero Eléctrico Electrónico.
- Conocimientos fundamentales de la teoría de telecomunicaciones, además de conocer algunas de las tecnologías aplicadas en este campo.
- Conocimientos de electrónica analógica, digital y de potencia.
- Uso de los equipos de cómputo como una herramienta para su desenvolvimiento profesional.
- Bases teóricas para poder realizar estudios de especialización o postgrado.
- Un idioma extranjero.
- Técnicas y bases que les permita la interacción dentro de un núcleo social.

Aptitudes y habilidades

Los alumnos de Ingeniería Eléctrica Electrónica deben adquirir las aptitudes y habilidades necesarias para ejercer su función y fomentar el desarrollo en la sociedad.

Por lo anterior, los egresados de la Licenciatura de Ingeniería Eléctrica Electrónica, según su área de preespecialización:

- Aplicarán sus conocimientos para la administración, mantenimiento y mejora de sistemas de telecomunicaciones. Por lo que deberán contar con los conocimientos fundamentales de la teoría de telecomunicaciones además de conocer algunas de las tecnologías aplicadas en este campo; que les permitan analizar, diseñar, planear, organizar, producir, instalar, desarrollar, además de mantener en operación y administrar redes y sistemas de telecomunicaciones.

- Desarrollarán, diseñarán, instalarán, mantendrán, mejorarán sistemas electrónicos y eléctricos con un propósito definido. Es por ello indispensable que cuenten con conocimientos de electrónica

analógica, digital y de potencia que permiten al ingeniero eléctrico electrónico el diseño y la fabricación de equipo y material eléctrico-electrónico de alta complejidad técnica para aplicaciones industriales, de investigación o comerciales de la electrónica.

- Aplicarán sus conocimientos para la administración y mejora de sistemas de distribución y generación de energía eléctrica además de sus sistemas de protecciones. Por lo anterior el egresado de la Licenciatura comprenderá las técnicas que le permitirán al ingeniero eléctrico electrónico planear la generación, la transmisión y la distribución de la energía eléctrica, además del diseño, la construcción y la operación de sistemas eléctricos de potencia.

- Aplicarán sus conocimientos para el diseño, mantenimiento, administración y mejora de sistemas de instrumentación y control de procesos. Por ello deberán manejar las técnicas que permitirán al ingeniero eléctrico electrónico la utilización, el mantenimiento, diseño o la fabricación de componentes, dispositivos, equipos automáticos, sistemas de control o sistemas autómatas empleados durante la producción en serie.

Actitudes

Las actitudes profesionales que tendrá del egresado de la Licenciatura de Ingeniería Eléctrica Electrónica son:

- Creativo e innovador.
- Disciplinado y dinámico.
- Actitud emprendedora y de liderazgo seguido de iniciativa propia.
- Confianza en su preparación académica.
- Mente abierta orientada hacia la solución de problemas en la ingeniería.
- Honesto, responsable y critico con deseos de actualización, superación y competencia en su profesión.

En cuanto a las actitudes sociales, debe desarrollar:

- Conciencia de la problemática nacional, basada en el conocimiento de la realidad del país.
- Vocación de servicio profesional.
- Un cambio en la mentalidad frente a la competitividad internacional.
- Actitud humanista y de servicio a la sociedad.

Además de asumir en todos los casos una actitud comprometida y responsable, que se refleje en el entorno que actúe.

La Licenciatura de Ingeniería Eléctrica Electrónica proporcionará al egresado una base sólida sobre la que pueda apoyar su formación específica en áreas particulares, que le permita comunicarse e interactuar con otros profesionales de áreas afines, estas características le facilitarán su incorporación al mercado de trabajo.

Por lo tanto, el egresado de la Licenciatura de Ingeniería Eléctrica Electrónica deberá adquirir durante el transcurso de sus estudios los conocimientos de carácter formativo que persistan durante su vida profesional, y le den una base para especializarse o emprender estudios de postgrado y sobre todo para que se pueda mantener actualizado en los constantes avances en las técnicas y tecnologías de la Ingeniería Eléctrica Electrónica.

1.3.3 Perfil profesional

El ingeniero eléctrico electrónico es el profesional que posee los conocimientos de las ciencias exactas, matemáticas y de la ingeniería para desarrollar su actividad profesional en aspectos tales como el control automático y la instrumentación de procesos, las telecomunicaciones, desarrollo y mantenimiento de sistemas electrónicos analógicos y digitales, las instalaciones de electricidad industrial y de potencia, las líneas de transmisión y el mantenimiento y diseño de dispositivos electrónicos para la industria. Esta formación le permite participar con éxito en las distintas ramas que integran a la Ingeniería Eléctrica Electrónica, así como adaptarse a los cambios de las tecnologías en estas áreas y en su caso generarlos respondiendo así a las necesidades que se presentan en las ramas productivas y de servicios del país, para lograr el bienestar de la sociedad a la que se debe.

El ingeniero eléctrico electrónico es requerido tanto por el sector público, como por el sector privado, en los campos de docencia, investigación, asesoría, diseño y control de sistemas productivos de bienes y servicios.

1.4 Descripción de la estructura y organización curricular del nuevo plan de estudios.

1.4.1 Estructura del plan de estudios

De acuerdo con el Reglamento General para la Presentación, Aprobación y Modificación de Planes de Estudio, en el capítulo II “De la presentación

de los planes de estudio", en su artículo 4, inciso d, norma que un proyecto de creación de un plan de estudios constará, entre otras cosas, de la estructura curricular; y en el artículo 11, del mismo reglamento se menciona que la estructura curricular debe incluir las áreas académicas, asignaturas, módulos y demás elementos curriculares, definidos por sus objetivos generales y sus unidades temáticas, así como las relaciones que guardan entre sí, a fin de precisar su ordenación y ubicación en los periodos previstos para acreditar el plan de estudios.

La estructura curricular del plan de estudios de la Licenciatura de Ingeniería Eléctrica Electrónica de la FES ARAGÓN comprenderá, de acuerdo con la clasificación adoptada, siete tipos de áreas de conocimiento:

I. Físico Matemáticas
II. Socio-humanísticas
III. Electrónica
IV. Electricidad
V. Comunicaciones
VI. Control
VII. Metodológicas

Estas áreas agrupan los conocimientos que le permitirán al egresado contar con la preparación requerida por el mercado laboral, además de permitirle permanecer actualizado de acuerdo los cambios tecnológicos y las nuevas necesidades de la sociedad.

Se decidió la agrupación de las asignaturas por áreas de conocimiento debido a la afinidad de algunos grupos de asignaturas, lo que a su vez permite un mejor manejo en los trabajos colegiados. La integración de las áreas de conocimiento se planteó bajo los siguientes parámetros:

Área Físico-Matemáticas
Está integrada por las asignaturas de física y matemáticas que tienen por objetivo proporcionar al estudiante fundamentos científicos mismos que sustentarán su formación académica, cursándose durante los primeros tres semestres de la Licenciatura.
Con este grupo de asignaturas el alumno adquirirá bases sólidas, con enfoques adecuados y actualizados, para el correcto desarrollo de las ciencias de ingeniería, y de aspectos específicos de la ingeniería eléctrica electrónica. Comprende 12 asignaturas, de las cuales 11 se cursan durante los primeros cuatro semestres y una es de carácter optativo, representan el 27.68% de los contenidos del plan propuesto, de acuerdo con su valor total de créditos.

Área socio humanística
La conforman asignaturas cuyo objetivo es contribuir al desarrollo integral del estudiante mediante conocimientos que le permitan comprender la problemática socioeconómica y necesidades del país.

El aspecto socio-humanístico está cubierto por un conjunto de asignaturas que ubican al alumno en su entorno social, asumiendo su papel de protagonista con amplio sentido de responsabilidad y competitividad. Comprende tres asignaturas obligatorias que representan el 5.65% del contenido en créditos del plan de estudios propuesto y tres asignaturas optativas.

Área electrónica
La conforman seis asignaturas obligatorias, tres asignaturas obligatorias de elección y tres asignaturas optativas; que canalizan al estudiante hacia un campo en particular de ejercicio profesional y comprenden aspectos relacionados con la electrónica en el nivel de Ciencias de la Ingeniería e Ingeniería Aplicada.

Esta es un área disciplinaria fundamental para la Licenciatura, ya da soporte al módulo de preespecialización en Electrónica, lo que la hace de gran importancia en el quehacer profesional, por ello representa el 16.38% en créditos obligatorios pero pueden aumentar a 31.63% de los créditos de la Licenciatura, si se toman las asignaturas obligatorias de elección del módulo de preespecialización en Electrónica, así como la asignatura optativa del área.

Área de electricidad
La conforman cuatro asignaturas obligatorias, tres asignaturas obligatorias de elección y cinco asignaturas optativas; canalizan al estudiante hacia un campo en particular de ejercicio profesional las cuales comprenden aspectos relacionados con la electricidad al nivel de Ciencias de la Ingeniería e Ingeniería Aplicada.

Esta es un área disciplinaria de conocimiento para la Licenciatura y da soporte al módulo o de preespecialización en Energía Eléctrica Esta área profesional es de vital importancia para el desarrollo del país, por ello la preparación del alumno en esta área debe ser muy sólida; representa el 10.73% en créditos obligatorios pero pueden aumentar al 25.42% de los créditos de la Licenciatura si se toman las asignaturas obligatorias de elección y optativas de esta área.

Área de comunicaciones
La conforman tres asignaturas obligatorias, tres asignaturas obligatorias de elección y seis asignaturas optativas, que canalizan al estudiante hacia un campo en particular de ejercicio profesional que comprende aspectos relacionados con las comunicaciones en el nivel de Ciencias de la Ingeniería e Ingeniería Aplicada.
Esta es un área disciplinaria para la Licenciatura y da soporte al módulo de especialidad en Comunicaciones, representa el 7.91% en créditos obligatorios pero pueden aumentar al 22.03% de los créditos de la Licenciatura si se toman las asignaturas obligatorias de elección y optativas de esta área.

Área de control
Esta área está conformada por las asignaturas que buscan desarrollar en el alumno los conocimientos relacionados con los aspectos de control de procesos y automatización industrial. Esta área cuenta con asignaturas en niveles de Ciencias de la Ingeniería e Ingeniería Aplicada.

El área de Control cuenta con cuatro asignaturas obligatorias y al ser un área que concluye con el módulo de preespecialización se pueden agregar tres asignaturas obligatorias de elección y tres asignaturas optativas del área, por lo anterior su contenido pude variar del 10.17% al 24.29% de los créditos.

Área metodológica
Son herramientas básicas y necesarias para la Licenciatura y la forman cuatro asignaturas de diversas temáticas, que representan el 7.91% del contenido de créditos, y en caso de que el alumno elija asignaturas optativas complementarias sólo en esta área, representaría el 14.69% del total de créditos del plan de estudios.

Créditos totales
Lo anterior da un total de 354 créditos, de los cuales 306 son obligatorios, al menos 24 créditos obligatorios de elección y 24 optativos como mínimo, lo que se traduce en 35 asignaturas obligatorias, tres obligatorias de elección y tres optativas.
Así, el plan de estudios proporciona al alumno una formación sólida para participar en el campo profesional, ya que le permite desarrollar su capacidad de síntesis para hacer frente a los problemas propios de la Ingeniería.

El plan de estudios, además contempla cuatro módulos de preespecialización, los cuales tienen la finalidad de dar una orientación profesional a los alumnos en los últimos semestres de acuerdo a las

tecnologías existentes en el mercado laboral. La selección de las asignaturas para los módulos de preespecialización se estructuran con el siguiente fin: las asignaturas obligatorias de elección tienden a dar el área de desempeño profesional, y las asignaturas optativas dan el enfoque de aplicación, lo cual permite tener un sin número de perfiles.

A continuación se mencionan los cuatro módulos de preespecialización que contempla el plan de estudios:

Módulo de Electrónica
Prepara al alumno para poder entender, diseñar y mantener circuitos electrónicos analógicos y digitales, los cuales pueden ser aplicados en distintas actividades cotidianas.

Módulo de Eléctrica
Prepara al alumno para poder entender, diseñar y mantener sistemas de distribución, generación, protección y utilización de energía eléctrica.

Módulo de Control
Prepara al alumno para poder entender, diseñar y mantener sistemas de control automático, automatización e instrumentación industrial, con tecnologías de hardware y software.

Módulo de Comunicaciones
Prepara al alumno para poder entender, diseñar y mantener sistemas de transmisión de información, con tecnologías de hardware y software.

Las asignaturas obligatorias de elección y las optativas se seleccionaran bajo el siguiente criterio:

Tres asignaturas obligatorias de elección del módulo de preespecialización (Electrónica, Control, Eléctrica y Comunicaciones) que el alumno escoge y tres asignaturas de cualquier otro módulo o del bloque de asignaturas optativas. Cabe mencionar que en caso de que un alumno desee cambiarse de módulo de preespecialización los créditos de las asignaturas cursadas del primer módulo seleccionado serán contabilizados como optativas.

Otra opción para cursar las asignaturas obligatorias de elección y optativas es mediante la movilidad estudiantil, es decir un alumno podrá cursar las asignaturas y créditos correspondientes a éstas en otras entidades académicas de la UNAM o en otras instituciones de educación superior (IES), siempre y cuando haya equivalencias académicas entre las asignaturas, se tenga la autorización del Coordinador de la Licenciatura, y haya un convenio para tal fin con la entidad académica o IES receptora.

Las asignaturas se encuentran ordenadas de tal forma que presentan un orden cronológico en el conocimiento, de tal forma que un alumno que toma sus asignaturas conforme a lo estipulado en el plan de estudios, podrá aprender los contenidos de la Licenciatura sin problemas de conocimientos precedentes.

1.4.2 Organización curricular de la carrera de Ingeniería Eléctrica Electrónica.

El orden cronológico del mapa curricular es el siguiente:

Primero se incluyen asignaturas del área Físico Matemáticas, en los semestres que van principalmente del 1º al 3er, que son las asignaturas que fundamentan los conocimientos científicos en física y matemáticas.

En un nivel intermedio en el mapa curricular se encuentran las asignaturas clasificadas como ciencias de la ingeniería, que fundamentan los conocimientos básicos de la ingeniería eléctrica electrónica. Éstas se encuentran principalmente entre el 4º y el 7º semestre, y comprenden especialmente las áreas de conocimiento Electrónica, Eléctrica, Control y Comunicaciones.

Al final del mapa curricular, en el 8º semestre se encuentran las asignaturas cuya clasificación corresponde a la de ingeniería aplicada. Estas asignaturas permiten hacer uso de los principios de la ingeniería para resolver problemas teóricos y prácticos en el ejercicio de la profesión, y comprenden principalmente las áreas de conocimiento Electrónica, Eléctrica, Control y Comunicaciones.

Las asignaturas de ciencias sociales y humanidades están distribuidas a lo largo de toda la Licenciatura, con la finalidad de mantener la visión socio-humanista del ingeniero.

Respecto a las asignaturas que son consideradas para ser cursadas bajo la modalidad de curso-laboratorio no podrán ser aprobadas si el alumno no acredita el laboratorio y la teoría.

1.5 Finalidad de la materia de microprocesadores y microcontroladores en el plan propuesto de la carrera de IEE.

La materia de microprocesadores y microcontroladores es una materia obligatoria del modulo de electrónica la cual tiene la finalidad de proporcionar el conocimiento teórico práctico necesario de estos

dispositivos permitiendo comprender su estructura y funcionamiento dentro de los sistemas digitales que hacen uso de ellos, proporcionando las bases científicas necesarias para diseñar sistemas con microprocesadores que resuelvan un problema de ingeniería.

1.5.1 Datos generales de la asignatura.

NOMBRE DE LA ASIGNATURA:			MICROPROCESADORES Y MICROCONTROLADORES (L)	
PLAN 2007				
Clave:	**Créditos:**	10	**Carácter:** Obligatoria	**Semestre:** Séptimo
Duración del Curso	Semanas:	16	**Área de Conocimiento:** Electrónica	
	Horas:	96	**Fecha de Aprobación**	
Horas/Semana	Teoría:	4.0	**Consejo Técnico de la FES Aragón:**	
	Práctica:	2.0	**Consejo Universitario:**	
MODALIDAD: Curso - Laboratorio				
SERIACIÓN INDICATIVA PRECEDENTE:			Diseño lógico, Diseño de Sistemas Digitales (L)	
SERIACIÓN INDICATIVA SUBSECUENTE:			Circuitos Digitales (Mod. Electrónica) y Diseño de Sistemas con Microprocesadores (Mod. Electrónica).	

1.5.2 Objetivo del curso:

Analizar y comprender los conceptos y técnicas básicas de los microprocesadores y microcontroladores, así como su forma de programación para aplicaras en la solución de problemas de ingeniería.

1.5.3 Temas.

No.	Nombre	HORAS Teoría	HORAS Práctica
I	INTRODUCCIÓN	4.0	0.0
II	ARQUITECTURA DE LOS MICROPROCESADORES Y MICROCONTROLADORES	10.0	4.0
III	INTRODUCCIÓN A LA PROGRAMACIÓN DE LOS MICROPROCESADORES	8.0	4.0
IV	PROGRAMACIÓN CON LENGUAJE ENSAMBLADOR	20.0	10.0
V	COMUNICACIÓN CON OTROS DISPOSITIVOS	10.0	6.0
VI	INTERRUPCIONES Y RESETS	6.0	4.0
VII	CIRCUITOS DE SOPORTE	6.0	4.0
	Total de horas	64.0	32.0
	Total :	96.0	

1.5.4 Objetivos y contenidos de los temas.

TEMA I INTRODUCCIÓN

Objetivo: Conocer que es un microprocesador, que es un microcontrolador y cuales son sus aplicaciones.

Contenido:

I.1 Introducción a los microprocesadores y microcontroladores.
- I.1.1 Conceptos de microprocesadores y microcontroladores.
- I.1.2 Diferencias entre los microprocesadores y microcontroladores.
- I.1.3 Tipos de microprocesador según su velocidad y ancho de palabra.
- I.1.4 Aplicaciones de los microprocesadores y microcontroladores.

TEMA II ARQUITECTURA DE LOS MICROPROCESADORES Y MICROCONTROLADORES

Objetivo: Analizar la arquitectura de un microprocesador y un microcontrolador.

Contenido:

II.1 Arquitectura del microprocesador.
- II.1.1 A través de diagrama a bloques.
- II.1.2 Arquitectura externa del microprocesador (terminales).

II.2 Conexión del microprocesador con dispositivos de:
- II.2.1 Memoria.
- II.2.2 Periféricos.
  - II.2.2.1 Para un sistema mínimo.
  - II.2.2.2 Interfase para programación.
  - II.2.2.3 De comunicación serial.

II.3 Arquitectura del microcontrolador.
- II.3.1 Arquitectura interna del microcontrolador (vohn neuman, harvard), diagrama a bloques.
- II.3.2 Arquitectura externa del microcontrolador (terminales).

II.4 Conexión del microcontrolador con dispositivos:
- II.4.1 Periféricos.
  - II.4.1.1 Para un sistema mínimo.
  - II.4.1.2 Interfase para programación.
  - II.4.1.3 De comunicación serial.
- II.4.2 Para expansión de memoria.

TEMA III INTRODUCCIÓN A LA PROGRAMACIÓN DE LOS MICROPROCESADORES"

Objetivo: Familiarizarse con la programación de los microprocesadores.

Contenido:

III.1 Lenguaje de Máquinas y Ensambladores.
- III.1.1 Concepto de lenguaje máquina.
- III.1.2 Concepto de lenguaje ensamblador.

III.1.3 Programas ensambladores.

III.2 Transferencia de información (entrada, salida y almacenamiento)y conceptos básicos.

III.3 Conjunto de instrucciones de un microprocesador o microcontrolador.

TEMA IV PROGRAMACIÓN CON LENGUAJE ENSAMBLADOR

Objetivo: Programar el microprocesador o microcontrolador usando su conjunto de instrucciones para el desarrollo de programas de aplicación.

Contenido:

IV.1 Herramientas de diseño y documentación.

IV.2 Direccionamiento de Memorias y E/S.
- IV.2.1 Registros.
- IV.2.3 Modos de direccionamiento.
- IV.2.4 Control de dispositivos de entrada/salida.

IV.3 Operaciones con registros.
- IV.3.1 Operaciones aritméticas.
- IV.3.2 Operaciones lógicas.

IV.4 Control de flujo de programa.
- IV.4.1 Salto incondicionado.
- IV.4.2 Salto condicionado.
- IV.4.3 Subrutinas.
- IV.4.4 Banderas.

IV.5 Conteo y Lazos de tiempo.
- IV.5.1 Base de tiempo.
- IV.5.2 Contadores.
- IV.5.3 Implementación de subrutinas de tiempo.

IV.6 Conceptos Avanzados.
- IV.6.1 Configuración del convertidor A/D, D/A.
  - IV.6.1.1 Modos de operación.
  - IV.6.1.2 Aplicaciones.
- IV.6.2 Otros dispositivos.

TEMA V COMUNICACIÓN CON OTROS DISPOSITIVOS

Objetivo: Aprender las técnicas de acceso al medio a ambiente a través de las entradas y salidas.

Contenido:

V.1 Conceptos básicos de entrada/salida.
- V.1.1 Uso de líneas programadas Entrada/Salida para el control de dispositivos.
- V.1.2 Interfases de comunicación.
- V.1.3 Protocolos.

V.2 Puertos paralelos de entrada/salida.
V.2.1 Programación de puertos paralelos de entrada /salida.

V.3 Puertos serie de entrada/salida.
V.3.1 Programación de puertos serie de entrada/salida asíncrona.
V.3.2 Programación de puertos serie de entrada/salida síncrona

TEMA VI INTERRUPCIONES Y RESETS

Objetivo: Aprender las diferencias entre interrupciones y reset, así como la programación de estos y sus aplicaciones.

Contenido:

VI.1 Conceptos fundamentales de las interrupciones.
VI.1.1 Concepto de interrupción.
VI.1.2 Interrupciones enmascaradas y no enmascaradas.
VI.1.3 Prioridad de interrupciones.
VI.1.4 Servicio a las interrupciones.
VI.1.5 Vectores de interrupción.
VI.1.6 Programación de interrupciones.

VI.2 Resets.
VI.2.1 Excepciones y resets.
VI.2.2 Vectores de reset.

TEMA VII CIRCUITOS DE SOPORTE

Objetivo: Aprender los mecanismos de expansión de memoria y puertos para los microprocesadores y microcontroladores.

Contenido:

VII.1 Configuración y expansión de memoria externa.
VII.1.1 Asignación de espacios de memoria.
VII.1.2 Diseño de decodificadores de dirección.

VII.2 Extensión de puertos.
VII.2.1 Paralelo.
VII.2.2 Serial.

1.6 Infraestructura relacionada con la enseñanza de la materia de microprocesadores en la actual carrera de Ingeniería mecánica eléctrica en la FES Aragón.

La licenciatura de Ingeniería Mecánica Eléctrica cuenta para la enseñanza de la asignatura de microprocesadores con:

- Laboratorio de ingeniería L3.
- Grupos de teoría: 2 con capacidad para 60 alumnos cada uno.

- Grupos de laboratorio: 10 con capacidad para 10 alumnos.
- Profesores de laboratorio: 2 (cada uno atiende 5 grupos de laboratorio)

Actualmente los laboratorios de Ingeniería ostentan la certificación ISO 9001:2000; donde se contempla un programa de mejora continua en los equipos que cuenta. Esta certificación para los procesos de docencia, ha mejorado sustancialmente las condiciones académicas de las asignaturas teórico-prácticas curriculares y extracurriculares.

En el laboratorio de eléctrica-electrónica L 3 se imparten los siguientes laboratorios:

- Laboratorio De Medición E Instrumentación
- Laboratorio De Electrónica
- Laboratorio De Control
- Laboratorio De Comunicaciones
- Laboratorio De Sistemas Eléctricos De Potencia
- Laboratorio De Electricidad y Magnetismo

Como podemos ver no hay ningún laboratorio que lleve el nombre de microprocesadores, sin embargo es una materia del área de conocimiento de la electrónica por lo que este se imparte en el laboratorio de electrónica.

El laboratorio de electrónica cuenta (inventariado hasta el mes de Diciembre de 2006) para la impartición del los laboratorios (entre ellos el de microprocesadores) con el siguiente equipo:

Nombre Del Laboratorio: Laboratorio De Electrónica		
Nº	**Equipo Principal Del Laboratorio**	**Cantidad**
1	**Osciloscopio**	**25**
2	**Multímetro**	**36**
3	**Programador Universal**	**3**
4	**Computadoras**	**19**
5	**Fuente de voltaje**	**17**
6	**Generador de Funciones**	**27**
7	**Borrador de Memorias**	**2**
8	**Interfases Osciloscopio / PC**	**4**
9	**Analizador de Estado Lógico**	**2**

El equipo de los laboratorios ha sido renovado parcialmente para asegurar un adecuado desempeño teórico-práctico.

1.7 Pros y contras de las prácticas del laboratorio de microprocesadores.

Es importante hacer notar que **el laboratorio de microprocesadores no tiene un manual de prácticas establecido** por lo que las prácticas realizadas por los alumnos son planteadas por los profesores del laboratorio, lo cual representa una enorme desventaja ya que no es posible estandarizar la enseñanza del laboratorio, ya que cada profesor plantea diferentes prácticas con diferentes dispositivos.

Para evaluar los pros y contras de las prácticas actuales del laboratorio de microprocesadores se realizo una encuesta entre los profesores que imparten el laboratorio y alumnos que están cursando la materia con su respectivo laboratorio. Esta investigación permitió evaluar el grado de aprendizaje de los alumnos así como la eficiencia que en la enseñanza de la materia de microprocesadores y en base a estos resultados se plantean un conjunto de siete prácticas con que permitirán mejorar la enseñanza del laboratorio.

Se realizaron 2 tipos de encuesta: para los alumnos y para los profesores.

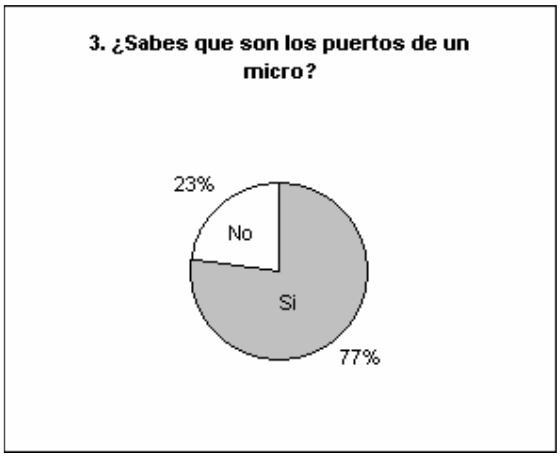
A continuación se presenta el formato de la encuesta realizada a los alumnos.

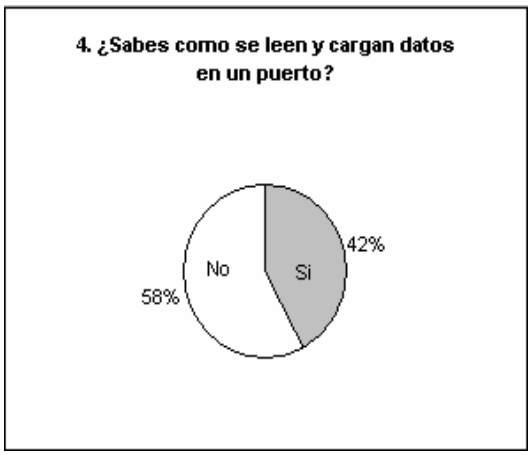
Cuestionario (alumnos)

1. ¿Sabes que es un microprocesador, un microcontrolador y sus diferencias?
   Si ( ) No ( )
2. ¿Sabes cuales son las terminales básicas con las que cuenta todo micro?
   Si ( ) No ( )
3. ¿Sabes que son los puertos de un micro?
   Si ( ) No ( )
4. ¿Sabes como se leen y cargan datos en un puerto?
   Si ( ) No ( )
5. ¿Sabes que es el lenguaje ensamblador?
   Si ( ) No ( )
6. ¿Sabes que es un programa fuente y como se escribe?
   Si ( ) No ( )
7. ¿Sabes que es un mnemónico y para que se utiliza?
   Si ( ) No ( )
8. ¿Sabes como se implementa un sistema mínimo con microprocesador?
   Si ( ) No ( )

9. ¿Sabes que es un salto condicional y uno incondicional y para que sirven?
   Si ( ) No ( )
10. ¿Sabes como se estructura una subrutina de tiempo?
    Si ( ) No ( )
11. ¿Sabes que es una interrupción?
    Si ( ) No ( )
12. ¿Consideras que las prácticas actuales te ayudan en tu desarrollo como ingeniero? Si ( ) No ( )
13. ¿Cuáles son los principales problemas que notaste en la realización de las prácticas laboratorio de microprocesadores?
14. ¿Cuáles fueron los principales beneficios que obtuviste con la realización de las prácticas del laboratorio de microprocesadores?
15. ¿Crees que se requieren cambios para mejorar en la impartición del laboratorio de microprocesadores?

La encuesta realizada a los alumnos arrojo los siguientes resultados:



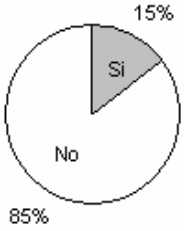


5. ¿Sabes que es el lenguaje ensamblador?

6. ¿Sabes que es un programa fuente y como se escribe?

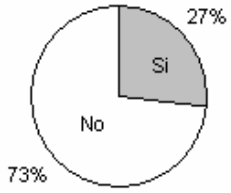
7. ¿Sabes que es un mnemónico y para que se utiliza?

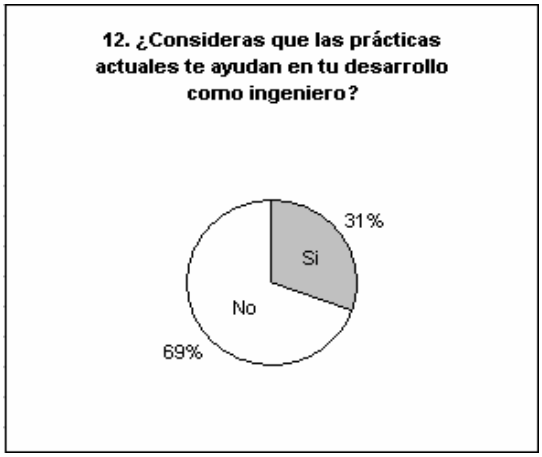
8. ¿Sabes como se implementa un sistema mínimo con microprocesador?



9. ¿Sabes que es un salto condicional y uno incondicional y para que sirven?

10. ¿Sabes como se estructura una subrutina de tiempo?





Para **la pregunta 13** (¿Cuáles son los principales problemas que notaste en la realización de las prácticas laboratorio de microprocesadores?) las respuestas en su mayoría son en el sentido de que no existe un vinculo entre la teoría vista en clase con la práctica que se lleva a cabo en el laboratorio ya que cada profesor enseña lo que considera es lo mas importante y con el dispositivo que el considere y consideran que esto se debe a que en el laboratorio no existe un manual de prácticas para la materia de microprocesadores.

Para **la pregunta 14** (¿Cuáles fueron los principales beneficios que obtuviste con la realización de las prácticas del laboratorio de microprocesadores?) las respuestas muestran que los principales beneficios fueron:

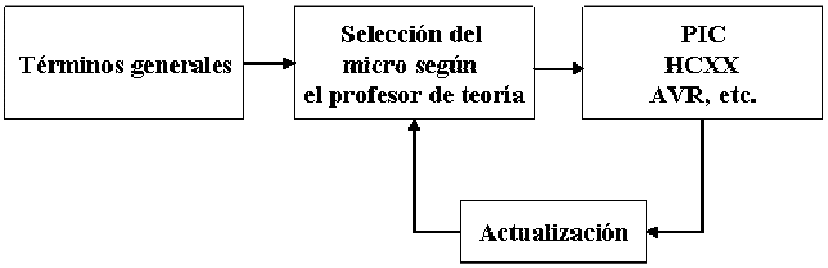
- Que el alumno conoció físicamente un sistema con microprocesador.
- Que el alumno aprendió un poco de la forma de programación de los micros.
- Algunos mencionan que no obtuvieron beneficio alguno y otros que aprendieron más por su cuenta.

En **la pregunta 15** (¿Crees que se requieren cambios para mejorar en la impartición del laboratorio de microprocesadores?) el 100% de los alumnos encuestados considera que se deben de realizar cambios en la impartición del laboratorio de microprocesadores. La mayoría opino que el principal cambio que se debe de realizar es en el sentido de tener un manual de prácticas en el cual se puedan guiar y el cual permita establecer una paridad con la enseñanza de la teoría con la práctica ya que mientras no se expongan primero los conceptos fundamentales de forma teórica, estos no podrán ser entendidos en la práctica. Consideran también que este manual debe de ser didáctico para favorecer el aprendizaje.

A continuación se muestra el formato de la encuesta realizada a los profesores del laboratorio de microprocesadores y las respuestas que proporcionaron.

Cuestionario (Profesor 1)

1. ¿Aplica las prácticas existentes para el laboratorio de microprocesadores (marque con una cruz)?
   - Si ( ) No (x)
2. En caso de que no las aplique explique la razón
   - No existen, hay algunas obsoletas
3. ¿Cuántas prácticas realiza durante un semestre?
   - 5 o 6
4. ¿Cuáles son los temas de las prácticas que aplica?
   - Conceptos y operaciones con el microcontrolador.
   - Configuración interna de un microprocesador.
   - Selección del microcontrolador y sus características (manual).
   - Configuración entrada/salida de puertos.
   - Rutina de retardo.
   - Modos de direccionamiento.
5. ¿Cuáles son los principales problemas al aplicar dichas prácticas?
   - El formato de las prácticas depende del profesor ya que no hay uno establecido.
   - No concuerda la teoría con la práctica.
   - No existe un micro específico.
   - El grabador se tiene que implementar y consume tiempo.
   - El material de apoyo tiene problemas (computadoras con virus).
6. ¿Cuáles son los beneficios de la realización de estas prácticas?
   - El complemento al conocimiento del tema.
7. ¿Qué método utiliza para aplicar las prácticas?
   Es un método constructivista, Se proporciona el formato y el alumno desarrolla el trabajo en casa, programa el circuito y el día de la práctica se verifica y se anotan conclusiones en la bitácora.
8. ¿Considera usted que el manual de prácticas actual necesita ser actualizado?
   Si. Y a mi parecer el esquema de estos manuales debería seguir la siguiente estructura.



9. ¿Qué temas considera usted que deben tocar las prácticas actualizadas?
   - Entrada/salida
   - Métodos de direccionamiento.
   - Operaciones aritméticas.
   - Subrutinas.
   - Interfaces (USB, I^2C)
10. ¿Cuáles son las tendencias actuales en la enseñanza del laboratorio de microprocesadores?
    - Los sistemas embebidos (aplicaciones particulares).
    - La comunicación (RF, infrarrojo, etc.), TCP/IP, etc.
    - Linux embebido (micros de 32 bits).

    Nota: Corregir el plan de estudios en comité académico para que exista seriación:
    - Diseño lógico.
    - Diseño de sistemas digitales.
    - Microprocesadores y microcontroladores.
    - Diseño de sistemas con microprocesadores.

Cuestionario (Profesor 2)

1. ¿Aplica las prácticas existentes para el laboratorio de microprocesadores (marque con una cruz)?
   Si ( ) No (x)
2. En caso de que no las aplique explique la razón
   - No existen un manual de prácticas
3. ¿Cuántas prácticas realiza durante un semestre?
   - 6 prácticas
4. ¿Cuáles son los temas de las prácticas que aplica?
   - Conceptos fundamentales.
   - Programación con lenguaje ensamblador.
   - Operaciones aritméticas con el microprocesador
   - Control de flujo de programa.
   - Interrupciones.
   - Modos de direccionamiento.
5. ¿Cuáles son los principales problemas al aplicar dichas prácticas?
   - El formato de las prácticas depende del profesor ya que no hay uno establecido.
6. ¿Cuáles son los beneficios de la realización de estas prácticas?
   - Que el alumno aprenda por si mismo los conceptos de estos dispositivos..
7. ¿Qué método utiliza para aplicar las prácticas?

Planteo el problema para que el alumno lo resuelva y tiene una semana para alambrarlo y hacerlo funcionar.

8. ¿Considera usted que el manual de prácticas actual necesita ser actualizado?
   Si.
9. ¿Qué temas considera usted que deben tocar las prácticas actualizadas?
   - Conceptos generales.
   - Utilización de los puertos de E/S
   - Programación con lenguaje ensamblador.
   - Control de flujo de programa.
   - Subrutinas de tiempo
   - Interrupciones
10. ¿Cuáles son las tendencias actuales en la enseñanza del laboratorio de microprocesadores?
    - El microprocesador como elemento de control.
    - El microprocesador y las comunicaciones.
    - Etc.

1.8 Conclusiones generales del estudio realizado.

El estudio realizado indica que es necesario mejorar en la calidad de la enseñanza del laboratorio de microprocesadores ya que los alumnos alcanzan una comprensión parcial de algunos temas y de algunos otros no tienen idea de lo que se trata. Una de las formas es mediante la elaboración de una manual de prácticas didáctico en el cual se exponga la teoría básica del tema que se abordara en la realización de la práctica ya que si el manual contiene esta información preliminar permite que el alumno tenga una mayor comprensión del tema y se interese en el maravilloso mundo de los microprocesadores. Ante esta situación en el presente trabajo se esta proponiendo un manual de prácticas el cual es apropiado para la enseñanza del laboratorio ya que no es necesario que un profesor explique el tema ya que el tema esta explicado en la introducción de las prácticas permitiendo llevar una coherencia entre la teoría de estos dispositivos y la práctica. Además se incluyen un apéndice que complementa la información teórica necesaria para la realización de las prácticas.

Capitulo 2

GENERALIDADES DE LOS MICROPROCESADORES Y LOS MICROCONTROLADORES

2.1 Conceptos básicos.

*El **microprocesador** es una unidad central de proceso contenida totalmente en un circuito integrado*. El microprocesador, o simplemente el micro, es el cerebro del ordenador. Es un chip, un tipo de componente electrónico en cuyo interior existen miles (o millones) de elementos llamados transistores, cuya combinación permite ejecutar el programa para realizar el trabajo que tenga encomendado el chip.

Las patitas de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Un sistema digital basado en microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que se destine ya que se pueden aumentar o disminuir el número de periféricos según sea necesario tal como se ve en la Figura A.

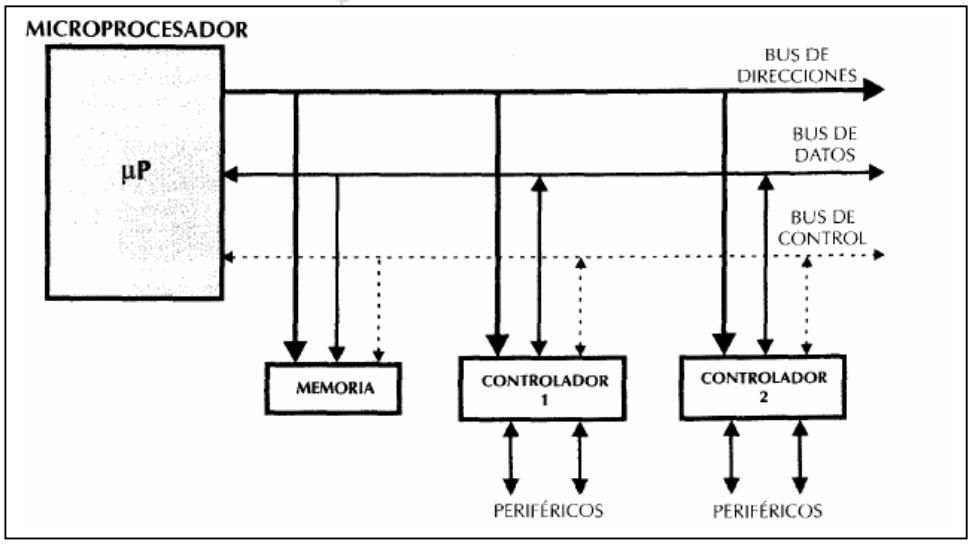


Figura A. Sistema digital con microprocesador

La capacidad de operación de un microprocesador está basada en la cantidad de bits que puede manejar por vez. Es obvio que a mayor

cantidad de bits manejados mayor será la capacidad. La cantidad de bits varía entre 4 y 64 y su conjunto se llama **palabra**.

Se distinguen tres tipos de bus:

- **De control**: forman parte de él las líneas que seleccionan desde dónde y hacia dónde va dirigida la información, también las que marcan la secuencia de los pasos a seguir para dicha transferencia.
- **De datos**: por él, de forma bidireccional, fluyen los datos entre las distintas partes del ordenador.
- **De direcciones**: El bus de direcciones consta de un conjunto de líneas que permite seleccionar de qué posición de la memoria se quiere leer su contenido. También direcciona los puertos de E/S.

Además de los buses se puede observar la presencia de una memoria. Se necesitan de 2 tipos de memoria para el correcto funcionamiento de este tipo de sistemas que son la memoria RAM y la memoria ROM.
El programa se almacena en la memoria ROM. Esta memoria puede constituirse de diferentes formas: UVPROM, EEPROM u OTPROM, cualquiera que sea la que se utilice es una memoria no volátil desde la que se ejecutará el programa una vez alimentado el sistema. Para poder trabajar correctamente, nuestro microprocesador necesita, a menudo, almacenar datos temporales en alguna parte, y aquí es donde interviene la memoria RAM, que no necesita ser de grandes dimensiones.

El último elemento son las interfaces o controladores que le permiten al micro comunicarse con elementos del mundo exterior como motores, pantallas de cristal líquido, etc. Estos elementos son generalmente, los más importantes en una aplicación susceptible de utilizar un microcontroladores.

En la Figura B se muestra un diagrama que explica la transferencia de datos que ocurre en un sistema basado con microprocesador. Se puede observar que el microprocesador sigue las instrucciones de un programa almacenado en la memoria ROM, este programa hace referencia a datos que se encuentran almacenados en la memoria de datos (RAM) o puede provenir de alguno de los puertos de entrada/salida, estos datos son procesados y después son almacenados en alguna parte de la memoria de datos o bien pueden ser enviados a alguno de los puertos de entrada/salida.

*El **microcontrolador** es un sistema digital programable que contiene todos los componentes de un computador (una unidad central de proceso (CPU) memoria RAM, memoria ROM, puertos de entrada/salida, temporizadores, comparadores, etc.).* Se emplea para controlar el funcionamiento de una tarea determinada. En su memoria sólo reside un

programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada. Un microcontrolador es de sistema cerrado ya que todos los elementos del computador están dentro de el y solo salen las líneas que gobiernan los periféricos (Figura C).

Figura B. Transferencia de datos en un sistema con microprocesador

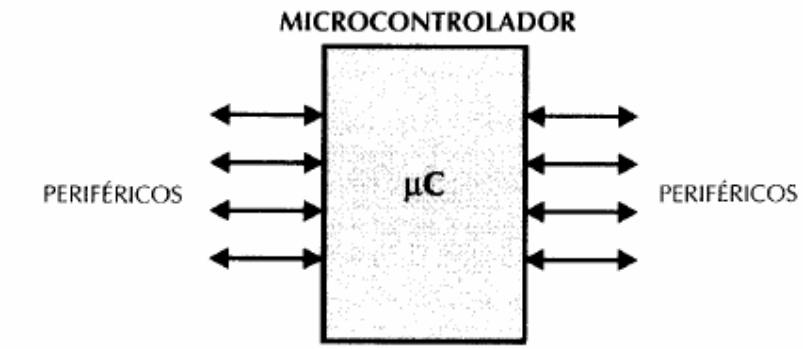


Figura C. Microcontrolador

Una representación a bloques que nos da una idea de los elementos básicos que conforman a un microcontrolador se muestra en la Figura D. Encontramos en él la unidad central (CPU) que es el elemento más importante del sistema. Se encarga de direccionar, recibir el código de operación de la instrucción en curso, su decodificación y la ejecución de la operación, que implica la búsqueda de operandos y almacenamiento de resultados.

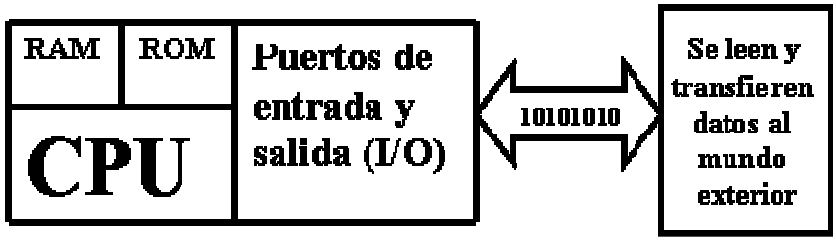


Figura D. Estructura de un microcontrolador

Los procesadores se pueden dividir, según su conjunto de instrucciones, en:

- **CISC** Set de instrucciones complejo (instrucciones sofisticadas y potentes. Más de 200 instrucciones.)
- **RISC** Set de instrucciones reducido (instrucciones simples y se ejecutan en un ciclo. Cerca de 30 instrucciones.)
- **SISC** Set de instrucciones específico (y reducido)

La CPU se ha simplificado con respecto a los microprocesadores clásicos. En contrapartida se le han añadido instrucciones de manejo de bits, muy útiles para las entradas/salidas. En ciertos circuitos, esta unidad central está dotada de un gran número de registros internos, que sirven de memoria RAM, por lo que puede parecer que ésta última está ausente de algunos esquemas.

A continuación podemos ver las memorias. Según su uso específico dentro del sistema, a las memorias las podemos llamar RAM para aquellas que manejan datos y variables, y ROM a las que almacenan el programa.

Existen 5 tipos de ROM:

1. ROM con máscara: se graba durante la fabricación del chip. Se fabrican más de 1000 unidades.
2. OTP (One Time Programming): permiten una única grabación. Se utilizan para series pequeñas de fabricación (prototipos). Tienen muy bajo costo.
3. EPROM: también llamadas UV-PROM. Se borran mediante luz ultravioleta. Son relativamente más caras que las OTP pero son más veloces en la lectura.
4. EEPROM: Borrables eléctricamente. Pueden grabarse sin retirarse del circuito. Son más lentas que las EPROM.
5. FLASH: Similar a las EEPROM pero de mayor densidad (más capacidad) y más veloces.

En lo referente a la memoria RAM, suele utilizarse una del tipo SRAM (RAM estática) de pequeño tamaño, por qué generalmente la unidad central posee suficientes registros para realizar operaciones intermedias. En algunos casos, esta memoria se completa con una EEPROM de datos, que memoriza de forma semi permanente datos del usuario que se manejan como constante en la ejecución del programa y que de vez en cuando (pasados meses o años) deben ser modificados.
Algo más delicado es hacer un esquema tipo para los circuitos de interfaz, ya que es un punto donde se distinguen los diferentes microcontroladores, en función de las aplicaciones que pretenden. No obstante se pueden encontrar los siguientes elementos básicos:

- Líneas de entrada/salida paralelo, en cantidad variable, según la finalidad y el tamaño del encapsulado (se plantea un problema de número máximo de pines debido al crecimiento del número de estas líneas).
- Uno o varios temporizadores internos cuyas posibilidades pueden ser muy variables pero que, generalmente, funcionan como contadores ascendentes y descendentes, generadores de impulsos programables, etc.
- Al menos una interfaz de entrada/salida serie asíncrona, más o menos evolucionada según los circuitos.

2.2 Arquitectura Von Neuhmann.

Un sistema digital programable basado en la arquitectura Von Neuhmann contiene una unidad central de proceso (CPU) conectada a una memoria única en donde se encuentran la memoria de programa y la memoria de datos tal como se muestra en la Figura E.

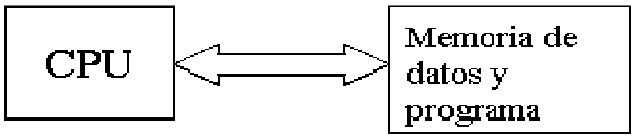


Figura E. Arquitectura Von Neuhmann

La comunicación entre la CPU y la memoria se lleva a cabo mediante buses. Estos buses pueden transmitir un cierto número de bits lo cual es una limitación ya que si se tiene un microprocesador de 8 bits y un bus de 8 bits que lo conecta con la memoria y si tiene que manejar datos e instrucciones de más de 8 bits (bytes) de longitud deberá realizar más de un acceso a la memoria. Por otro lado este bus único limita la velocidad de operación del microprocesador, ya que no se puede buscar de memoria una nueva instrucción, antes de que finalicen las transferencias de datos que pudieran resultar de la instrucción anterior.

2.3 Arquitectura Harvard.

En la arquitectura Harvard son independientes la memoria de instrucciones y la memoria de datos y cada una dispone de su propio sistema de buses para el acceso a ellas como se muestra en la Figura F.

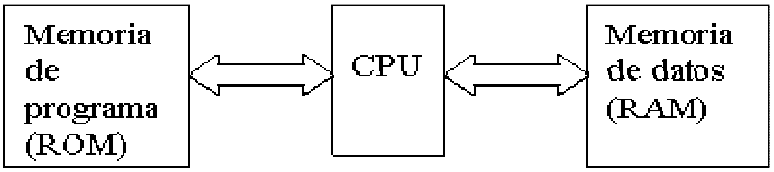


Figura F. Arquitectura Harvard

Ambos buses son totalmente independientes y pueden ser de distintos anchos. Ya que los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar.

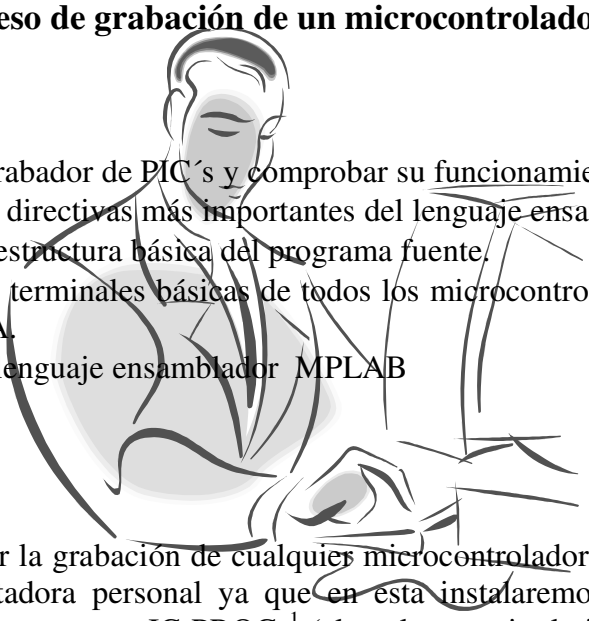
Los microcontroladores más modernos utilizan sus procesadores con arquitectura Harvard y son llamados “microcontroladores RISC”. Donde RISC significa Reduced Instrction Set Computer (Computadores de Juego de Instrucciones Reducido), que se identifica por poseer un repertorio de instrucciones máquina pequeño y simple, de forma que la mayor parte de las instrucciones se ejecuta en un ciclo de instrucción.
El microcontrolador PIC 16F84 posee arquitectura Harvard tipo RISC, con una memoria de datos de 8 bits, y una memoria de programa 14 bits.

Capitulo 3

PRÁCTICAS PROPUESTAS

3.1.- Práctica 1:

"El proceso de grabación de un microcontrolador PIC".



Objetivos:

- Armar un grabador de PIC´s y comprobar su funcionamiento.
- Conocer las directivas más importantes del lenguaje ensamblador.
- Conocer la estructura básica del programa fuente.
- Conocer las terminales básicas de todos los microcontroladores usando el PIC 16F84A.
- Conocer el lenguaje ensamblador MPLAB

Introducción.

Para realizar la grabación de cualquier microcontrolador PIC necesitamos una computadora personal ya que en esta instalaremos el software de grabación IC-PROG [1] (el cual transmite la información de la computadora al PIC), también se instalara el lenguaje ensamblador MPLAB[2] en el cual escribiremos los programas para que el PIC lleve a cabo las acciones solicitadas (convierte los mnemónicos y símbolos alfa numéricos del programa ensamblador en lenguaje maquina) y finalmente se utiliza un grabador que es el medio por el cual se transmite la información de la computadora al PIC.

Directivas del lenguaje ensamblador:

El lenguaje ensamblador (MPLAB) necesita información en forma de comandos insertados en el programa que le permiten ensamblar un programa de forma automática. No forman parte del set de instrucciones del microcontrolador por lo que se les conoce también como pseudoinstrucciones. Las más importantes son:

[1] El programa IC-PROG se encuentra de forma gratuita en la dirección www.ic-prog.com.
[2] El programa MPLAB puede obtenerse en forma gratuita de la dirección: http://www.microchip.com/10/Tools/mTools/MPLAB/index.htm

- ***__CONFIG***: Esta directiva indica la configuración elegida para el proceso de grabación (es muy importante hacer notar que se deben escribir 2 guiones bajos antes de la palabra CONFIG). Por ejemplo si se escribe:

  __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC.

  Esto indica que:

  - ***_CP_OFF:*** no hay protección de código
  - ***_WDT_OFF***: watch dog inhabilitado
  - ***_PWRTE_ON***: se habilita el reset mediante Power-up timer
  - ***_XT_OSC***: se esta utilizando un oscilador de cristal de cuarzo

- ***LIST P=16F84A***: Esta directiva le indica al lenguaje ensamblador el micro utilizado.
- ***INCLUDE <P16F84A.INC>*:** Indica el fichero donde se localizan las etiquetas que nombran a los diferentes registros y el valor que le corresponde a cada uno de ellos. Este fichero se localiza en el directorio principal del lenguaje ensamblador.
- ***ORG*:** esta directiva indica al lenguaje ensamblador la dirección de memoria de programa a partir de la cual debe ensamblar el programa. Por ejemplo:

  ```
  ORG   0x0;  los códigos maquina de las instrucciones que
              ; aparecen en las líneas siguientes deben
              ; ensamblarse a partir de la dirección 4.
  ```

- ***EQU*:** es una directiva de asignación. Usualmente van al principio del programa (antes de las instrucciones). Su sintaxis es: <etiqueta> EQU <valor>. Por ejemplo:

  ```
  Contador   EQU   d'55';  se le asigna a la etiqueta contador
                          ;el valor 55 decimal.
  ```

- ***END*:** esta directiva indica el fin del programa. Es la única directiva obligatoria. Las instrucciones que estén después de esta directiva serán ignoradas.
- ***CBLOCK*:** esta directiva permite asignar direcciones (generalmente de la memoria RAM de datos) a una lista de etiquetas. A cada etiqueta se la asigna un valor inmediatamente superior al anterior. La lista de etiquetas finaliza cuando se encuentra la directiva **ENDC.** Por ejemplo.

  ```
  CBLOCK  0x0C;  las etiquetas se posicionan a partir de esta
  ```

```
                    ;   posición de RAM
Primero   ;  La etiqueta Primero ocupa la posición 0x0C de RAM
Segundo   ;  La etiqueta Segundo ocupa la dirección 0x0D de RAM
Tercero   ;  La etiqueta Tercero ocupa la dirección 0x0E de RAM
Cuarto    ;  La etiqueta Cuarto ocupa la dirección 0x0F de RAM
ENDC  ;  Fin a la asignación de direcciones de RAM de las etiquetas
```

Observe que las direcciones son a partir de la posición de memoria RAM 0C, esto debido a que las 12 primeras posiciones de la memoria están asignadas a registros fijos propios del PIC.

- ***#DEFINE*:** esta directiva define un nombre para un texto el cual representara a dicho texto donde quiera que el nombre aparezca: Ejemplo

```
#DEFINE  LED     PORTB,0
         ------------
Bsf    LED  ; se activa la línea 0 del puerto B
            ; encendiendo el led
```

Programa fuente:

El programa fuente esta compuesto por una sucesión de líneas de programa. Cada línea de programa esta compuesta por 4 campos separados por uno o más espacios o tabulaciones. Estos campos son:
[Etiqueta] Comando [Operando(s)] [; Comentario]
La etiqueta es opcional y es el campo que empieza en la primera posición de la línea. No se pueden insertar espacios o tabulaciones antes de la etiqueta sino será considerado comando.
El comando es un nemónico del set de instrucciones del microcontrolador.
El operando esta asociado al comando, si no hay comando no hay operando, e inclusive algunos comandos no llevan operando.
El comentario es opcional para el compilador. El campo de comentario debe comenzar con un carácter punto y coma. No necesita tener espacios o tabulaciones separándolo del campo anterior, e incluso puede empezar en la primera posición de la línea. El compilador ignora todo el texto que contenga la línea después de un carácter punto y coma.

Terminales del PIC 16F84A:

Este microcontrolador es diseñado por la empresa Microchip (http://www.microchip.com). EL PIC16F84 está fabricado con tecnología

CMOS de altas prestaciones y encapsulado en plástico con 18 patillas, con la nomenclatura que se muestra en la Figura 1.1 junto con el símbolo que utilizaremos en estas prácticas.

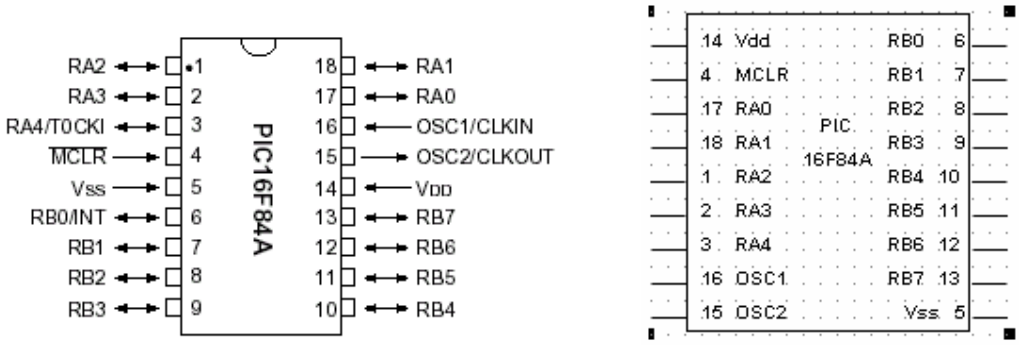


Figura 1.1. Terminales y símbolo del PIC 16F84

Su sistema de grabación de datos se realiza en serie. Para escribir un programa en la memoria se manda la información en serie usando sólo dos terminales, una para la señal de reloj y otra para los datos serie.

La misión de cada patilla comentada brevemente es:

V_{DD}: Tensión positiva de alimentación.
V_{SS}: Tensión conectada a tierra o negativa de alimentación.
OSC1/CLKIN: Entrada del circuito oscilador externo que proporciona la frecuencia de trabajo del microcontrolador.
OSC2/CLKOUT: Patilla auxiliar del circuito oscilador.
MCLR#: Patilla activa con nivel lógico bajo, lo que se representa con el símbolo #. Su activación origina la reinicialización o Reset del PIC. También se usa durante la grabación de la memoria para introducir por ella la tensión V_{PP}.
RA0-RA4: Son las 5 líneas de E/S digitales correspondientes a la Puerta A. La línea RA4 multiplexa otra función expresada por TOCKI. En ese caso sirve para recibir una frecuencia externa para alimentar al temporizador TMR0.
RB0-RB7: Son las 8 líneas de E/S digitales de la Puerta B. La línea RB0 multiplexa la función de servir como entrada a una petición externa de una interrupción.

Carga de datos en el PIC:

Para programar un PIC se deben conectar varias de sus terminales en una disposición determinada (Figura 1.2) que hace accesible su memoria para

recibir datos serie desde el puerto paralelo de la computadora. Para lograr esto necesitamos de una herramienta que es el grabador de PIC's que es el medio por el cual transferimos la información de la computadora al PIC. El puerto paralelo de la computadora, en donde se conecta habitualmente la impresora, tiene varias terminales de conexión, cuyo estado puede modificarse por medio de un programa adecuado como el IC-PROG en una rápida sucesión que se corresponda con la información serie necesaria para cargar el PIC. En una palabra que cada hilo de un puerto paralelo puede transformarse en un puerto serie y es en este puerto en donde conectamos nuestro grabador.

El proceso de grabación ocurre de la siguiente forma. Al colocar el PIC en el grabador, este se polariza con 5V entre sus terminales 14 (VDD) y 5 (Vss), inmediatamente se aplica un voltaje de 13V en su terminal 4 (VPP). Para la transferencia del primer dato la terminal 13 (RB7) recibe un potencial alto (5V) o bajo (0V) y espera a que la terminal 12 (RB6) en la cual entra la señal de reloj pase a un estado alto y en este momento se graba el dato en la memoria del PIC. Para los demás datos el proceso es el mismo y se repite hasta que se graba el último dato.

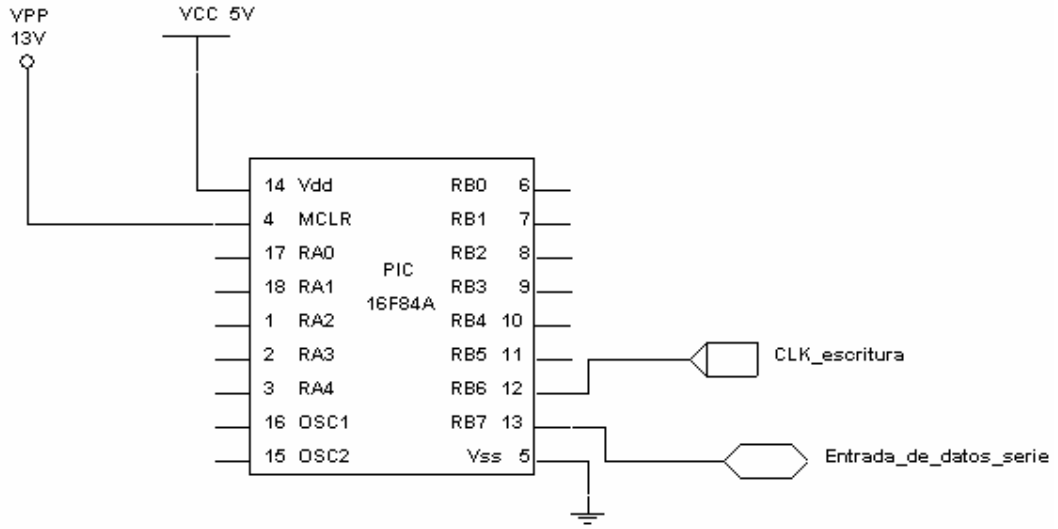


Figura 1.2. Conexión del PIC para su grabación

Material y equipo para la práctica.

1-PIC16F84A
1-Cristal de cuarzo de 4MHZ
2-Capacitores de 22pF
2-Dip switch

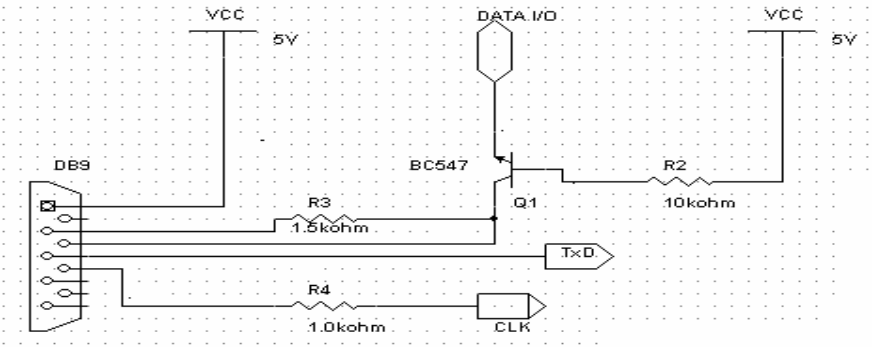
5-Diodos led
5-Resistencias de 330 Ω
7-Resistencias de 10 $k\Omega$
1-PC
1-Fuente de alimentación

Material para el grabador

C1- 22μF /16V	Q1, Q3-BC547
C2-100μF /16V	Q2-BC557
D1-D2,D5,D6- 1N4148	R1- 100$K\Omega$
D4-Zéner de 5V1 $\frac{1}{2}W$	R2-10$K\Omega$
D3- Zéner de 8V2 $\frac{1}{2}W$	R3-1.5$K\Omega$
U1-zócalo de 8 pines	R4-1$K\Omega$
U2-zócalo de 18 pines	
U3-zócalo de 28 pines	
DB9-conector DB9 hembra	

Trabajo de casa

1. Arme el grabador de PIC's TE-20 cuyo diagrama se muestra en la figura 1.3, el cual se utilizara en todas las prácticas posteriores (de preferencia soldelo).
2. Alambre el circuito de la Figura 1.14



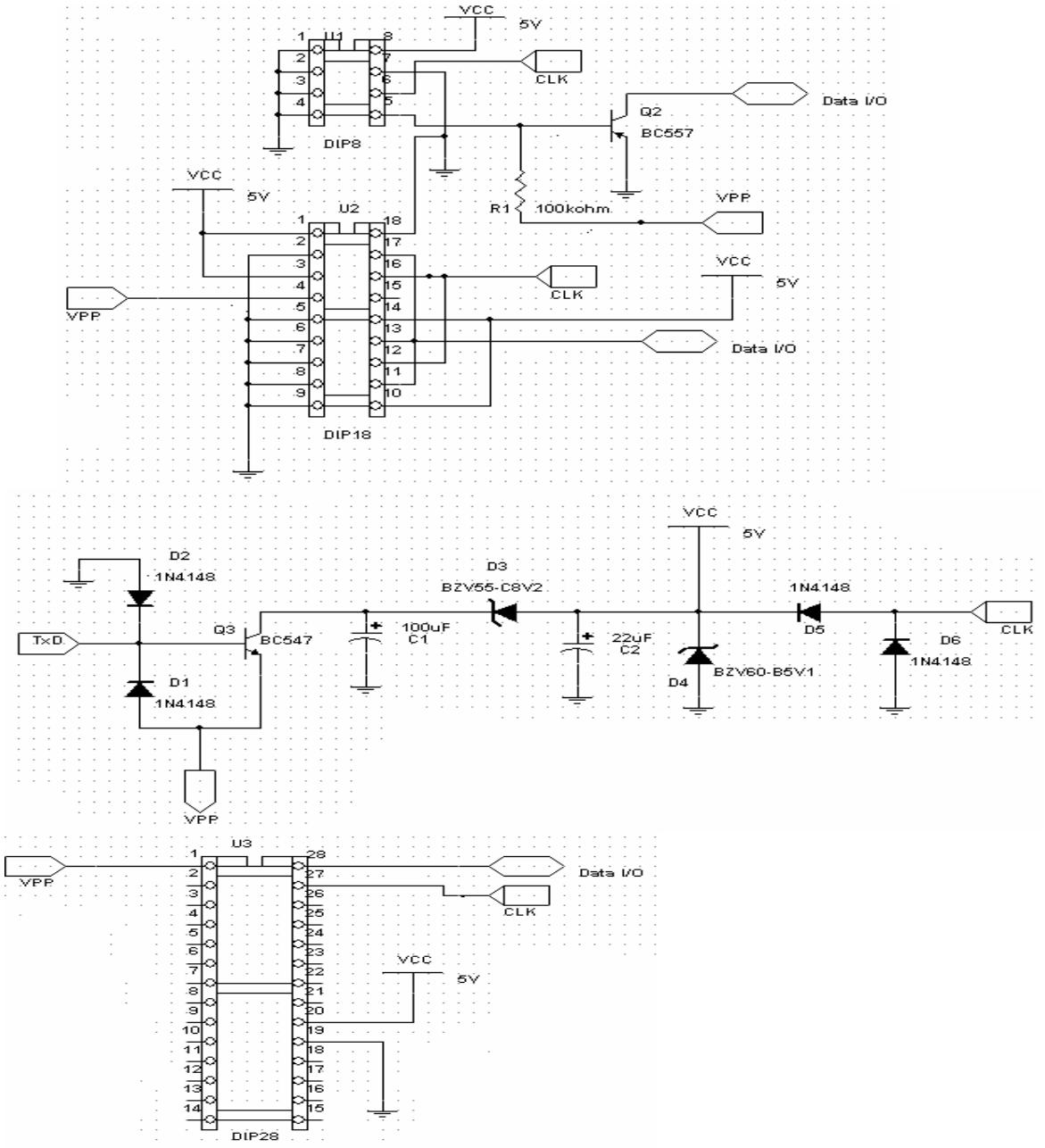


Figura 1.3. Grabador universal de PIC's TE-20

Desarrollo

1. Una vez instalado el MPLAB se debe configurar por primera vez. Primero se debe iniciar el programa MPLAB dando doble clic en el icono del escritorio. Una vez abierto vaya al menú File-New y se entrara en la pantalla de trabajo en donde se escribirán los programas fuente. Es conveniente crear una carpeta en el disco duro en donde se guardarán todos los programas hechos por ejemplo C: \PIC.

2. Se selecciona el tipo de microcontrolador activando el menú Configure-Select device y seleccionar PIC16F84A tal como se muestra en la Figura 1.4.

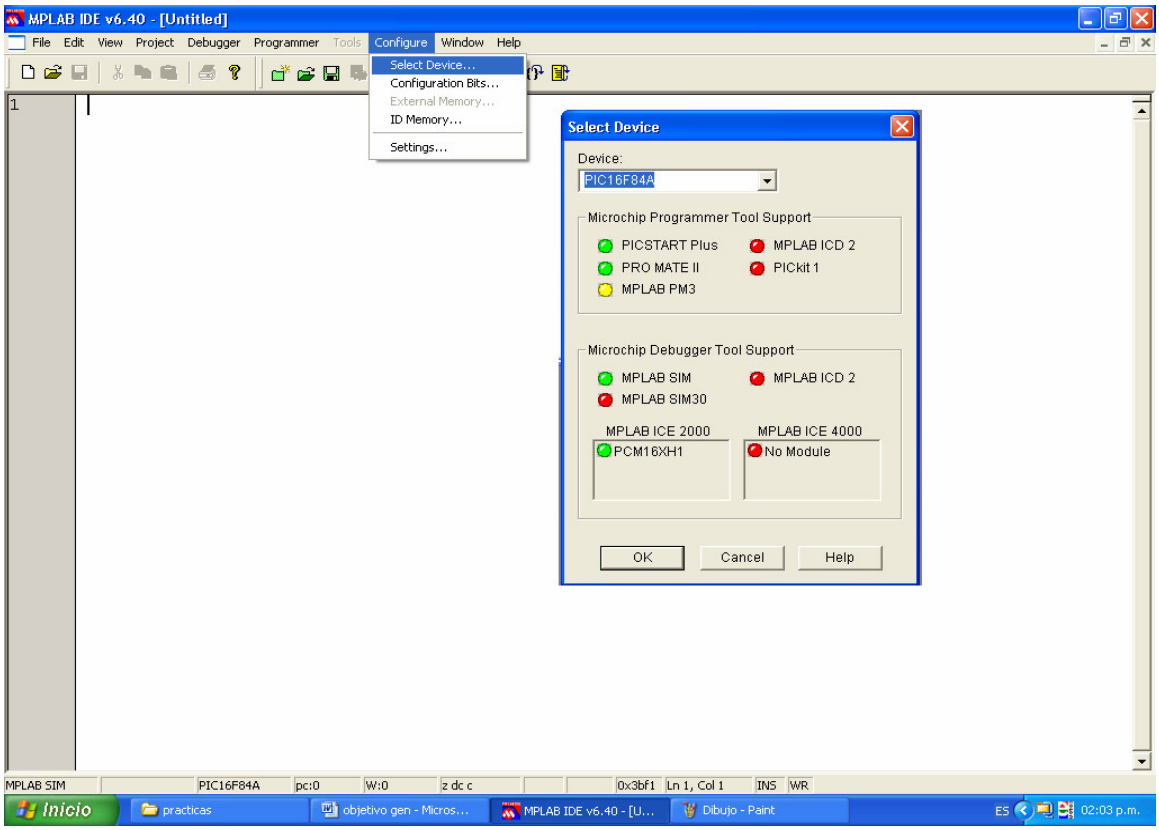


Figura 1.4. Selección del microcontrolador utilizado

3. Es conveniente activar el simulador activando el menú Debugger-select Tool-MPLAB SIM tal como se ve en la Figura 1.5.
4. Se debe de configurar la frecuencia con la que va a trabajar el PIC que para nuestro caso es de 4MHz. Para esto se activa el menú Debugger-Settings-Clock y se pone a 4MHz como se muestra en la Figura 1.6.
5. A partir de este momento ya se puede escribir el primer programa fuente activando el menú File-New.

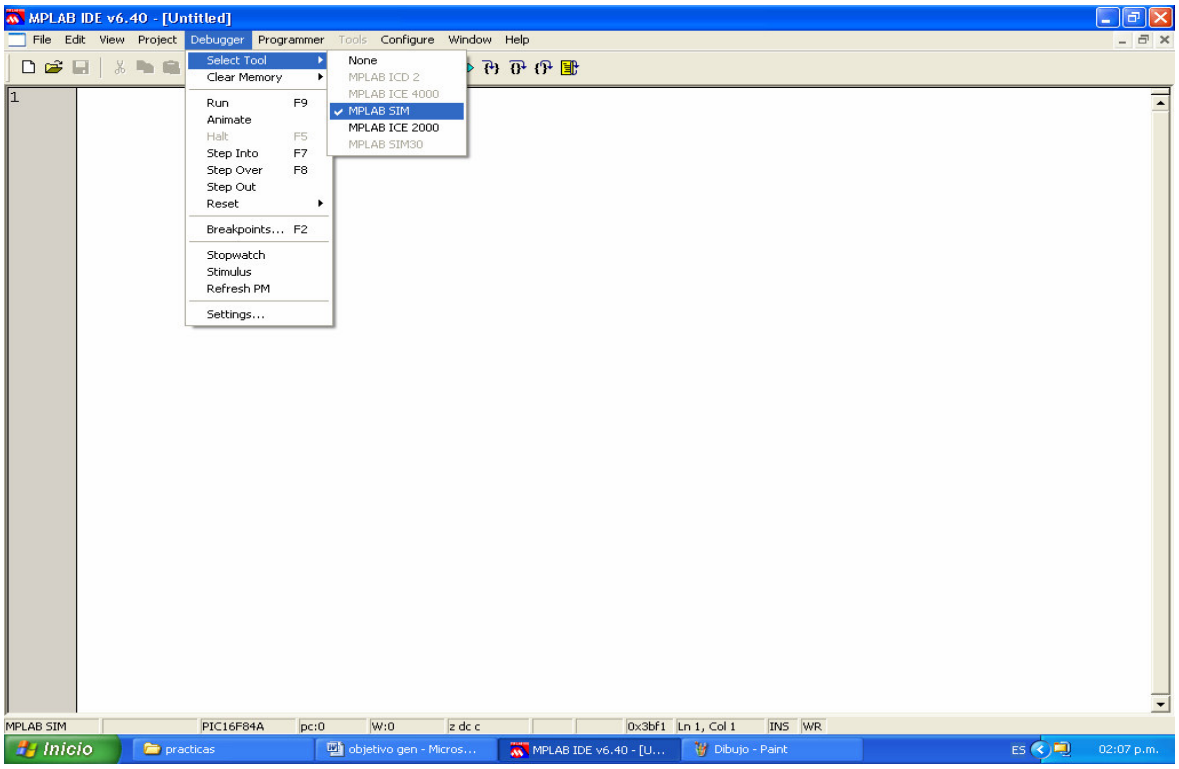


Figura 1.5. Activación del simulador

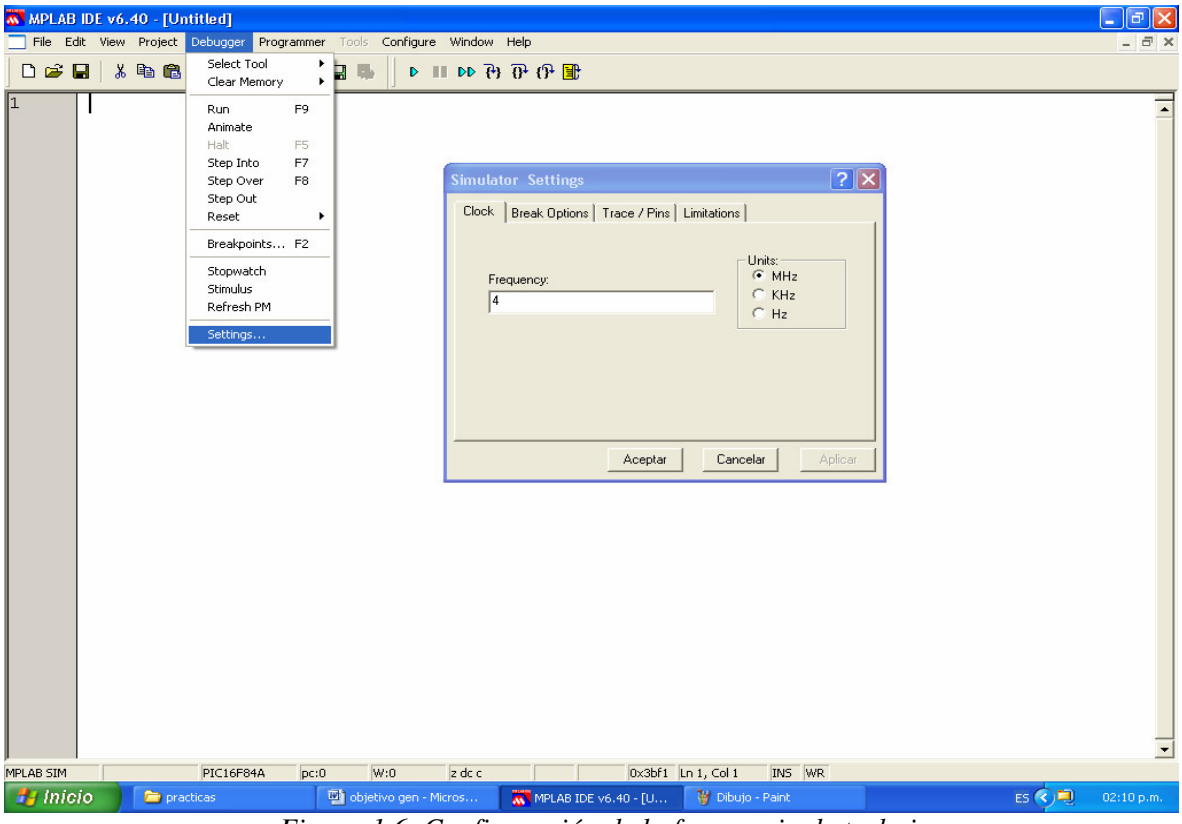


Figura 1.6. Configuración de la frecuencia de trabajo

6. Si desea visualizar el número de cada línea active el menú Edit-Properties y seleccione las opciones que se muestran en la figura 1.7.

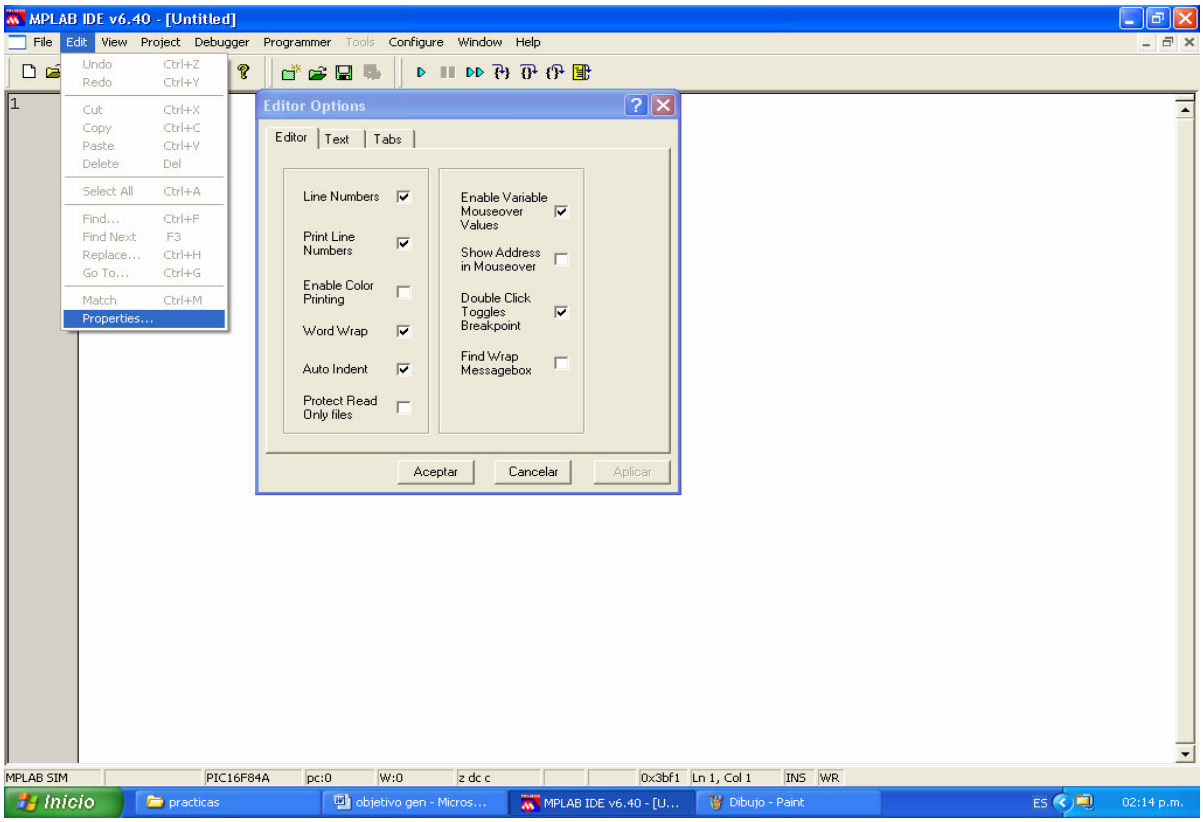


Figura 1.7. Para visualizar el número de cada línea

7. Escriba el siguiente programa fuente considerando las reglas para escribirlo.

```
; PRACTICA_1.asm
; Por los leds conectados en el puerto A se visualiza el dato introducido por los
;dipswitch conectados de RB0 a RB4 es decir, si se introduce el dato
 ;xx10101 por los dipswitch conectados en el puerto B se visualizara
;10101 por los leds conectados en el puerto A (un 1 lógico
; indica un led encendido y un 0 lógico un led apagado)
;Zona de datos ( a partir del siguiente renglones se escribe el programa)

   __CONFIG _CP_OFF & _WDT_OFF& _PWRTE_ON & _XT_OSC

;Configuración para el  grabador. Observe que antes de la palabra CONFIG
;van 2 guiones bajos y no uno. Esta configuración la llevan todos los programas.

        LIST        P=16F84A     ; Se indica al ensamblador el procesador
                                 ;utilizado.
    INCLUDE <P16F84A.INC> ; Definición de los operandos utilizados.
```

```
            ORG  0   ;Se indica que el programa comienza en la dirección 0 de
                     ; memoria   de programa.
Inicio      bsf      STATUS,RP0     ;Pone a 1 el bit 5 del STATUS. Acceso al
                                    ;Banco 1
            clrf     TRISA          ; Las líneas del Puerto A se configuran
                                    ;como salidas
            movlw b'11111111'
            movwf TRISB             ; Las líneas del Puerto B se configuran
                                    ; como entradas
            bcf      STATUS,RP0     ;Pone a 0 el bit 5 de STATUS. Acceso al
                                    ;Banco 0.
Principal                           ;etiqueta  de programa llamada Principal.
            movf   PORTB,W          ; Lee el Puerto B y lo almacena en W.
            movwf PORTA             ; El contenido de W se visualiza por el
                                    ;Puerto A.
            goto     Principal      ; Crea un bucle cerrado.
            END                     ; Fin del programa.
```

8. Guarde el programa en la carpeta que fue creada para tal fin activando el menú File-save as asignándole el nombre Practica_1.asm.
9. Ensamble el programa activando el menú Proyect-Quickbuild Practica_1.asm (Figura 1.8)

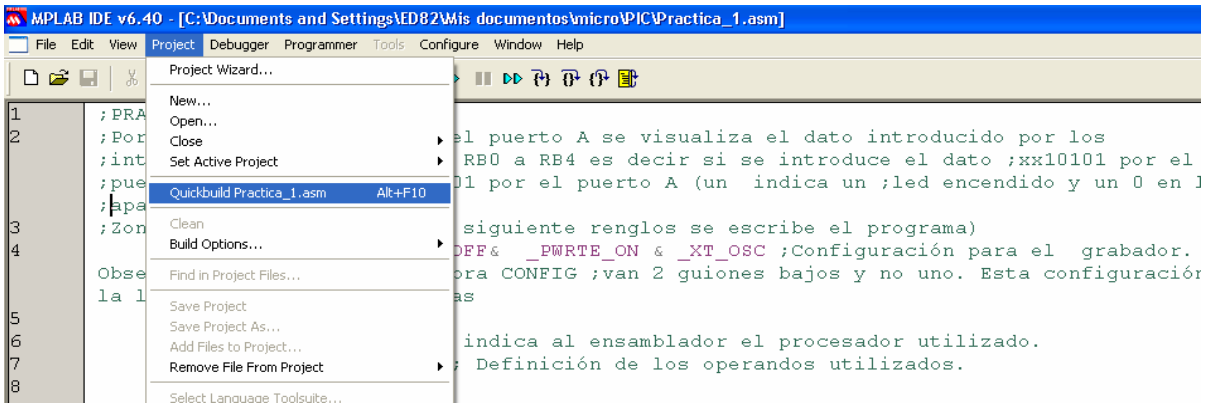


Figura 1.8. Ensamblar el programa

Hecho esto se observara el proceso de ensamblado
Y si aparece una pantalla con la leyenda "BUILD SUCCEEDED" se esta en condiciones de grabar el PIC. En caso de que aparezca la leyenda "BUILD FAILED" se indica que existen errores los cuales hay que corregir. El proceso de ensamblado crea una lista de errores y haciendo doble clic en ellos se accede a la línea en donde se encuentra el error y hay que corregirlo. Una vez corregido es necesario volver a ensamblar el programa.

10. Conecte el grabador de PIC's a uno de los puertos serie COM de la computadora.
11. inserte el PIC en el grabador.
12. Inicie el programa IC-PROG cuyo icono se encuentra en el escritorio.
13. Seleccione el PIC que se esta utilizando (PIC 16F84A) así como el tipo de oscilador que se esta usando que en nuestro caso es un cristal de cuarzo (XT) y habilite la casilla PWRT tal como se muestra con las flechas en la Figura 1.9.

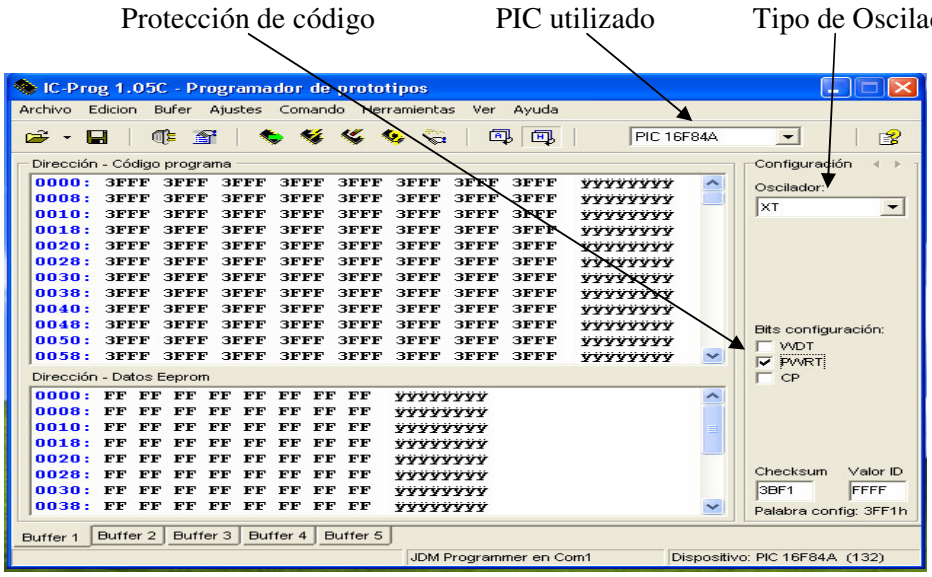


Figura 1.9. Configuración del grabador de PIC

14. Si el programa esta en español continué con el paso 14. En caso contrario seleccione el menú *setting-option-languge* y elija el lenguaje español.
15. Es necesario configurar el hardware del programador que en nuestro caso es un programador compatible con JMD y seleccionar el puerto serie al cual se esta colocando el grabador (si no sabe a que puerto serie esta conectado el grabador pregunte a su instructor). Para realizar esto se selecciona el menú *ajustes-Tipo de hardware* tal como se muestra en la Figura 1.10.

    Una vez hecho esto el programa grabador (icprog) esta listo para ser usado para la grabación del PIC.
16. Abra de la carpeta PIC (la que fue creada para guardar todos los programas) que se encuentra en la computadora el archivo Practica_1.hex (Figura 1.11).
17. Observe que aparecen una serie de caracteres en la ventana de código de programa, los cuales corresponden al lenguaje maquina codificado en forma hexadecimal tal como se ve en la Figura 1.12.

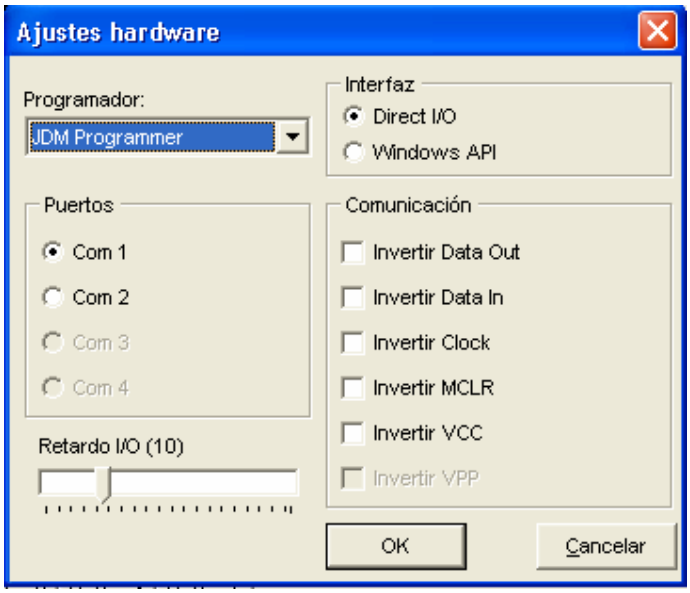


Figura 1.10. Selección del puerto serie a utilizar

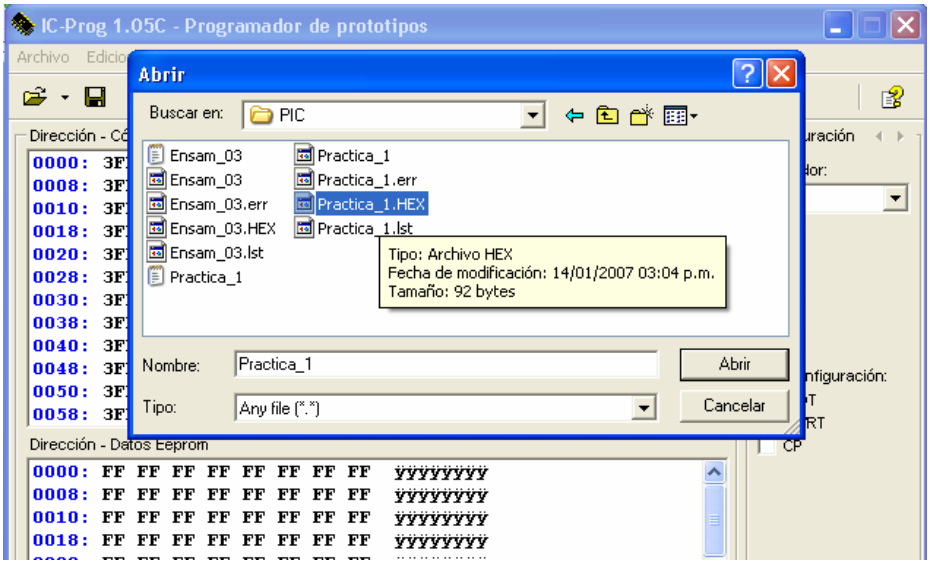


Figura 1.11. Abrir el programa escrito

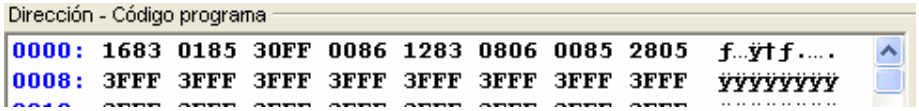


Figura 1.12. Lenguaje maquina del programa escrito

18. Para grabar el PIC vaya al menú *comando-programar todo* o también puede utilizar el icono sobre la barra de herramientas que se muestra en la Figura 1.13.

Figura 1.13. Botón para grabar el PIC

19. Observe que en la pantalla aparece una ventana donde se observa el proceso de grabación y en seguida se verifican los datos escritos en el chip y también se observa el proceso y en caso de que la verificación sea correcta aparece una ventana que confirma tal hecho con lo que el proceso de grabación termina. Extraiga el PIC programado del grabador.
20. Introduzca el PIC programado en el circuito de la figura 1.14 .
21. En caso de que el grabador de PIC's este bien hecho se debe observar que el dato que se introduce por los interruptores es el mismo que el que se observa por los leds.
22. De sus conclusiones.

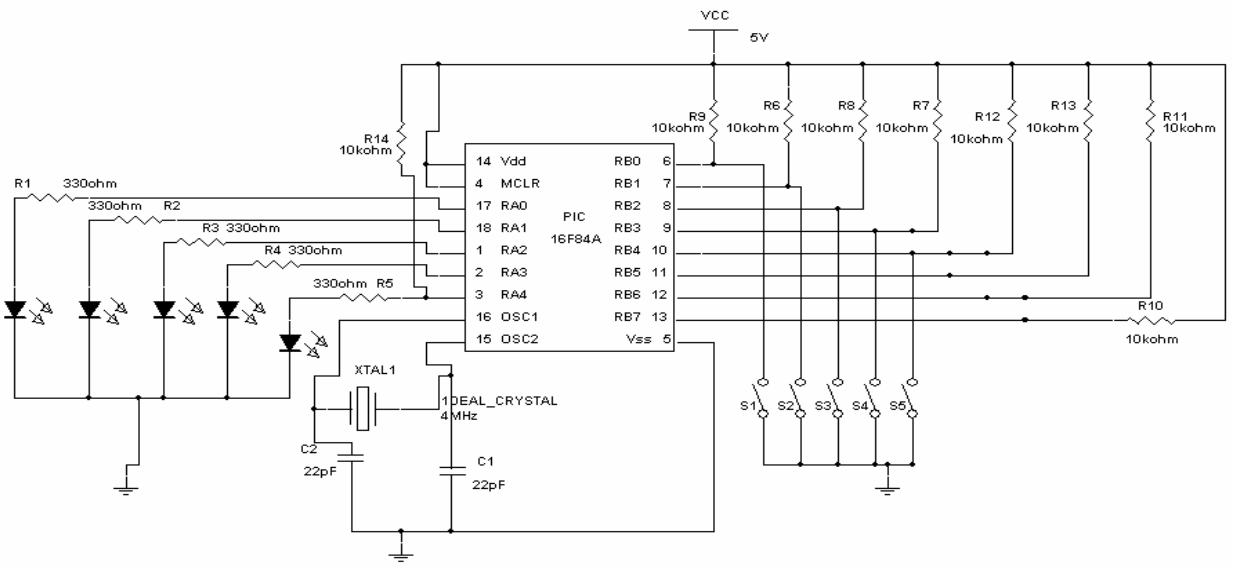


Figura 1.14. Circuito de la Práctica 1

3.2.- Práctica 2:

"PROGRAMACIÓN ELEMENTAL DEL PIC UTILIZANDO SU SET DE INSTRUCCIONES".

Objetivos.

- Conocer la metodología básica para la implementación de sistemas con microcontroladores.
- Conocer la metodología básica de programación para el desarrollo de programas para el PIC16F84A utilizando su set de instrucciones.
- Familiarizar al alumno con el set de instrucciones del PIC 16F84A
- Aprender a configurar los puertos del PIC 16F84A

Introducción.

El PIC16F84A cuenta con un set de 35 instrucciones que nos permite programar al dispositivo para que realice las tareas que le encomendemos. Todas las instrucciones tardan en ejecutarse un ciclo maquina excepto las instrucciones de salto que duran 2. Las 35 instrucciones del PIC se muestran en la Tabla 2.1.

Escritura de constantes:

Cuando se introducen constantes es necesario avisarle al programa ensamblador. El ensamblador MPLAB permite introducir constantes en forma decimal, hexadecimal, binario, octal y códigos alfanuméricos ASCII. El formato de escritura se muestra en la Tabla 2.2.

Las constantes hexadecimales que comiencen con una letra (A-F) deben ir precedidas de un cero por que de lo contrario se pueden ser confundidas con una etiqueta, por ejemplo: 0EDh.

Diseño de sistemas con un microcontrolador:

La implementación de sistemas con microcontroladores tiene la finalidad de resolver un problema específico para el cual se necesita de un hardware (microcontrolador y periféricos) y un software (se realiza utilizando el set de instrucciones del microcontrolador). Para poder implementar dicho sistema se deben de seguir una serie de pasos que a continuación se enumeran:

1. Lo primero que se necesita saber es específicamente que es lo que necesito que el microcontrolador realice (planteamiento del problema).

Nemónico/operando	Descripción	Flags afectados
Instrucciones para operar con registros de 1 byte		
addwf f,d	Sumar w y f y el resultado se almacena en d	C,DC,Z
andwf f,d	AND entre w y f, el resultado se almacena en d	Z
clrf f	Pone a 0 el registro f	Z
clrw	Pone a 0 w	Z
comf f,d	Complementar f, el resultado se almacena en d	Z
decf f,d	Decrementa f y el resultado se almacena en d	Z
decfsz f,b	Decrementar f, si es 0 salta la siguiente instrucción	
incf f,d	Incrementa f y el resultado se almacena en d	Z
incfsz f,b	incrementar f, si es 0 salta la siguiente instrucción	
iorwf f,d	OR entre w y f, el resultado se almacena en d	Z
movf f,d	Mover f a d	Z
movwf f	Mueve el dato de w a f	
nop	No operación	
rlf f,d	Rota a la izquierda a través del carry y el resultado se	C
rrf f,d	almacena en d	C
subwf f,d	Rota a la derecha a través del carry y el resultado se	C,DC,Z
swapf f,d	almacena en d	
xorlwf f,d	Resta f-w y el resultado se almacena en d	Z
	Intercambia los nibbles de f y se almacena en d	
	XOR entre w y f, el resultado se almacena en d	
Instrucciones que operan con un bit de un registro		
bcf f,b	Pone a 0 el bit ‘b’ del registro f	
bsf f,b	Pone a 1 el bit ‘b’ del registro f	
btfsc f,b	Checa el bit ‘b’ de ‘f’, salta si es 0	
btfss f,b	Checa el bit ‘b’ de ‘f’, salta si es 1	
Instrucciones de control y operación con una literal (constante)		
addlw k	Sumar k con w y el resultado se almacena en w	C,DC,Z
andlw k	AND entre k y w, el resultado se almacena en w	Z
call k	Llamada de la subrutina k	
clrwdt	Borrar el timer del watchdog	/TO,/PD
goto k	Salta a la dirección k	
iorlw k	OR entre w y k, el resultado se almacena en w	Z
movlw k	Mueve la literal k a w	
retfie	Retorno de una interrupción	
retlw k	Retornar y cargar w con el valor de k	
return	Retorno de subrutina	
sleep	Entra en modo de bajo consumo	/TO,/PD
sublw k	Resta k-w y el resultado se almacena en w	C,DC,Z
xorlw k	XOR entre w y k, el resultado se almacena en w	Z

Si d=0 el resultado se almacena en w.

Si d=1 el resultado se almacena en el registro

Tabla 2.1

Tipo	Sintaxis	Ejemplo
Decimal	D'constante' d'constante' **.**constante	movlw D'69' movlw d'69' movlw **.**69
Hexadecimal	H'constante' h'constante' 0xconstante constanteH constanteh	movlw H'F8' movlw h'F8' movlw 0xF8 movlw 0F8H movlw 0F8h
Octal	O'constante' o'constante'	movlw O'672' movlw o'672'
Binario	B'constante' b'constante'	movlw B'10110' movlw b'10110'
ASCII	A'carácter' A'carácter' 'carácter'	movlw A'E' movlw a'E' movlw 'E'

Tabla 2.2

2. Diseñar (dibujar) el circuito físico que controlara el programa.
3. Realizar un algoritmo que resuelva dicho problema especificando paso por paso. Cabe mencionar que pueden existir varios algoritmos diferentes que resuelvan el mismo problema. Lo importante aquí es que la solución planteada sea eficiente.
4. Realizar el programa en el lenguaje ensamblador utilizando el set de instrucciones del microcontrolador interpretando el algoritmo.
5. Compilar y programar el microcontrolador.
6. Alambrar el circuito físico que realiza la función para la cual fue diseñado y se planteo en el paso 1.
7. Compruebe el funcionamiento del circuito. En caso de que no cumpla con el objetivo para el cual fue hecho repetir el procedimiento hasta que se cumpla el objetivo.

Uno de los métodos mas sencillos para plantear algoritmos es mediante la realización de diagramas de flujo en donde se plantea la solución de un programa de forma grafica. Este método muestra de forma grafica el orden en que los pasos de la solución se llevan a cabo. Los diagramas de flujo están constituidos por una serie de símbolos unidos por flechas que indican el orden en que sigue el programa. Los símbolos más importantes son los que se muestran en la Figura 2.1.

A continuación indicaremos el significado de estos símbolos.

- **Inicio o Fin:** Indican el inicio o el final del programa.
- **Proceso:** son la mayor parte de las operaciones realizadas por el programa.

- **Decisión:** Proporciona dos caminos distintos dependiendo de que la respuesta a la pregunta sea si o no.
- **Subrutinas:** son el conjunto de operaciones que se repiten varias veces durante el programa.
- **Conectores:** indican el camino que sigue un programa muy largo que no cabe en una hoja. Dos conectores con el mismo numero o símbolo indican el punto donde se interrumpe y continúa el programa.
- Las **flechas** indican el camino que sigue el programa.

Una vez que se planteo el algoritmo con un diagrama de flujo se procede a escribir el programa utilizando el set de instrucciones del microcontrolador.

Así por ejemplo para el programa de la práctica 1 cuyo enunciado es:

```
;PRACTICA_1.asm
; Por los leds conectados en el puerto A se visualiza el dato introducido por los
;dipswitch conectados de RB0 a RB4 es decir, si se introduce el dato
;xx10101 por los dipswitch conectados en el puerto B se visualizara
;10101 por los leds conectados en el puerto A (un 1 lógico
; indica un led encendido y un 0 lógico un led apagado)
```

El diagrama de flujo queda como se ve en la Figura 2.2.

Configuración de los puertos como entrada/salida

Para configurar cada una de las líneas de los puertos se utilizan los registros especiales TRISA y TRISB. Si el bit correspondiente a una de las líneas de cualquier puerto se pone a "0" se esta configurando esta línea como salida y si se pone a "1" es configurada como entrada. Por ejemplo:

```
bsf   STATUS,RP0          ; Pone a 1 el bit 5 del STATUS.
                          ; Acceso al Banco 1
movlw     b'00000'        ; Las líneas del Puerto A se
movwf     TRISA           ; configuran como salidas
movlw          b'11111111' ; Las líneas del Puerto B se
movwf          TRISB       ;configuran como entradas
bcf       STATUS,RP0      ; Pone a 0 el bit 5 de STATUS.
                          ; Acceso al Banco 0.
```

Esta es la estructura general para la configuración de los puertos.

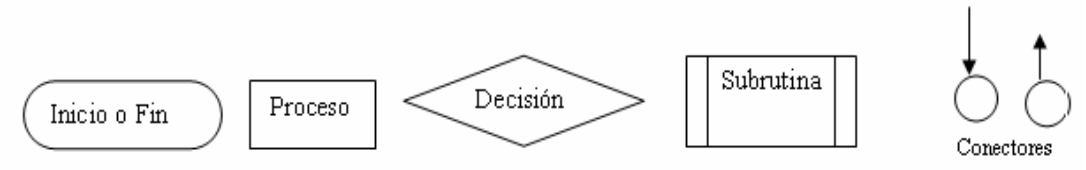


Figura 2.1. Símbolos de los diagramas de flujo

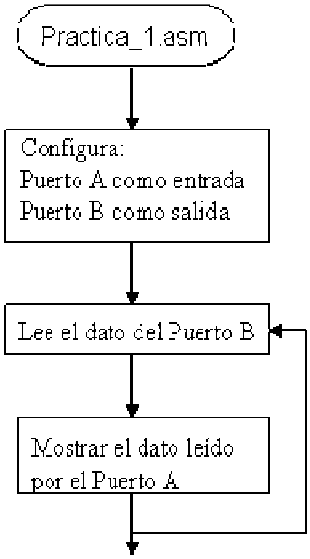


Figura 2.2. Diagrama de flujo de la Práctica 1

Material y equipo.

Grabador de PIC's
PIC 16F84A
Cristal de cuarzo de 4MHz
2-Capacitores de 22pF
2-Dip switch
8-Diodos led
8-resistencias de 330 Ω
8-resistencias de 10 *k*Ω
1-PC
1-Fuente de alimentación

Desarrollo.

1. Lea el siguiente enunciado

   Por el puerto B se obtiene el dato del puerto A pero invertido. Es decir si se introduce por el puerto A el dato "01101" por el puerto B se observara el dato "xxx10010"
2. Alambre el circuito de la Figura 2.3

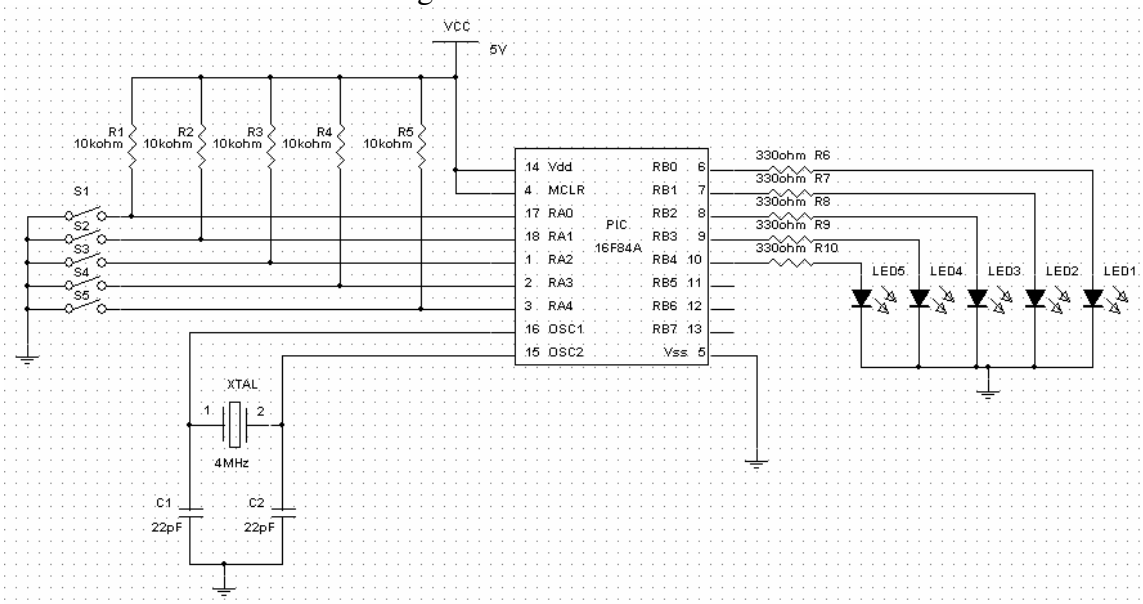


Figura 2.3. Circuito de la Práctica 2

   Donde cada uno de los interruptores cerrados representa un "0" y abiertos representan un "1". Un led encendido representa un "1" y uno apagado representa un "0".
3. Realice el diagrama de flujo para este problema.
4. Realizar el programa en el lenguaje ensamblador utilizando el set de instrucciones del microcontrolador interpretando el diagrama de flujo.
5. Compilar y programar el microcontrolador.
6. Alambrar el circuito físico del paso 2
7. Compruebe el funcionamiento del circuito. En caso de que no cumpla con el objetivo para el cual fue hecho repetir el procedimiento hasta que se cumpla el objetivo.
8. De sus conclusiones.

3.3.- Práctica 3:

"CONEXIÓN DEL PIC 16F84A CON DISPOSITIVOS PERIFÉRICOS".

Objetivos:

- Conocer los elementos necesarios para implementar un sistema mínimo con un microcontrolador.
- Realizar programas donde el PIC interactué con algunos de los dispositivos periféricos conectados en sus puertos

Introducción.

Todos los sistemas con microprocesadores necesitan de un hardware y un software.

El hardware es la parte tangible y esta formado por los elementos físicos del sistema (microcontrolador y periféricos)
El software es la parte intangible y es el programa que le indica al microcontrolador la función que debe realizar. Este software se realiza utilizando el set de instrucciones del microcontrolador.

Los elementos necesarios para la implementación de un sistema mínimo con microprocesadores se muestran en la Figura 3.1.

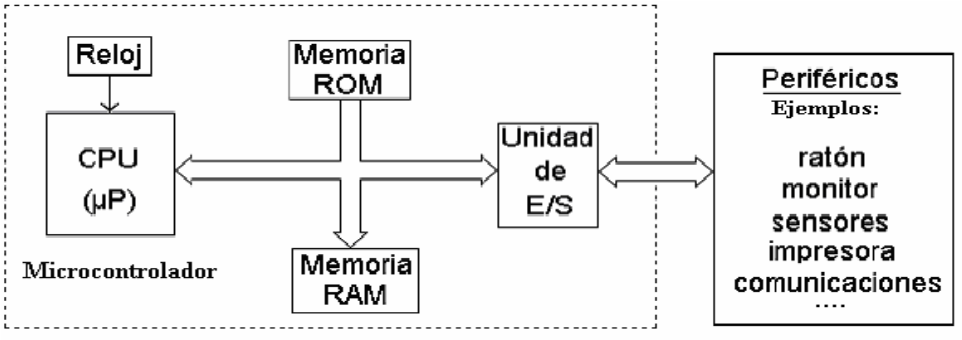


Figura 3.1. Sistema mínimo

Como puede observarse en la Figura 3.1 los elementos que se encuentran dentro del recuadro son los elementos que forman un microcontrolador. Por lo que un microcontrolador solo necesita de la conexión de periféricos en sus puertos (A o B) para la implementación de un sistema mínimo.

Para la conexión de periféricos es necesario conocer las características eléctricas del PIC.

Los puertos (A y B) son las líneas con las cuales el microcontrolador se comunica con el mundo exterior. Estos puertos pueden manipular señales digitales con valores entre 0 y 5V.

Estos puertos pueden configurarse como entradas para recibir datos o como salidas para gobernar dispositivos externos.

Cada línea de salida puede suministrar una corriente máxima de 20 mA y si es de entrada puede absorber hasta 25 mA. Al existir una limitación en la disipación máxima de la potencia del chip la suma de la corriente de las 5 líneas del Puerto A no puede exceder de 80 mA cuando absorbe corriente y de 50 mA cuando suministra corriente. La Puerta B puede absorber un máximo de 150 mA y suministrar un total de 100 mA (que son la suma de las corrientes de las 8 líneas del puerto).

Ya que el PIC es un dispositivo diseñado con tecnología CMOS las líneas de los puertos que no se estén utilizando se deberán conectar a la fuente de 5V por medio de una resistencia de $10K\Omega$ ya que si se dejan sin conectar se puede dañar el integrado. Esto lo puede observar en el circuito utilizado en la práctica 1.

Una vez que se configuran los puertos como entradas o salidas, el PIC esta en condiciones de interactuar con el mundo exterior y para esto necesita de algunos dispositivos que adapten las manifestaciones del mundo físico a una forma que él pueda interpretar es decir en forma digital o de dispositivos los cuales controla para que realizan una función determinada.

Algunos de los dispositivos básicos que sirven para esta función se describen a continuación.

- **Diodo led:** son dispositivos con los cuales se pueden usar para la comprobación de los circuitos mediante la emisión de luz. Estos dispositivos generalmente son controlados por el microcontrolador por lo que se conectan en una línea configurada como salida. Existen dos formas de conectar el led, las cuales se muestran en la Figura 3.2.

  El led 2 se activa con un nivel alto (“1”) proporcionado por el microcontrolador y el led 1 se activa con un nivel bajo (“0”).

- **Relevadores:** la utilización de relevadores es la forma más adecuada de controlar procesos que utilizan corriente alterna. La

figura 3.3 muestra el diagrama de un circuito que activa el relevador de 5V y 200mA con un estado alto en su salida.

- **Interruptores y pulsadores:** estos dispositivos permiten introducir niveles lógicos "0" (0V) cuando están cerrados y "1" (5V) cuando están abiertos. La forma de conectarlos se muestra en la Figura 3.4.

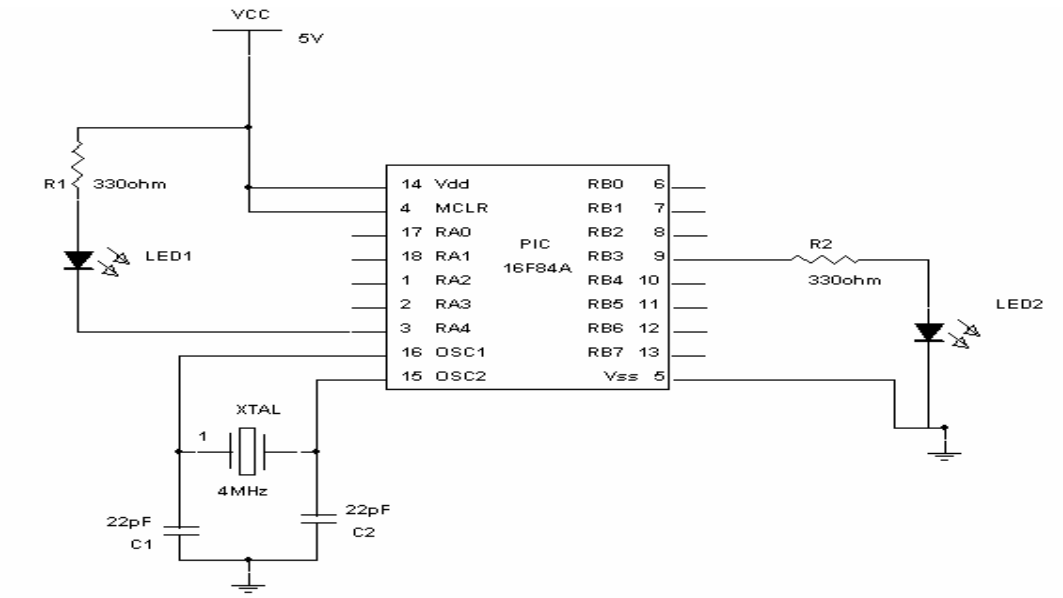


Figura 3.2. Conexión de diodo led.

- **Opto acopladores:** estos dispositivos permiten la introducción de datos en forma digital cuando los niveles de voltaje de CD son muy grandes e incluso cuando se trabaja con voltaje de corriente alterna de 127V. La forma de conexionar estos dispositivos con el PIC se muestra en la Figura 3.5 en donde se utiliza un opto acoplador 4N25. Cuando se aplica un voltaje Vin el diodo conduce emitiendo luz y por tanto el transistor se satura, por lo que el voltaje de colector será de casi 0V o un "0". Cuando el voltaje Vin es 0V el diodo no conduce por lo que el transistor se corta teniendo un voltaje de colector de casi 5V o un "1". El voltaje de polarización del diodo en directa es de 1.15V y su corriente de conducción es de 10mA. Es necesario hacer circular esta corriente a través del diodo por lo que se coloca una resistencia en serie cuyo valor se calcula como sigue:

  $R = \frac{VIN - 1.15}{10mA}$ En el diagrama siguiente R1 y R3se calcularon con esta formula. Nótese que cuando se trabaja con CA se debe proteger el diodo del opto acoplador con un diodo de protección en inversa, esto debido a que el diodo del opto acoplador permite un

voltaje máximo de polarización inversa de 3V.Es necesario tomar en cuenta la potencia que disipa R1 ya que si no se considera se puede quemar.

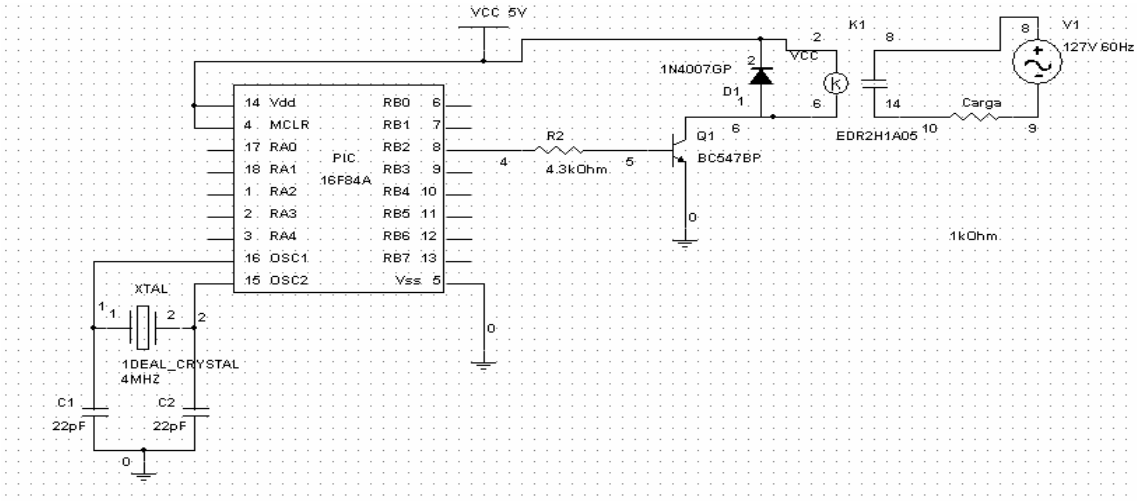


Figura 3.3. Conexión de relevadores

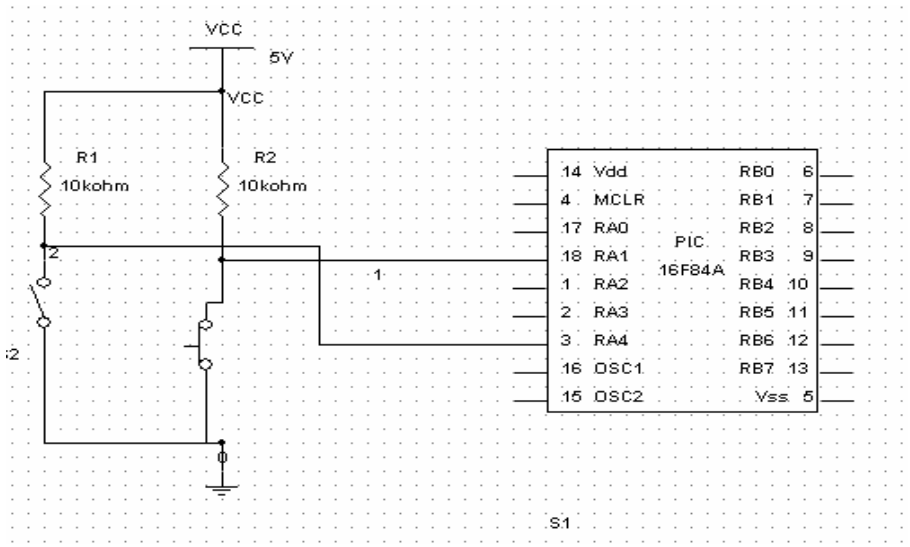


Figura 3.4. Conexión de switch pulsadores

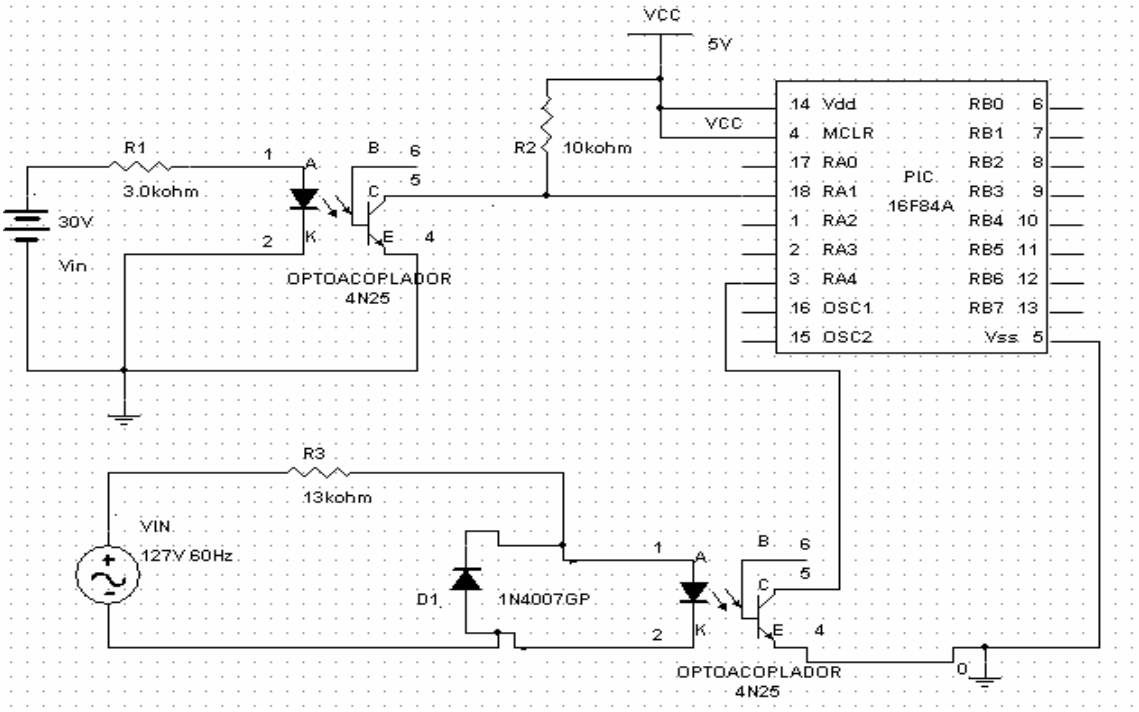


Figura 3.5. Conexión de opto acopladores

Material y equipo.

1-Grabador de PIC's
1-PIC16F84A
1-cristal de cuarzo de 4MHZ
2-Capacitores de 22pF
5-resistencias de 330 Ω
7-resistencias de 10 $k\Omega$
1-resistencia de 1.1 $k\Omega$
1-resistencia de 4.3 $k\Omega$
1-opto acoplador 4N25
1-transistor BC547
1-diodo 1N4007
1-Relevador de 5V y 200mA
1-PC
1-Fuente de alimentación
1-foco a 127V/100W

Desarrollo.

1. Para el circuito de la Figura 3.6 realice un programa en el cual cada vez que entre un "0" por la terminal RA0 active el relevador conectado en RB2 ("1") y cuando entre un "1" por RA0 se desactive el relevador en RB2 ("0")

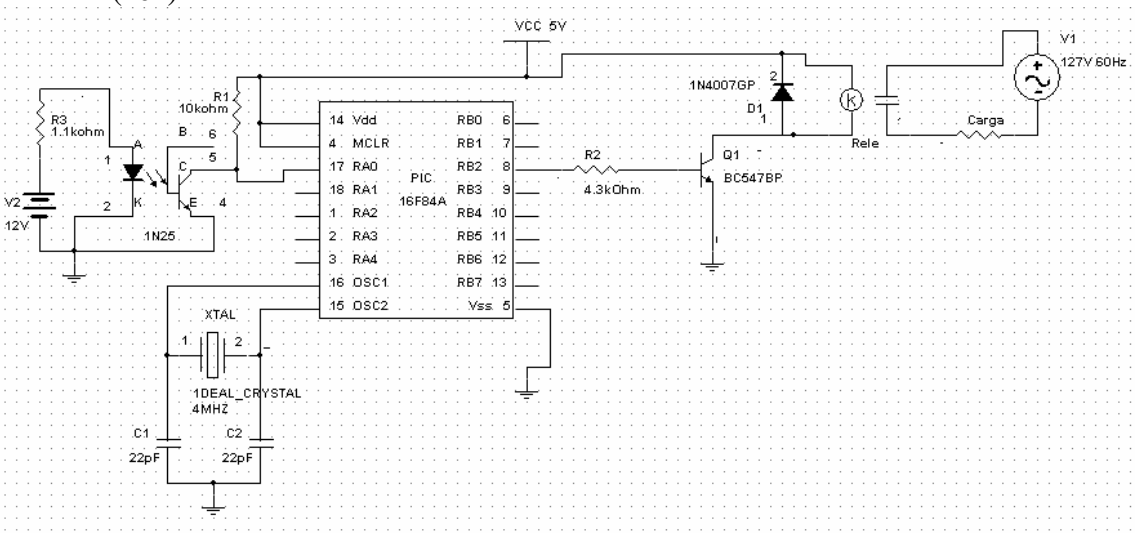


Figura 3.6. Circuito de la Práctica 3

2. Realizar un algoritmo que resuelva dicho problema especificando paso por paso. Cabe mencionar que pueden existir varios algoritmos diferentes que resuelvan el mismo problema. Lo importante aquí es que la solución planteada sea eficiente.
3. Realizar el programa en el lenguaje ensamblador utilizando el set de instrucciones del microcontrolador interpretando el algoritmo.
4. Compilar y programar el microcontrolador.
5. Alambrar el circuito de la Figura 3.6 que realiza la función para la cual fue diseñado y se planteo en el paso 1. En la carga conecte un foco.
6. Compruebe el funcionamiento del circuito variando el voltaje de V2, cuando este es casi 0V debe de haber un "1" en el colector de transistor del opto acoplador (terminal 17 del PIC) y cuando hay casi 12V hay un "0". En caso de que no cumpla con el objetivo para el cual fue hecho repetir el procedimiento hasta que se cumpla el objetivo.
7. De sus conclusiones

3.4.- Práctica 4:

"SALTOS".

Objetivo.

- Implementar sistemas con microcontroladores capaces de tomar decisiones.

Introducción.

En la mayoría de las aplicaciones es muy difícil que se realicen procesos cuyo programa siga una secuencia lineal de pasos, es decir la ejecución una instrucción tras otra. Muchas veces el programa salta de un punto del programa a otro debido a que el programa tomo una decisión o por algún otro motivo. A continuación analizaremos las instrucciones de salto con las que cuenta el PIC16F84A.

Salto incondicional:

Este tipo de salto se da con la instrucción ***goto k***. Esta instrucción le indica al programa que salte a la dirección *k* que es la nueva dirección a partir de la cual se comenzaran a leer las instrucciones después del salto. Esta dirección *k* puede ser una etiqueta.
Ejemplo:

```
        -------
        goto   primero
        --------
        --------
primero --------
        ---------
```

Saltos condicionales:

Se produce este tipo de saltos cuando se cumple con una condición determinada. Las instrucciones que sirven para este tipo de saltos se describen a continuación.

Instrucción: ***btfsc f,b*** (checa el bit'b de 'f' salta si es 0)
Esta instrucción verifica el estado lógico de un bit de un registro o de cualquier puerto de entrada o salida. Si el estado es "0" se ignora la siguiente instrucción y se salta. Si el estado es "1" la instrucción que sigue a esta se ejecuta normalmente.

Ejemplo:

```
         ---------
         btfsc     PORTA,0; se checa el estado del bit 0 del puerto A
uno      goto      x ; si es "1" se ejecuta esta instrucción mandando el programa a x
cero     ---------- ; si es "0" se ejecuta esta instrucción ignorándose la de arriba.
```

Instrucción: ***btfss f,b*** (checa el bit 'b' de 'f' salta si es 1).
Esta instrucción verifica el estado lógico y salta en forma contraria a la anterior.
Ejemplo:

```
         ---------
         btfss     PORTB,7; se checa el estado del bit 7 del puerto B
cero     goto      x ; si es "0" se ejecuta esta instrucción mandando el programa a x
uno      ---------- ; si es "1" se ejecuta esta instrucción ignorándose la de arriba.
```

Instrucción: ***decfsz f,d*** (decrementa 'f' salta si es "0").
Esta instrucción decrementa en una unidad el valor del registro 'f' y salta si el resultado es "0" ignorando la siguiente instrucción. Si es "1" ejecuta la instrucción siguiente. El valor de 'd' determina el lugar en donde se almacena el resultado, si 'd'=0 es lo mismo que 'd'=w por lo que el resultado se almacena en w; si 'd'=1 se almacena en el registro 'f'.
Ejemplo:

```
         -------------
         decfsz       contador,1 ; decrementa el registro contador y se almacena en este
mismo.
         goto         es uno ;si es uno se ejecuta esta instrucción
es cero  ---------------   ; si es cero se ignora la instrucción de arriba y se ejecuta esta.
```

Instrucción: ***inccfsz f,d*** (incrementa 'f' salta si es "0").
Esta instrucción incrementa en una unidad el valor del registro 'f' y salta si el resultado es "0" ignorando la siguiente instrucción. Si es "1" ejecuta la instrucción siguiente. El valor de 'd' determina el lugar en donde se almacena el resultado, si 'd'=0 es lo mismo que 'd'=w por lo que el resultado se almacena en w; si 'd'=1 se almacena en el registro 'f'.
Ejemplo:

```
         -------------
         incfsz       contador,1 ; decrementa el registro contador y
                                 ;se almacena en este mismo.
         goto         es uno ;si es uno se ejecuta esta instrucción
es cero  ---------------   ; si es cero se ignora la instrucción de arriba y se ejecuta esta.
```

Una de las principales aplicaciones de las anteriores instrucciones es la comparación de registros y en base a si son iguales o diferentes se realiza un proceso u otro cuando la señal CLK cambia de estado. Esta comparación se muestra en la Figura 4.1.

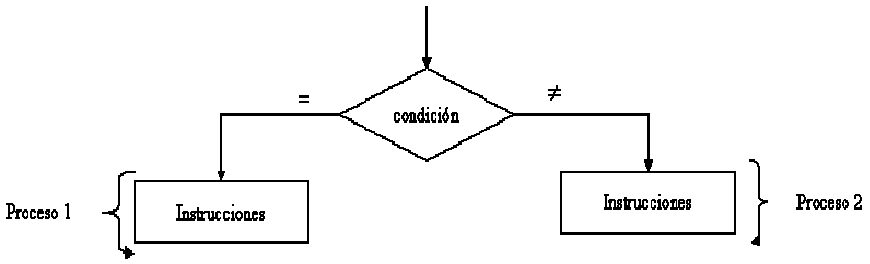


Figura 4.1. Comparación de registros

Un ejemplo de este tipo de bucle es el siguiente:

```
          --------
          btfsc   PORTA,0 ;checa el bit0 del puerto A salta si es "0"
          goto    alto    ; si el bit 0 del puerto A es "1" ve a la etiqueta alto
          goto    bajo    ; si el bit 0 del puerto A es "0" ve a la etiqueta bajo
alto      btfsc   PORTA,1  ;checa el bit 1 del puerto A salta si es "0"
          goto    iguales      ; si el bit 1 del puerto A es "1" ve a la etiqueta iguales
          goto    diferentes ; si el bit 1 del puerto A es "0" ve a la etiqueta diferentes
bajo      btfsc   PORTA,1  ;checa el bit 1 del puerto A salta si es "0"
          goto    diferentes  ; si el bit 1 del puerto A es "1" ve a la etiqueta diferentes
          goto    iguales     ; si el bit 1 del puerto A es "0" ve a la etiqueta iguales
diferentes ------    ; se ejecuta el Proceso 2 ya que el estado de ambos bits es diferente
          -------
iguales   -------    ;se ejecuta el proceso 1 ya que el estado de ambos bits es igual.
```

Otra de las principales aplicaciones es la creación de bucles que son partes del programa que se repiten un número finito o infinito de veces.

Por ejemplo la instrucción ***goto primero*** crea un **bucle cerrado infinito**. Como se muestra en la Figura 4.2.

Se pude dar la posibilidad de crear un **bucle cerrado finito** partiendo de una condición. Si esta condición no se cumple no se puede seguir con el programa y se espera a que se cumpla la condición para que se ejecute la parte del programa que falta. Por ejemplo

```
bajo
     btfsc  PORTB,0  ;se verifica el estado del pin 0 del puerto B si es "0"
                     ; salta la siguiente  instrucción
     goto   bajo      ;si es "1"se ejecuta esta instrucción yendo a la etiqueta bajo.
     ----------      ;no se puede ejecutar esta instrucción hasta que RB0 esta en "0"
```

Este tipo de bucle tiene la forma que se muestra en la figura 4.3.

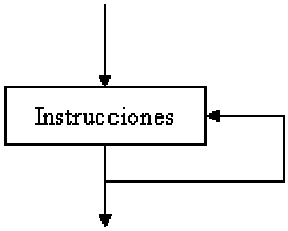


Figura 4.2. Bucle cerrado infinito

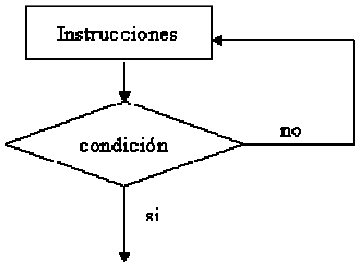


Figura 4.3.Bucle cerrado finito

Existe otro tipo de bucle el cual es muy útil para la creación de tiempos (este tema se tratara en la práctica 5) el cual se repite un numero conocido de veces y cuando finaliza ese numero de veces continua ejecutando el programa. Por ejemplo:

```
          ----------
          movlw     d'60'   ;se carga w con el valor 60 que es el numero de veces
                            ;que se repite el bucle
          movwf     contador    ;se carga el registro contador con el valor
                                ; de w (60 en este caso)
bucle     decfsz    contador,F  ;se decrementa el valor del registro contador
                                ;y se almacena en contador
          goto      bucle       ;si no es cero se manda a la etiqueta bucle
          ----------            ; si es cero se ejecuta esta instrucción ignorándose
                                ;el goto de arriba.
```

Nota: en la instrucción ***decfsz contador,F*** al comienzo el valor de contador es 60, enseguida se decrementa y el nuevo valor de contador es 59 y así sucesivamente hasta cero.
La estructura general para este tipo de saltos se muestra en la Figura 4.4.

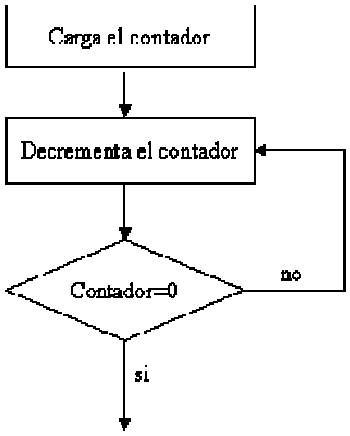


Figura 4.4. Bucle que se repite n veces

Material y equipo.

1-Grabador de PIC's
1-PIC16F84A
1-cristal de cuarzo de 4MHZ
2-Capacitores de 22pF
1-Dip switch
2-Diodos led
2-resistencias de 330 Ω
8-resistencias de 10 $k\Omega$
1-PC
1-Fuente de alimentación

Desarrollo.

1. Arme el circuito de la Figura 4.5.

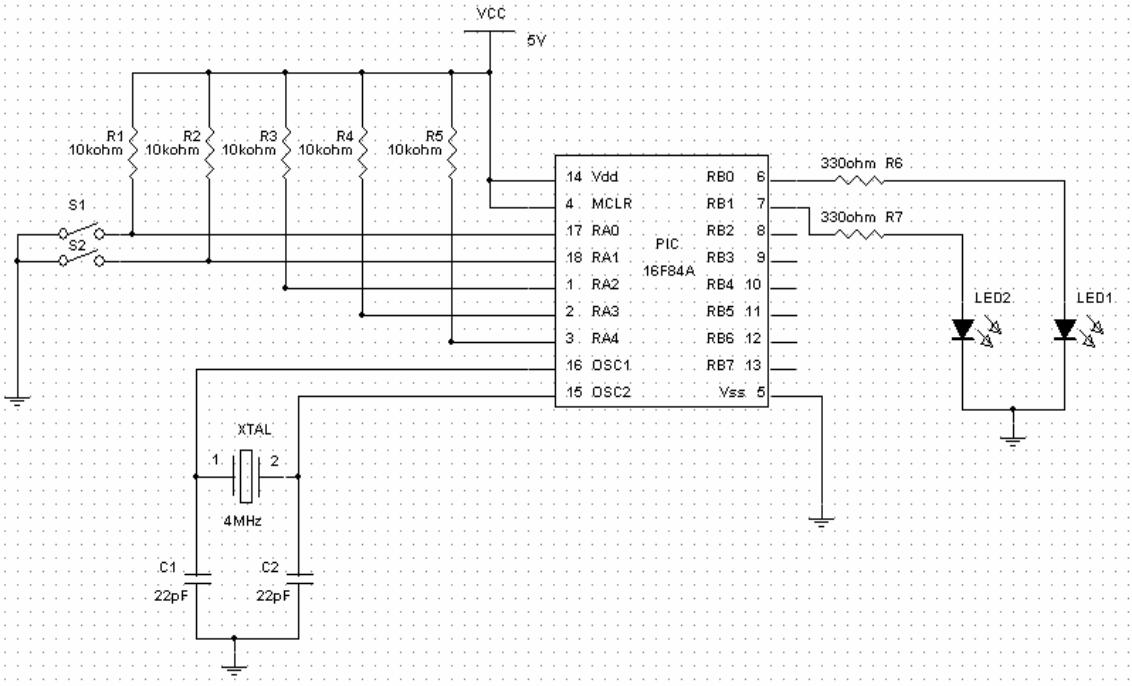


Figura 4.5. Circuito de la Práctica 4

2. Lea el siguiente enunciado.
   "El led 1 se encenderá cuando S1 y S2 tengan el mismo estado (es decir los dos en alto o los dos en bajo) y el led 2 encenderá cuando tengan estados diferentes."
3. Dibuje el diagrama de flujo que resuelva este problema
4. Escriba el programa correspondiente usando el set de instrucciones del PIC.
5. Compile el programa
6. Grabe el PIC
7. Compruebe el funcionamiento del circuito.
8. De sus conclusiones.

3.5.- Práctica 5:

"SUBRUTINAS DE TIEMPO".

Objetivo.

- Comprender que es una subrutina y la forma en que puede utilizarse para controlar el tiempo en que tarda un proceso en ejecutarse.

Introducción.

Subrutinas:

Cuando un grupo de instrucciones son ejecutadas en diferentes partes de un programa es conveniente el uso de subrutinas para ahorrar espacio en la memoria. Ya que si se escribieran en cada parte del programa este se haría muy largo y por tanto el espacio de memoria utilizado es mayor.

Se define una **subrutina** como un conjunto de instrucciones al cual se puede acceder desde cualquier punto del programa principal, es decir es un subprograma que se ejecuta cada vez que el programa principal lo llama.

Para poder llamar a una subrutina se utiliza la instrucción ***call etiqueta*.**
Una vez que se realizaron las acciones de la subrutina esta debe volver al punto inmediatamente después de la instrucción call. Esto se logra mediante la instrucción ***return*** (retorno de subrutina). Es importante mencionar que todas las subrutinas deben finalizar con esta instrucción

La pila es una zona de memoria que se encuentra separada tanto de la memoria de programa como de la de datos dentro del microcontrolador. Es aquí en donde se carga el valor del contador de programa (CP) cuando se ejecuta una instrucción ***call*** para que cuando se ejecute una instrucción ***return*** retome este valor y continué la ejecución del programa en el punto inmediatamente después de la instrucción ***call*.**

Subrutinas de tiempo:

En la mayoría de las aplicaciones es necesario controlar el tiempo en que tarda en ejecutarse un proceso.
La velocidad con la que el PIC ejecuta un programa depende del oscilador que este conectado al microcontrolador y del número de ciclos maquina ejecutados.

Se define como **ciclo maquina** a la unidad básica de tiempo que utiliza el microcontrolador para ejecutar una instrucción y corresponde a 4 ciclos de reloj. En la figura 5.1 pueden observarse los 4 ciclos de reloj que corresponden con el significado de un ciclo maquina.

Para nuestro caso utilizamos un cristal de cuarzo de 4MHz por lo que el tiempo que tarda un ciclo de reloj es de 590ns por lo tanto un ciclo maquina tarda $1\mu s$ en ejecutarse.
Todas las instrucciones del PIC16F84A tardan un ciclo maquina en ejecutarse, con excepción de las instrucciones de salto que tardan 2 ciclos maquina en ejecutarse. Debido a esto las instrucciones de salto tardan $2\mu s$ en ejecutarse y las demás instrucciones tardan $1\mu s$. De lo anterior puede deducirse que el tiempo que tarda en ejecutarse cada instrucción viene dado por la ecuación $T = \frac{4cm}{f}$ donde:

T: es el tiempo que tarda en ejecutarse una instrucción
cm: el número de ciclos maquina que consume la instrucción
f : La frecuencia del cristal utilizado.

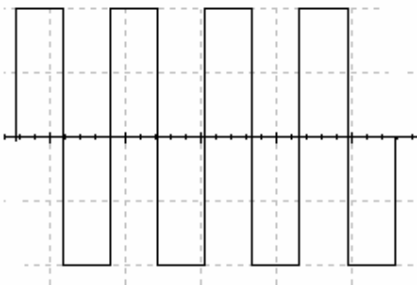


Figura 5.1. Ciclo maquina

La instrucción ***nop*** no realiza ninguna operación, solo consume un ciclo maquina sin hacer nada. Se utiliza para hace gastar tiempo al microprocesador sin alterar ningún registro.

A continuación expondremos una técnica para generar tiempos de espera utilizando bucles de lazo cerrado finito.

Esta técnica consiste en la repetición de algunas instrucciones tantas veces como sea necesario hasta conseguir el tiempo que requerimos. Estas instrucciones se agrupan en una subrutina denominada subrutina de tiempo. La estructura de la Figura 5.2 muestra el formato general para un programa que utiliza una subrutina de tiempo. La Figura5.2.A muestra la estructura básica de una subrutina de tiempo para tiempos de μs, nótese que la instrucción ***carga el contador con el valor inicial*** se ejecuta después de una llamada ***call tiempo***, una vez finalizada la ejecución de la

subrutina se regresa a ejecutar la siguiente instrucción del programa fuente utilizando la instrucción ***return***.
Debido a que el tiempo en que tarda en ejecutarse una instrucción es conocido lo único que se necesita saber es el valor de "z" que es el valor inicial con el cual se va a cargar el contador y corresponde al número de iteraciones en el bucle para obtener el tiempo deseado.

El valor inicial de "z" cuando se utiliza un cristal de 4MHz viene dado por la ecuación: $z = \frac{T-5}{4}$

Donde:
z: es el valor inicial con el que se cargara el contador
T: es el tiempo deseado en μs

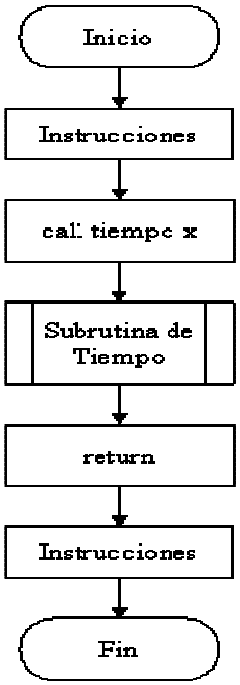


Figura 5.2. Digrama de flujo de un programa que utiliza una subrutina de tiempo

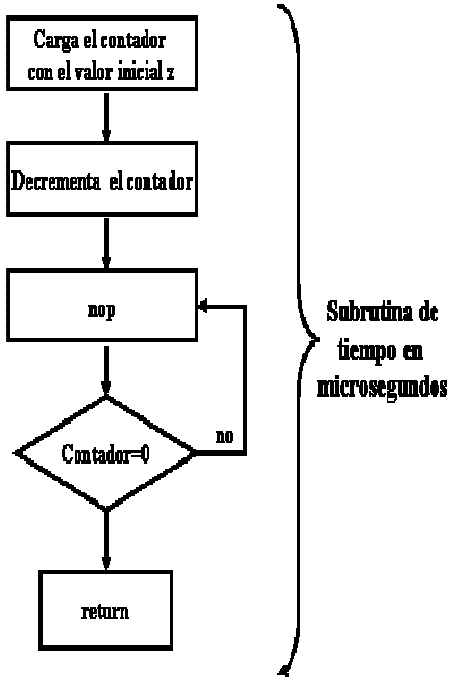


Figura 5.2.A. Diagrama de flujo para una subrutina de tiempo para tiempos de μs

Así por ejemplo si queremos un tiempo de 500 μs el valor inicial de z debe de ser

$$z = \frac{T-5}{4} = \frac{500-5}{4} = 123.75$$

Por lo que se toma el valor de 123 teniéndose un tiempo real de:

T=5+4z= 5+(4)(123)=4999997 μs =497 μs

Por lo que para completar los 500 μs se utilizan 3 instrucciones nop al principio de la subrutina quedando finalmente como sigue:

```
T-500 µs        ; la instrucción call nos trajo a esta subrutina y consume 2cm
        movlw   d'123'  ;se carga con este valor que corresponde a 5s (consume 1cm)
        movwf   contador        ;se carga el contador con el valor anterior (1cm)
        nop                     ;nop utilizado para conseguir los 5s exactos (1cm)
        nop                     ; nop utilizado para conseguir los 5s exactos (1cm)
        nop                     ; nop utilizado para conseguir los 5s exactos (1cm)
bucle   nop             ;nop  que esta dentro de la estructura de tiempo (1cm*123)
        decfsz  contador,F  ;decrementa contador y guarda el resultado en F, si es
                            ; 0 salta (1cm*122 cuando no salta y 2cm cuando salta)
        goto    bucle       ;si contador no es cero ve a la etiqueta bucle (2cm*122)
        return          ;si el contador es cero ya pasaron los 5s, por lo que regresa al
                            ; programa fuente (2cm)
```

Viendo el fragmento de programa anterior se deduce que los ciclos maquina empleados por esta subrutina son: 2+1+1+1+1+1+123+122+2+244+2=500cm y como cada cm dura 1 μs esta subrutina tarda efectivamente 500 μs en ejecutarse.

Cuando se necesita crear tiempos de mayor duración (segundos) es necesario emplear subrutinas de tiempo con bucles anidados los cuales contienen lazos de tiempo uno dentro de otro. La estructura general se muestra a continuación con un fragmento de programa para un tiempo de 5 segundos.

```
T_5s    ; la instrucción call nos trajo a esta subrutina y consume 2cm
        movlw d'50'         ; Aporta 1 ciclo máquina. Este es el valor de "x".
        goto    zor         ; Aporta 2 ciclos máquina.
zor
        movwf contador_3    ; Aporta 1 ciclo máquina.
zor_BucleExterno2
        movlw d'100'        ; Aporta x*1 ciclos máquina. Este es el valor de "y".
        movwf contador_2            ; Aporta x*1 ciclos máquina.
zor_BucleExterno
        movlw d'249'        ; Aporta x*y*1 ciclos máquina. Este es el valor de "z".
        movwf contador_1    ; Aporta x*y*1  ciclos máquina.
zor_BucleInterno
        nop                 ; Aporta  x*y*z*1  ciclos máquina.
        decfsz  contador_1,F    ; (z-1)* x*y*1  cm (si no salta) + x*y*2 cm (al saltar).
        goto    zor_BucleInterno    ; Aporta (z-1) *x*y*2 ciclos máquina.
        decfsz  contador_2,F    ; (y-1)*x*1 cm (cuando no salta) + x*2 cm (al saltar).
        goto    zor_BucleExterno    ; Aporta (y-1)*x*2 ciclos máquina.
```

```
decfsz  contador_3,F        ; (x-1)*1 cm (cuando no salta) + 2 cm (al saltar).
goto    zor_BucleExterno2   ; Aporta (x-1)*2 ciclos máquina.
return                      ; El salto del retorno aporta 2 ciclos máquina.
```

Por lo que esta subrutina consume 2+1+2+1+(*x*1*)+ (*x*1*)+ (*x*y*1*)+ (*x*y*1*)+ (*x*y*z*1*)+(*z*-1) (*x*y*1*)+ (*x*y*2*) +(*z*-1) (*x*y*2*)+ (*y*-1) (*x*1*)+*x*2*+ (*y-1*)(*x*2*)+ (*x*-1)*1 + 2 +(*x*-1)*2+2, por lo que reduciendo esta ecuación queda: cm=4x+4xy+4xyz+7.

Es importante señalar que los contadores se deben de declarar al principio del programa asignándole espacios de memoria RAM de datos. Estos se pueden direccional a partir de la posición 0x0C (13 decimal) que es la posición de memoria destinada a los registros generales. Esto se hace utilizando la instrucción EQU. Por ejemplo para los tres contadores utilizados se debe declarar de la siguiente forma.

```
contador_1   EQU   0x0C
contador_2   EQU   0x0D
contador_3   EQU   0x0E
```

El diagrama de flujo que representa a esta subrutina de tiempo se muestra en la Figura 5.3. Esta forma general nos permite crear tiempos desde 500ms hasta n segundos. Para tiempos desde 1ms hasta 500ms se puede omitir la parte del contador_3 tal como se muestra en la Figura 5.4.

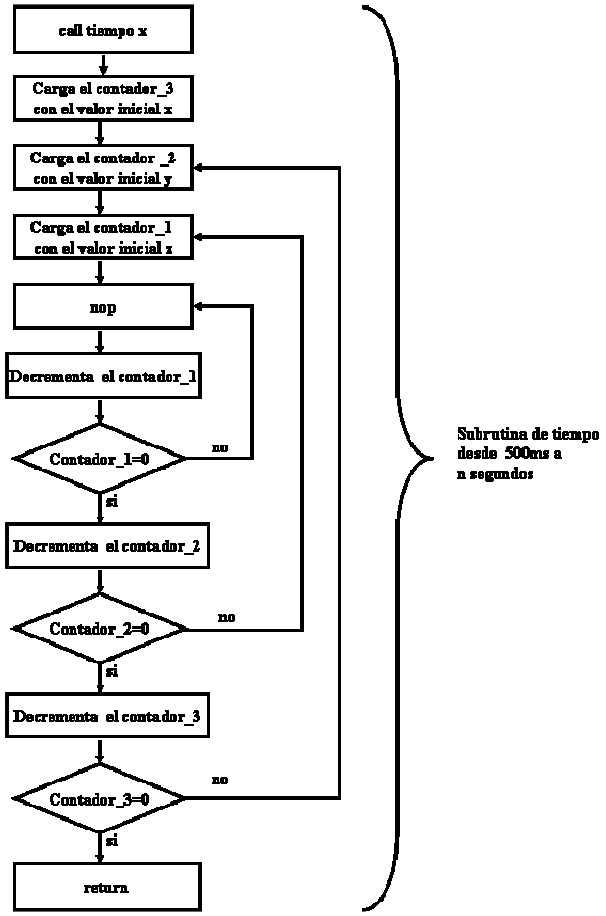


Figura 5.3. Subrutina de tiempo de 500 ms a n segundos.

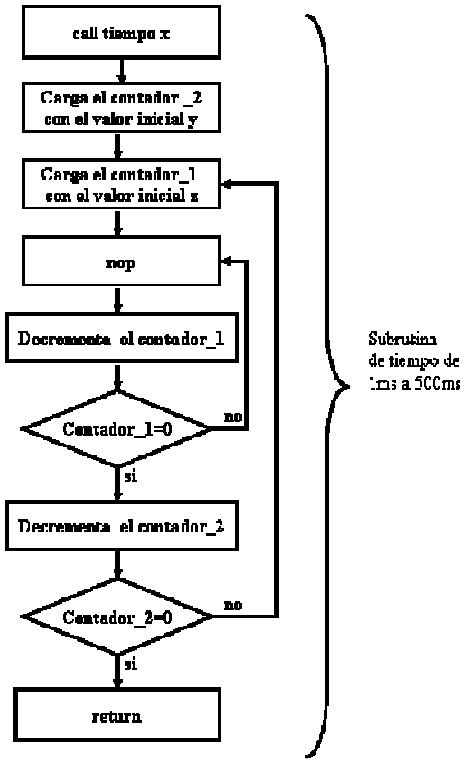


Figura 5.4. Subrutina de tiempo de1ms a 500 ms.

Material y equipo.

1-Grabador de PIC's
1-PIC 16F84A
1-Cristal de cuarzo de 4MHz
2-Capacitores de 22pF
1-Diodos led
1-resistencias de 330 Ω
1-PC
1-Fuente de alimentación

Desarrollo.

1. Conecte un led a una de las terminales del puerto B y haga que el led encienda por un segundo, luego que se apague por un segundo, que vuelva a encender 3 segundos, que se apague 3 segundos, luego que se encienda 8 segundos y se apague 8 segundos y vuelva a comenzar el ciclo.
2. Realizar un diagrama de flujo que muestre la solución del problema planteado.
3. Realizar el programa en el lenguaje ensamblador utilizando el set de instrucciones del microcontrolador interpretando el diagrama de flujo.
4. Compilar y programar el microcontrolador.
5. Alambrar el circuito físico que realiza la función para la cual fue diseñado y se planteo en el paso 1.
6. Compruebe el funcionamiento del circuito. En caso de que no cumpla con el objetivo para el cual fue hecho repetir el procedimiento hasta que se cumpla el objetivo.
7. De sus conclusiones

3.6.- Práctica 6:

"COMUNICACIÓN SERIE".

Objetivos.

- Conectar al microcontrolador con otros dispositivos utilizando comunicación serie a través del BUS 12C.

Introducción.

Existen 2 formas en la cual un microcontrolador y un microprocesador se pueden comunicar con dispositivos periféricos: La *comunicación paralelo* y la *comunicación serie*.

En la *comunicación en paralelo* existe transferencia de información (generalmente 8 bits) utilizando 8 terminales (una terminal para cada bit) del micro a la vez tal como se ve en la Figura 6.1 Además de las terminales de comunicación se necesita una terminal de reloj que sincroniza la transferencia de datos (un byte tras otro con cada ciclo de reloj) del micro a los periféricos. Esta característica hace que la comunicación paralelo sea muy rápida sin embargo al utilizar varias terminales del micro hace que el circuito se vuelva muy grande e impide conectar un numero mayor de dispositivos a las terminales del micro.

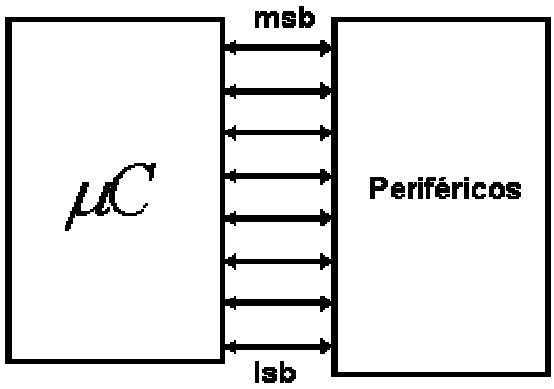


Figura 6.1. Comunicación paralelo.

La *comunicación serie* se da cuando la transferencia de información (un byte) se da un bit tras otro con cada ciclo de reloj utilizando solo una de las terminales del micro (Figura 6.2) para la transferencia de información y otra terminal de reloj que sincroniza la transferencia de información. Este tipo de comunicación al utilizar solo una de las terminales del micro reduce el tamaño del circuito así como el costo y la complejidad del mismo además de que permite conectar varios dispositivos en sus

terminales comunicándose con ellos a través de un protocolo de comunicación. Sin embargo una de sus principales desventajas es que la transferencia de datos no es tan rápida como la comunicación paralelo, sin embargo en la mayoría de los sistemas (un sistema consta de un microcontrolador y uno o mas dispositivos periféricos como memorias, teclados, etc.) la tasa de transferencia de datos que se requiere no es muy alta por lo que la transmisión de datos serie es útil.

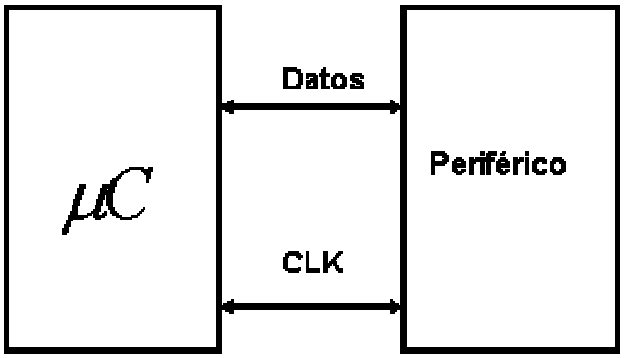


Figura 6.2. Comunicación serie

El bus 12C

Este Bus serie fue desarrollado por la empresa Philips semiconductors, esta formado por dos hilos, en los cuales se pueden conectar varios dispositivos periféricos tal como se muestra en la Figura 6.3. A través de los dos hilos se realiza la transmisión serie de datos, bit a bit.

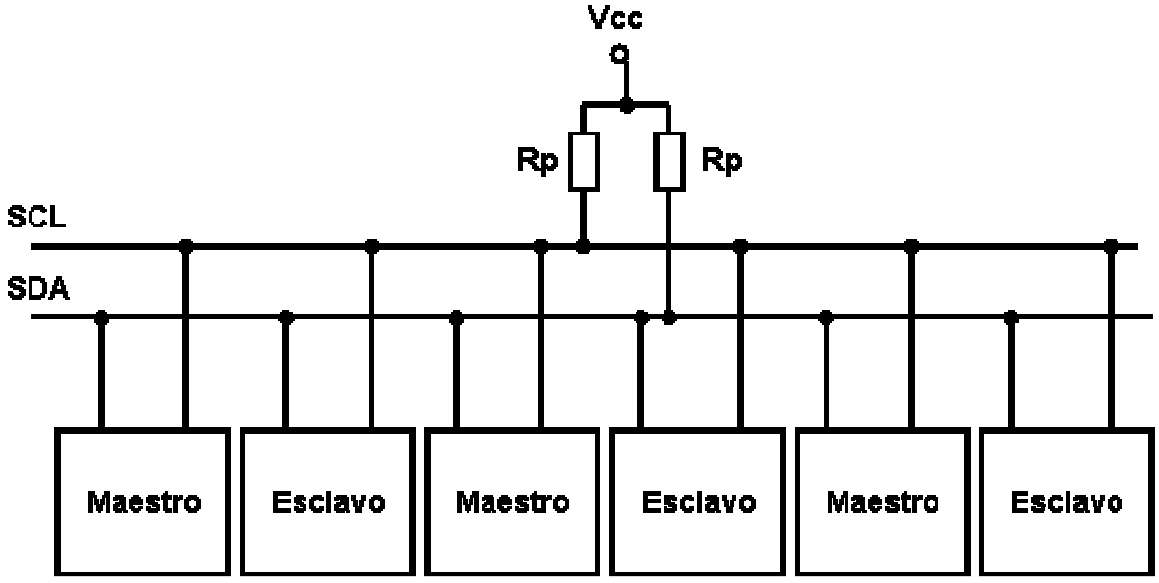


Figura 6.3.Conexión al bus 12C

Las resistencias Rp se calculan a base de unas tablas del fabricante Philips semiconductors las cuales se encuentran es su pagina de Internet, sin embargo una resistencia de $4.7K\Omega$ es apta para la mayoría de las aplicaciones.

Por las líneas SCL y SDA se transmiten dos señales:

- **SCL (serial clock line)**. Por esta línea se transmite la señal de reloj que se utiliza para la sincronización de los datos.
- **SDA (serial data line).** Por esta línea se realiza la transferencia serie de datos. Esta línea es bidireccional por lo que se pueden enviar y recibir datos entre los dispositivos.

El Protocolo de comunicaciones que utilizan los dispositivos conectados a un bus 12C es del tipo maestro/esclavo.

- El maestro inicia y termina la transferencia de información además de que controla la señal de reloj. Generalmente es un microcontrolador.
- El esclavo es direccionado por el maestro.

Cuando se conecta más de un dispositivo al bus 12C estos deben de tener una dirección única que los diferencia del resto. Los dispositivos compatibles con este bus suelen llevar 3 terminales que permiten modificar su dirección y hacerlo único en el bus. También es posible conectar más de un dispositivo que controle el bus, a estos circuitos se les llama multimaster.

La velocidad máxima de transferencia de datos por el bus 12C es de 100Kbits por segundo.

Transferencia de datos por el bus 12C

Para poder transferir un bit por la línea SDA, debe ser generado un pulso de reloj por la línea SCL. Los bits de datos transferidos por la línea SDA no cambian mientras SCL este en alto. El estado de la línea SDA solo puede cambiar cuando la línea SCL esta en estado bajo tal como se muestra en la Figura 6.4.

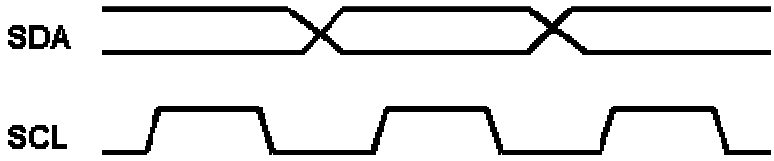


Figura 6.4. Transmisión de un bit por el bus 12C.

Para poder comenzar la transmisión de datos se debe comprobar que el bus no este ocupado y para hacer esto tanto la señal SDA y SCL deben de estar en un nivel alto (condición de START). Después de que se comprueba que el bus no esta ocupado comienza la transmisión de los bits con cada pulso de reloj (SCL). Para indicar que la transferencia de datos ha terminado la línea SDA pasa a estado alto mientras que la línea SCL permanece en estado alto (condición de STOP). Este proceso se muestra en la Figura 6.5.

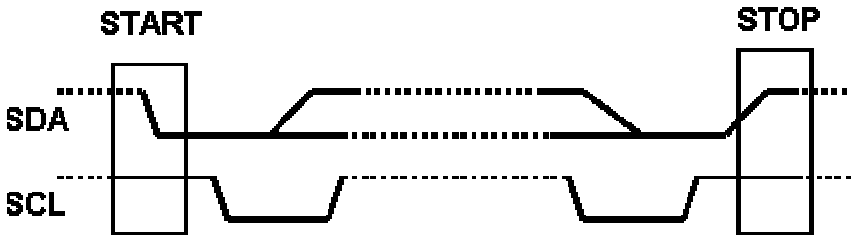


Figura 6.5. Inicio y fin de la transmisión de datos.

Cada dato a transmitir debe de tener 8 bits comenzando por el más significativo (MSB) y terminando por el menos significativo (LSB). Cuando se finaliza de transmitir los 8 bits el receptor manda un bit de reconocimiento llamado ACK en el noveno pulso de reloj poniendo la señal de SDA en un nivel bajo. Este bit es obligatorio cada que finaliza la transferencia de un dato.

Formato de una transferencia de datos.

El formato que deben de tener los datos a transferir llevan la siguiente secuencia (Figura 6.6):

1. Un bit de START, que señaliza el comienzo de la transmisión de datos.
2. Siete bits de direccionamiento de un esclavo
3. Un bit de lectura y escritura (R/W) que define si el esclavo es transmisor o receptor.
4. Un bit de reconocimiento ACK.
5. mensaje de ocho bits seguido de un bit ACK.
6. Un bit de STOP que indica el fin de la comunicación.

Start	1-7	8	9	1-8	9	1-8	9	Stop
	Dirección	R/W	ACK	Dato	ACK	Dato	ACK	

Figura 6.6. Formato de la transferencia de datos por el bus 12C

Puede haber varios esclavos conectados al bus, por lo que el maestro debe de indicar la dirección del esclavo al cual hace referencia. Cada esclavo tiene una dirección única. Los siete primeros bits del primer byte indican la dirección del esclavo, el octavo bit de lectura y escritura (R/W) determina la dirección de los datos.

- Si R/W=0, el esclavo es el receptor; es decir el maestro escribira información en el esclavo seleccionado.
- Si R/W=1 el esclavo es emisor; es decirle maestro leera información del esclavo.

Conexión de bus 12C a un PIC 16F84A

En la Figura 6.7 se muestra la forma de conectar un bus 12C al PIC 16F84 utilizando las líneas RA3 (SCL) y RA4 (SDA) como líneas del bus 12C.

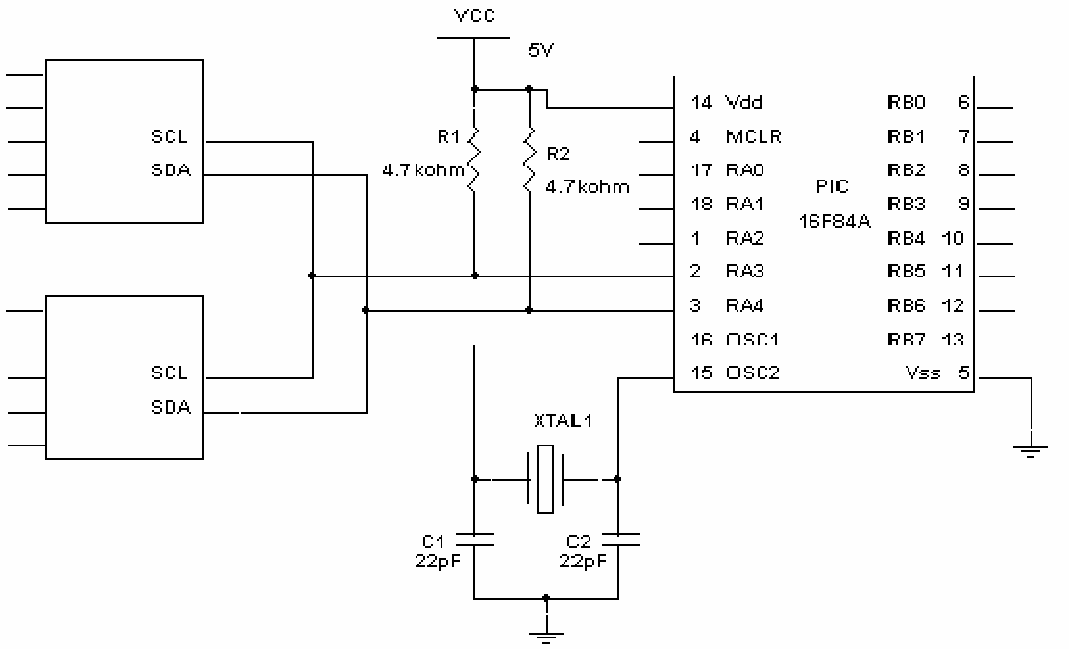


Figura 6.7. Conexión de un bus 12C al PIC 16F84

Para que el sistema funcione correctamente, el programa del microcontrolador debe de permitir la comunicación serie mediante el protocolo del bus 12C. Todo el protocolo se ha explicado ya y se encuentra resumido en la subrutina del Apéndice II.

Dispositivos compatibles con el bus 12C

Existen en el mercado una gran cantidad de dispositivos compatibles con el bus 12C. Algunos ejemplos de estos son:

- PCF8574, interfase de puerto de 8 líneas a bus 12C.
- 24LC256, Memoria EEPROM serie de 32 kbytes
- DS1624, termómetro digital.
- PCF8591, Conversor ADC de cuatro canales de 8 bits mas un canal DAC.
- Etc.

En esta práctica utilizaremos el PCF8574 (Figura 6.8) ya que es un dispositivo que convierte un dato paralelo de 8 bits en dato serie y viceversa. Solo explicaremos sus características más importantes ya que toda la información referente a este dispositivo se encuentra en su hoja de especificaciones.

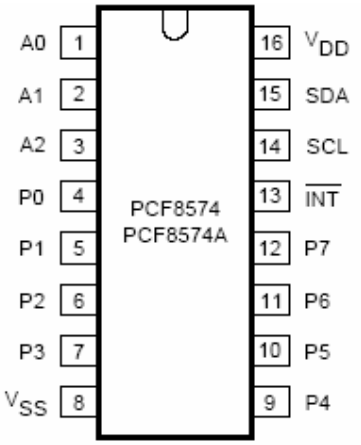


Figura 6.8. El PCF8574

Este dispositivo cuenta con un puerto de entrada/salida (P0-P7), tres terminales que permiten direccionarlo (A0-A2), las líneas SDA y SCL que permiten la comunicación serie y una terminal de interrupción ($\overline{INT}$) que se puede conectar a la entrada de interrupción del micro la cual genera un pulso bajo cuando detecta un cambio de estado en cualquiera de las líneas del puerto (P0-P7), con esto el micro puede leer el nuevo dato por el Bus 12C. Esta señal solo puede ser generada cuando el puerto del PCF8574 es configurado como entrada.

El procedimiento de lectura y escritura del dispositivo sigue el protocolo del bus 12C ya explicado y se resume en la subrutina del Apéndice III.

Material y equipo.

Grabador de PIC's
PIC 16F84A
2-PCF8574 (conversor bus paralelo-bus 12C)
Cristal de cuarzo de 4MHz
2-Capacitores de 22pF
2-Dipswitch de 5 interruptores cada uno
8-Diodos led
8-resistencias de 330 Ω
2-resistencias de 4.7 *k*Ω
1-PC
1-Fuente de alimentación

Desarrollo.

1. Lea el siguiente enunciado: “Haga un programa para comprobar el funcionamiento del PCF8574 que es un expansor de bus I2C.Utiliza dos circuitos integrados tal como se indica en el esquema de la Figura 6.9 Uno como entrada (su pin A0 se conecta a masa), leyendo 8 interruptores Otro como salida (Su pin A0 se conecta a Vcc), visualizando los datos por un arreglo de diodos LEDs.”
2. Realizar un diagrama de flujo que muestre la solución del problema planteado.
3. Realizar el programa en el lenguaje ensamblador utilizando el set de instrucciones del microcontrolador interpretando el diagrama de flujo. Sugerencia: utilice las subrutinas que se proporcionan en los Apéndices II y III
4. Compilar y programar el microcontrolador.
5. Arme el circuito de la Figura 6.9.
6. Compruebe el funcionamiento del circuito. En caso de que no cumpla con el objetivo para el cual fue hecho repetir el procedimiento hasta que se cumpla el objetivo.
7. De sus conclusiones

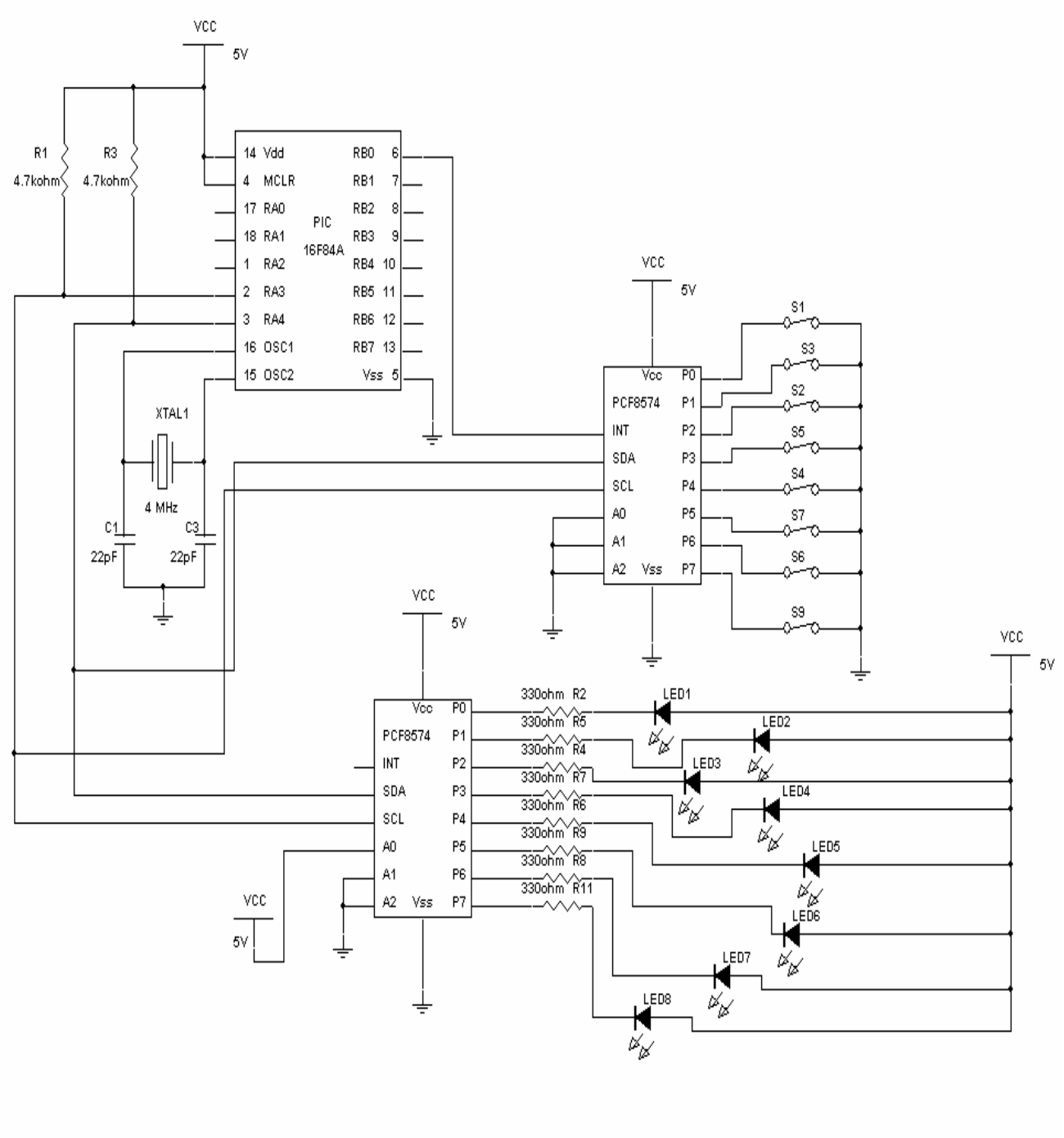


Figura 6.9. Circuito de la práctica 6.

3.7.- Práctica 7:

"INTERRUPCIONES".

Objetivos.

- Comprender que es una interrupción y los distintos tipos de estas para poder utilizarlas en el diseño de sistemas con microcontroladores.

Introducción.

Se produce una **interrupción** cuando en algún punto del programa este deja de ejecutarse, pasando a ejecutar una subrutina de interrupción y al terminar de ejecutarla regresa al punto inmediatamente después de que se produjo la interrupción ejecutando el resto del programa. Las interrupciones pueden ser causadas por eventos internos o externos.
Las interrupciones permiten sincronizar la ejecución del programa con lo que ocurre con los periféricos externos conectados al microcontrolador. Esto es muy útil sobre todo cuando es necesario atender eventos que son de muy corta duración como por ejemplo un pulsador en el que el cambio de estado dura por un tiempo muy cortó o eventos que están cambiando constantemente de estado como la detección de pulsos. Por lo que las interrupciones son la forma más importante de conectar dispositivos externos al microcontrolador.

Las interrupciones funcionan de una manera muy similar a las subrutinas con la diferencia que una subrutina es ejecutada cuando se presenta una instrucción call, mientras que una interrupción es ejecutada en cualquier instante debido a que a ocurrido un evento externo o interno.
La estructura que muestra el funcionamiento de una interrupción se muestra en la Figura 7.1

Los PIC16F84 tienen 4 causas o fuentes posibles de interrupción:

- Activación del pin RB0/INT.
- Desbordamiento del temporizador TMR0
- Cambio de estado en una de los 4 pines de más peso (RB7-RB4) del Puerto B
- Finalización de la escritura en la EEPROM de datos

Cuando ocurre cualquiera de los 4 sucesos indicados se origina una petición de interrupción, que si se acepta y se atiende comienza depositando el valor del contador de programa actual en la Pila, poniendo el bit GIE = 0 y cargando en el contador de programa el valor 0004 H, que

es el Vector de Interrupción a donde se ejecuta la subrutina de interrupción.

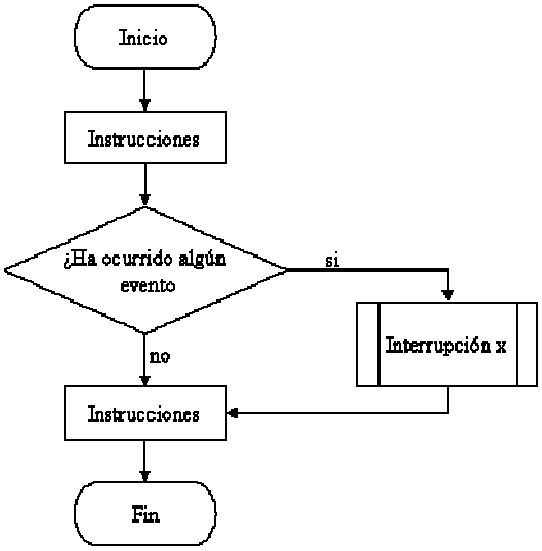


Figura 7.1. Diagrama de flujo de un programa con una interrupción.

Cada fuente de interrupción dispone de un señalizado o "flag", que es un bit que se pone automáticamente a 1 cuando se produce. Además cada fuente de interrupción tiene otro bit de permiso, que según su valor permite o prohíbe la realización de dicha interrupción. Estos bits se encuentran en el registro INTCON.

*El Registro de Control de Interrupciones INTCON***:**

La mayor parte de los flags y bits de permiso de las fuentes de interrupción en los PIC16F84 están implementados sobre los 8 bits del registro INTCON, que ocupa la dirección 0B H del banco 0, hallándose duplicado en el banco 1. Los bits del registro INTCON se muestran en la Tabla 7.1.

Cada uno de los bits o flags se describen a continuación:

GIE: Permiso Global de Interrupciones

Se pone a "0" automáticamente cuando detecta una interrupción para asegurarse que no ocurrirá ninguna otra interrupción mientras se atiende a la primera. Al retornar de la interrupción mediante la instrucción **retfie** se vuelve a activar automáticamente poniéndose a "1"

0: Prohíbe todas las interrupciones.

1: Permite la ejecución de todas las interrupciones, cuyos bits de permiso individuales también las permitan.

GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Tabla 7.1. Bits del registro INTCON

EEIE: Permiso de Interrupción por fin de escritura en la EEPROM
0: Prohíbe que se produzca esta interrupción.
1: Permite que se origine esta interrupción cuando termina la escritura en la EEPROM de datos.

T0IE: Permiso de Interrupción por desbordamiento del TMR0
0: Prohíbe esta interrupción.
1: Permite una interrupción al desbordarse el TMR0.

INTE: Permiso de Interrupción por activación de la terminal RB0/INT
0: Prohíbe esta interrupción.
1: Permite la interrupción al activarse RB0/INT.

RBIE: Permiso de Interrupción por cambio de estado en los pines RB7-RB4
0: Prohíbe esta interrupción
1: Permite esta interrupción.

T0IF: Señalizador de desbordamiento del TMR0
0: Indica que el TMR0 no se ha desbordado.
1: Toma este valor cuando ha ocurrido el desbordamiento.

INTF: Señalizador de activación del pin RB0/INT
0: Indica que RB0/INT aún no se ha activado.
1: Se pone a 1 al activarse RB0/INT.

RBIF: Señalizador de cambio de estado en los pines RB7-RB4
0: No ha cambiado el estado de RB7-RB4.
1: Pasa a 1 cuando cambia el estado de alguna de esas líneas.

Para conocer que causa ha provocado la interrupción se exploran los señalizadores, tres de los cuales se ubican en el registro INTCON y el cuarto, EEIF, que se pone a 1 cuando finaliza la escritura en le EEPROM, se halla en el bit 4 del registro EECON1. Los señalizadores deben ponerse a 0 por programa antes del retorno de la interrupción y son operativos aunque la interrupción esté prohibida con su bit de permiso correspondiente.

Interrupción Externa INT:

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

Mediante el bit 6, llamado INTDEG, del registro OPTION se puede seleccionar cual será el flanco activo en RB0/INT. Si se desea que sea ascendente se escribe un 1 en dicho bit, y si se desea descendente se escribe un 0. El procesador explora el señalizador INTF al final del primer ciclo de reloj de cada ciclo maquina. Recuérdese que cada ciclo maquina constaba de 4 ciclos de reloj. Al terminar el primer ciclo de reloj se exploran los señalizadores produciéndose un período de espera de 3 ó 4 ciclos máquina desde el momento que hay un señalizador activado hasta que se inicializa la interrupción.

Interrupción por desbordamiento del TMR0:

Cuando el TMR0 se desborda y pasa del valor FF H al 00 H, el señalizador T0IF se pone automáticamente a 1. Sí además, el bit de permiso de interrupción del TMR0 T0IE=1 y el bit de Permiso Global de Interrupciones GIE=1, se produce una interrupción.

Instrucción "RETFIE":

La instrucción ***retfie*** es utilizada al final de la subrutina de interrupción y es idéntica a la instrucción RETURN (retorno de subrutina). Además coloca de manera automática el bit GIE a "1" volviendo a habilitar las interrupciones. Al terminar de ejecutar la subrutina de interrupción el programa continúa su ejecución inmediatamente después de donde fue interrumpido.

Registros alterados por una interrupción:

Cuando se ejecuta la subrutina de interrupción es posible que esta pueda modificar el contenido de los registros del microcontrolador y al momento de reanudar el programa después de la subrutina de interrupción el programa trabaje con datos que no son los mismos a los que tenían los registros antes de la interrupción y por tanto el programa no funcionara correctamente. Por esta razón es necesario guardar el contenido antes de ejecutar la subrutina de interrupción y restaurarlos antes de la ejecución de la instrucción ***retfie*.**

Detectar la causa de una interrupción:

Cuando se produce una interrupción el contador de programa se carga con la dirección 004h sea cual sea la causa de la interrupción por lo que es necesario verificar el estado de los flags de interrupción para detectar que es lo que esta causando una interrupción.
A continuación se muestra el fragmento de programa que permite detectar la causa de la interrupción:

```
----------
btfsc  INTCON,INTF    ; ¿la interrupción es causada por la activación del pin
                      ;RB0/INT?
call   inter_INT      ;ejecuta la subrutina de interrupción inter_INT
btfsc  INTCON,RBIE    ; ¿la interrupción es causada por el cambio de estado
                      ; en el puerto B?
call     inter_RBI    ;ejecuta la subrutina de interrupción inter_INT
btfsc    INTCON,TOIF  ; ¿la interrupción es causada por el desbordamiento
                      ;del TMR0?
call     inter_TOI    ;ejecuta la subrutina de interrupción inter_TOI
------------
```

Pasos a seguir para la realización de programas con interrupciones:

1. Habilitar las interrupciones correspondientes
   Por ejemplo para habilitar todas las interrupciones con excepción de la interrupción por fin de escritura en la EEPROM se utilizan las siguientes instrucciones.

```
movlw  b'10111000'
movwf  INTCON
```

   Con esta configuración se permite acceder a las interrupciones siempre que alguno de los bits de estas interrupciones (TOIF, INTF o RBIF) este a "1".

2. Cuando se produce una interrupción se produce un salto a la posición de memoria que contiene los datos de la subrutina de interrupción. Esta posición de memoria se debe declarar al principio del programa.

```
ORG    0
goto   Inicio
ORG    4
goto   sub_int
```

3. A continuación se deben guardar todos los registros que se pueden alterar debido a la subrutina de interrupción. Esto puede hacer como se explica en el fragmento de programa llamado subrutina de interrupción.

```
 ; subrutina de interrupción
; Con este procedimiento se logra guardar el valor de los registros STATUS, W
; y otros registros
; Que resultan alterados por la subrutina de interrupción.

    CBLOCK
    Guarda_w
    Guarda_STATUS
    Guarda_RegistroA
    Guarda_RegistroB
    -----------------
    ENDC

; Se debe almacenar primero los valores de los registros w
; y STATUS antes de la interrupción
; No se puede usar la instrucción "mov  STATUS,W" por que se altera
; el bit z del registro STATUS

sub_int   movwf Guarda_W              ; Guarda W y STATUS.
          swapf   STATUS,W    ; Ya que "movf STATUS,W", altera el bit Z.
          movwf  Guarda_STATUS
          movf    RegistroA,W         ; Guarda el valor del RegistroA ya que la
          movwf  Guarda_RegistroA     ; subrutina lo altera.
          movf    RegistroB,W         ; Guarda el valor del RegistroB
         movwf   Guarda_RegistroB
          -------------------------
          bcf      STATUS,RP0         ; Para asegurarse que trabaja con el
                                      ;banco 0.
```

4. A continuación se debe de detectar la causa de la interrupción tal como ya se explico.
5. Dependiendo de la causa de interrupción se bifurca a la subrutina correspondiente.
6. Una vez finalizada la subrutina de interrupción se deben poner los registros tal como estaban antes de la interrupción.

```
 movf  Guarda_RegistroA,W      ; Restaura los registros utilizados en esta
movwf RegistroA                ; subrutina y también en la principal.
movf   Guarda_RegistroB,W
movwf Guarda_RegistroB
 swapf  Guarda_STATUS,W        ; Restaura el STATUS.
movwf STATUS
```

```
swapf  Guarda_W,F          ; Restaura W como estaba antes de producirse
swapf  Guarda_W,W            ; la  interrupción.
bcf    INTCON,INTF         ; borra el bit de señalización de la interrupción
                             ; por activación del pin RBO/INT
retfie                       ;regresa y rehabilita las interrupciones
```

7. El programa debe borrar los bits de señalización (INTF, RBIF, TOIF o EEIF) que indican las fuentes de las interrupciones antes de retomar el programa principal.

```
bcf   INCON,INTF      ; limpia los bits de señalización de interrupcion
bcf   INTCON,RBIF
bcf   INTCON,TOIF
retfie                ;regresa y rehabilita las interrupciones
```

Modo de bajo consumo o "SLEEP":

Muchas veces un microcontrolador esta esperando durante muchos intervalos de tiempo a que ocurra un evento externo. Durante estos periodos de "espera" es conveniente que el PIC entre en el modo de bajo consumo. Para lograr que el PIC entre en este estado se utiliza la instrucción ***sleep*** y cuando ocurre un suceso (una interrupción que no sea por el desbordamiento del TMR0 o un reset) el micro opera en condiciones normales. Por ejemplo cuando se espera una interrupción se hace que el micro entre en modo de bajo consumo y cuando ocurre la interrupción el micro sale del modo de bajo consumo atendiendo la interrupción correspondiente. La manera de lograrlo se muestra a continuación.

```
----------
Espera   sleep    ;el micro entra en modo de bajo consumo y espera una interrupción
         goto     Espera

Subrutina de interrupción   ; ha ocurrido una interrupción y el micro sale
                            ; del modo de bajo consumo
----------
```

Cuando el micro entra en modo de bajo consumo (si durante la ejecución del programa se encuentra la instrucción sleep) ocurre lo siguiente:

- El consumo de corriente del microcontrolador baja unos pocos de micro amperes.
- El oscilador principal del sistema deja de funcionar.
- Los puertos de entrada y salida mantienen el mismo estado que tenían antes de ejecutarse la instrucción ***sleep***.
- Los bits TO y PD del registro STATUS toman valores "1" y "0" respectivamente.

Material y equipo.

Grabador de PIC's
PIC 16F84A
Cristal de cuarzo de 4MHz
2-Capacitores de 22pF
5-microswitch
2-Diodos led
5-resistencias de 330 Ω
5-resistencias de 10 $k\Omega$
1-PC
1-Fuente de alimentación

Desarrollo.

1. Lea el siguiente enunciado: “Cada que ocurre un interrupción debida al cambio de estado en la terminales RB4-RB7 un led conectado en RA1 cambia de estado y cuando ocurre una interrupción por la activación de la terminal RB0 un led conectado en RA2 cambia de estado”.
2. Diseñar (dibujar) el circuito físico que controlara el programa Sugerencia: conecte las terminales RB4-RB7 directamente a cero para que cuando se oprima es microswitcht cambie el estado de “0” a “1”.
3. Realizar un diagrama de flujo que muestre la solución del problema planteado.
4. Realizar el programa en el lenguaje ensamblador utilizando el set de instrucciones del microcontrolador interpretando el diagrama de flujo.
5. Compilar y programar el microcontrolador.
6. Alambrar el circuito físico que realiza la función para la cual fue diseñado y se planteo en el paso 1.
7. Compruebe el funcionamiento del circuito. En caso de que no cumpla con el objetivo para el cual fue hecho repetir el procedimiento hasta que se cumpla el objetivo.
8. De sus conclusiones

Conclusiones y resultados del trabajo de titulación.

El presente trabajo entra dentro de la modalidad de desarrollo de un caso práctico por lo cual busca resolver un problema que se presenta y en este caso es la carencia de un manual de prácticas de la materia de microprocesadores y microcontroladores. Este trabajo esta pensado para que los profesores y alumnos del laboratorio de la asignatura propuesta en el nuevo plan de estudio de la carrera de ingeniería eléctrica electrónica, puedan usarlo como manual de prácticas base.

Este manual fue diseñado pretendiendo que los alumnos puedan entender el principio de funcionamiento de los microcontroladores con mayor facilidad, esto se pretende lograr mediante un proceso ordenado y secuencial de construcción de conocimiento. Lo anterior puede sonar muy ambicioso, sin embargo ante la situación actual de la enseñanza de dicho laboratorio es menester realizar cambios que permitan un mayor aprovechamiento, y uno de estos cambios es la implementación de un manual de prácticas para poder estandarizar la enseñanza de la materia, bajo el sistema de gestión de la calidad existente en los laboratorios.

Otro de los objetivos que se pretenden alcanzar es el que una vez realizadas y comprendidas todas las prácticas el alumno sea capaz de utilizar otros dispositivos mas avanzados y con mayores prestaciones (diferentes modelos u otras marcas) ya que las prácticas expuestas en el presente trabajo son la teoría básica común en todos los microcontroladores.

Se recomienda que la práctica 1 se lleve a cabo en dos sesiones de laboratorio debido a que es muy laboriosa aunque su grado de dificultad es sencillo. Las prácticas 2, 3 y 4 pueden realizarse en una hora cuando son realizadas por un experto (alambrado y programación), sin embargo para los alumnos noveles en la materia puede llevar hasta el triple de tiempo la realización de dichas prácticas por ello se sugiere recomendar al alumno llegar a la sesión de laboratorio (la cual dura hora y media) con el circuito alambrado para que pueda seguir los pasos subsecuentes de las prácticas y en su caso dedicarse de lleno a la realización de los programas ya que de lo contrario la realización de la práctica puede prolongarse una sesión más lo cual representaría un problema, siendo el alambrado el que se puede llevar una sesión lo cual no es el objetivo del las prácticas y por consecuencia no se alcanzarían a realizar todas. Las prácticas 5, 6 y 7 tienen un grado de complejidad mayor por lo que un experto puede tardar una sesión de laboratorio completa (hora y media) en la realización estas

(alambrado y programación), por lo que es recomendable para los alumnos dediquen dos sesiones de laboratorio para poder alcanzar los objetivos planteados satisfactoriamente siguiendo la recomendación de llevar el circuito alambrado a la sesión de la práctica. En la tabla de resultados se muestra a manera de resumen lo anterior

Tabla de resultados			
	Grado de dificultad	*# de sesiones*	*Aportaciones generales*
Práctica 1	Sencillo	2	Se propone un manual de prácticas, el cual permite entender de forma práctica los conceptos básicos de los microprocesadores y microcontroladores para poder diseñar sistemas con estos dispositivos y poder resolver problemas de ingeniería.
Práctica 2	Sencillo	1	
Práctica 3	Medio	1	
Práctica 4	Medio	1	
Práctica 5	Complejo	2	
Práctica 6	Complejo	2	
Practica 7	Complejo	2	

Se recomienda llevar el circuito de cada práctica armado a la sesión de laboratorio. Se espera que este material ayude a futuras generaciones de ingenieros eléctricos electrónicos en su formación académica. Es de vital importancia hacer entender a estas futuras generaciones que la teoría es necesaria para la realización de una práctica, esto debido a que son ingenieros y la ingeniería es una ciencia aplicada además de un arte, y como ciencia debe tener una metodología y un rigor científico teórico-práctico.

Existe una frase que me motivo durante toda mi carrera ya que aplica perfectamente a nosotros los ingenieros, la cual dice así:

“La ciencia es el capitán y la práctica el soldado… el hombre que se enamora de la práctica sin la ciencia es como el capitán de un barco sin brújula ni timón y por consiguiente no sabe a donde va”

Leonardo Da Vinci

Quiero finalizar este trabajo mencionando que mi estancia en la UNAM ha sido una de las etapas más hermosas de mi vida ya que en esta gran institución me formó no solo como ingeniero sino como persona además de que conocí a grandes personas que han marcado mi vida.

Apéndice I "El PIC 16F84A".

I.1 Generalidades del PIC 16F84A.

El PIC 16F84A es un microcontrolador de 8 bits diseñado por la empresa Microchip (http://www.microchip.com). Contiene una memoria RAM de 14 bits del tipo Flash y una memoria ROM del tipo EEPROM.

La memoria EEPROM y la Flash son eléctricamente gravables y borrables, lo que permite escribir y borrar el programa bajo prueba manteniendo el microcontrolador en el mismo zócalo y usando el mismo dispositivo para grabar y borrar.

Otra ventaja del PIC16F84 es su sistema de grabación de datos, que se realiza en serie. Para escribir un programa en la memoria se manda la información en serie usando sólo dos patillas, una para la señal de reloj y otra para los datos serie. A continuación se exponen las características más significativas de este microcontrolador:

MEMORIA DE PROGRAMA: Tipo ROM FLASH de 1 K x 14 bits
MEMORIA RAM DE DATOS: 68 bytes
MEMORIA EEPROM DE DATOS: 64 bytes
PILA (Stack): De 8 niveles
INTERRUPCIONES: 4 tipos diferentes
JUEGO DE INSTRUCCIONES: 35 con longitud de 14 bits
ENCAPSULADO: Plástico DIP de 18 patillas
FRECUENCIA DE TRABAJO: 10 MHz máxima
TEMPORIZADORES: Sólo uno el TMR0. También tiene Perro Guardián (WDT)
LINEAS DE E/S DIGITALES: 13 (5 Puerta A y 8 Puerta B)
VOLTAJE DE ALIMENTACION (VDD): De 2 a 6 V DC
VOLTAJE DE GRABACION (VPP): De 12 a 14 V DC
NUMERO DE TERMINALES: 18
PUERTOS DISPONIBLES: 2 (Puerto A y Puerto B)

I.2 Aspecto externo.

El PIC 16F84 tiene 18 terminales de conexión las cuales se muestran en la Figura I.1:

Estas terminales se pueden dividir en: Terminales de polarizacion, terminales del oscilador, terminal de RESET y terminales de los puertos. A continuación se describe cada una de ellas.

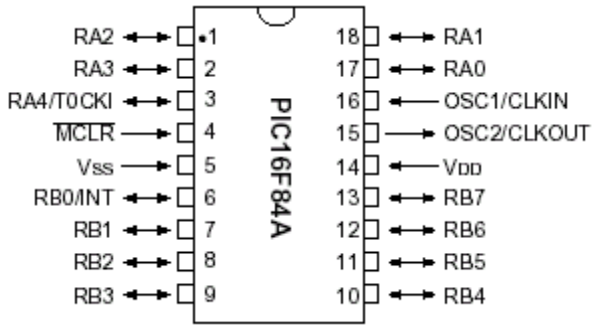


Figura I.1. El PIC 16F84.

Terminales de polarizacion del PIC:

El microcontrolador PIC16F84 se polariza con una tensión de 5V que se aplican entre las terminales VDD (pin 14) y VSS (pin 5) que son la alimentación y la masa respectivamente.

Terminales del Oscilador:

Todo microcontrolador necesita de un oscilador externo que le indique la velocidad de trabajo. Este oscilador genera una onda cuadrada de alta frecuencia que se utiliza como señal para sincronizar todas las operaciones del sistema.

EL PIC admite cuatro tipos de osciladores externos para aplicarles la frecuencia de funcionamiento, se colocan entre las patillas OSC1(terminal 16) y OSC2 (terminal 15). Los tipos que se pueden emplear son:

- *Oscilador XT:* Es un oscilador de cristal o resonador cerámico para frecuencias estándar comprendidas entre 100 KHz y 4 MHz. Es el oscilador mas utilizado y la forma de conexionarlo se muestra Figura I.2. Los capacitores C1 y C2 con alta capacidad sirven para incrementar la estabilidad del oscilador, sin embargo retrasan el comienzo de la ejecución de las instrucciones.
  En muchas aplicaciones se utiliza un cristal de 4 MHz acompañado de 2 capacitores de entre 15 y 33pF.

- *Oscilador RC:* Es un oscilador de bajo costo formado por una simple resistencia y un condensador. Proporciona una estabilidad mediocre de la frecuencia, cuyo valor depende de los valores de los dos elementos R-C. En la Figura I.3 se muestra la conexión típica de este oscilador.

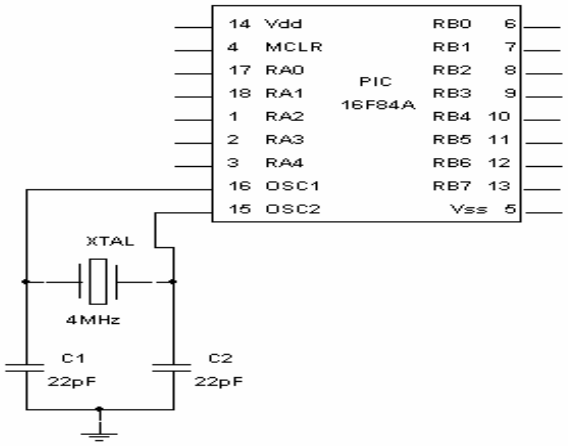


Figura I.2. Conexión de un oscilador XT al PIC16F84.

- *Oscilador HS:* Es un oscilador que alcanza una alta velocidad comprendida entre 4 y 10 MHz y está basado en un cristal de cuarzo o un resonador cerámico. La conexión es la misma que para un cristal de cuarzo
- *Oscilador LP:* Oscilador de bajo consumo con cristal o resonador diseñado para trabajar en un rango de frecuencias de 35 a 200 KHz. La conexión es la misma que para un cristal de cuarzo

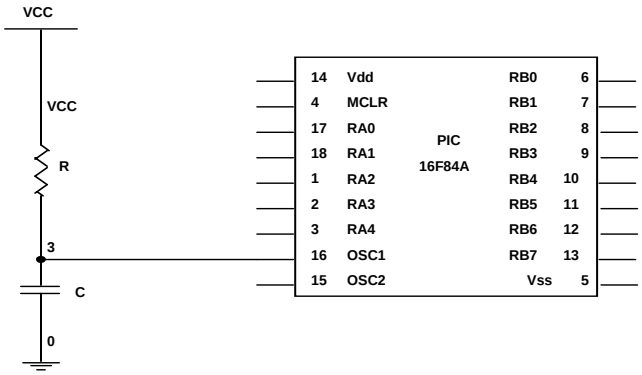


Figura I.3. Conexión de un oscilador RC al PIC16F84.

La frecuencia de trabajo del microcontrolador es un parámetro fundamental a la hora de establecer la velocidad de ejecución de instrucciones y el consumo de energía. Cuando un PIC16F84 funciona a 10 MHz, que es su máxima frecuencia, le corresponde un ciclo de instrucción de 400 ns, puesto que cada instrucción tarda en ejecutarse 4 períodos de reloj, o sea, 4 x 100 ns = 400 ns. Todas las instrucciones del PIC se realizan en un ciclo de instrucción o ciclo maquina, menos las de salto que tardan el doble de tiempo en ejecutarse.

Los impulsos de reloj entran por la patilla OSC1/CLKIN y se dividen por 4 internamente, dando lugar a las señales Q1, Q2, Q3 y Q4. Durante un

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

Q1: Durante este impulso se incrementa el Contador de Programa.
Q2-Q3: Durante la activación de estas dos señales se produce la decodificación y la ejecución de la instrucción.
Q4: Durante este impulso se busca el código de la instrucción en la memoria del programa y se carga en el Registro de Instrucciones.

En la Figura I.4 se observa una grafica en donde se muestran las cuatro señales en las que se divide una instrucción cuando es ejecutada.

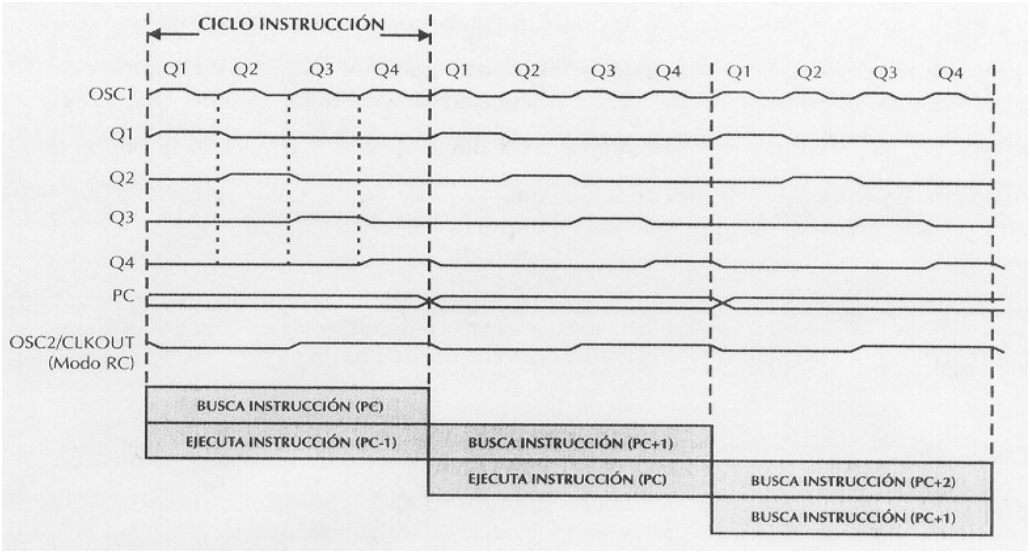


Figura I.4. Ejecución de las instrucciones en un ciclo de reloj.

Para conseguir ejecutar cada instrucción en un ciclo de instrucción (excepto las de salto), se aplica la técnica de segmentación o *pipe-line*, que consiste en realizar al mismo tiempo la ejecución de una instrucción y la búsqueda de código de la siguiente. Cuando la instrucción ejecutada corresponde a un salto no se conoce cuál será la siguiente hasta que se realice, por eso en esta situación se sustituye la fase de búsqueda por un ciclo “vacío”, originando que las instrucciones de salto tarde en realizarse dos ciclos de instrucción.

Terminal de RESET:

Cuando se aplica un nivel lógico bajo a la terminal MCLR# (terminal 4) el microcontrolador reinicializa su estado. Dos acciones importantes se producen en la reinicialización o RESET:

1. El Contador de Programa se carga con la dirección 0, apuntando a la primera dirección de la memoria de programa en donde deberá estar situada la primera instrucción del programa de aplicación.
2. La mayoría de los registros de estado y control del procesador toman un estado conocido y determinado.

Se puede provocar el RESET de varias maneras, pero si se desea realizar manualmente, habrá que colocar, conectado a la patilla MCLR#, un circuito con un pulsador, que al ser apretado genere un nivel lógico bajo. El circuito más simple para provocar un reset se muestra en la Figura I.5.

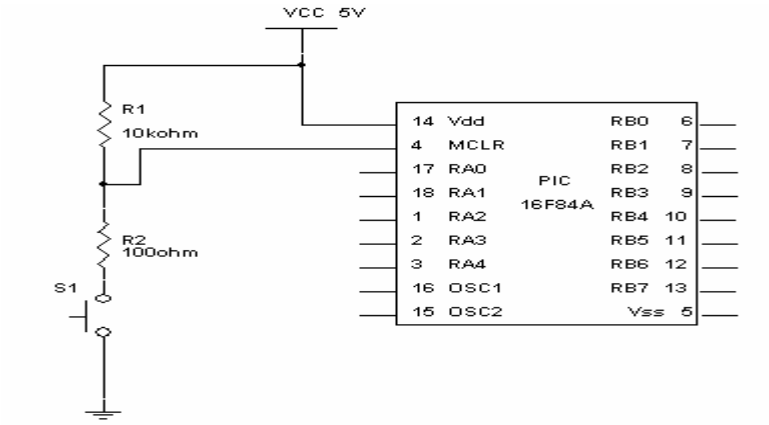


Figura I.5. Circuito de RESET para el PIC16F84.

En muchos proyectos que no se necesita aplicar un reset al PIC resulta útil conectar la terminal 4 directamente a 5V tal como se realiza en las prácticas planteadas.

Terminales de los puertos:

El PIC 16F84 cuenta con 13 terminales destinadas a la comunicación del PIC con el mundo exterior, estas terminales trabajan con niveles lógicos TTL de entre 0 y 5V. Estas 13 terminales se dividen en 2 puertos, el puerto A con 5 líneas RA0-RA4 (terminales 17, 18, 1, 2 y 3 respectivamente) y el puerto B con 8 líneas RB0-RB7 (terminales 6-13 respectivamente). Cada una de estas líneas se puede configurar como entrada (para recibir datos) o como salida (para enviar datos). Para realizar esta configuración se utilizan los registros especiales TRISA y TRISB tal como se explica mas adelante.

El puerto B tiene internamente unas resistencias de pull-up conectadas a sus pines (sirven para fijar el pin a un nivel de 5V), su uso puede ser habilitado o deshabilitado por medio del programa de control. Todas las resistencias pull-up se habilitan o deshabilitan utilizando el bit RBPU del

registro OPTION. Las resistencias pull-up se desconectan automáticamente en un pin si éste se programa como salida. RB0 puede programarse además como entrada de interrupciones externas INT. Los pines RB4 a RB7 pueden programarse para responder a interrupciones por cambio de estado. Las terminales RB6 y RB7 corresponden con las líneas de entrada de reloj y entrada de datos respectivamente, cuando se está programando el microcontrolador.

Como estos dispositivos son de tecnología CMOS, todos los pines deben estar conectados a alguna parte, nunca dejarlos al aire porque se puede dañar el circuito integrado. Los pines que no se estén usando se deben conectar a la fuente de alimentación de +5V, como se muestra en la Figura I.6:

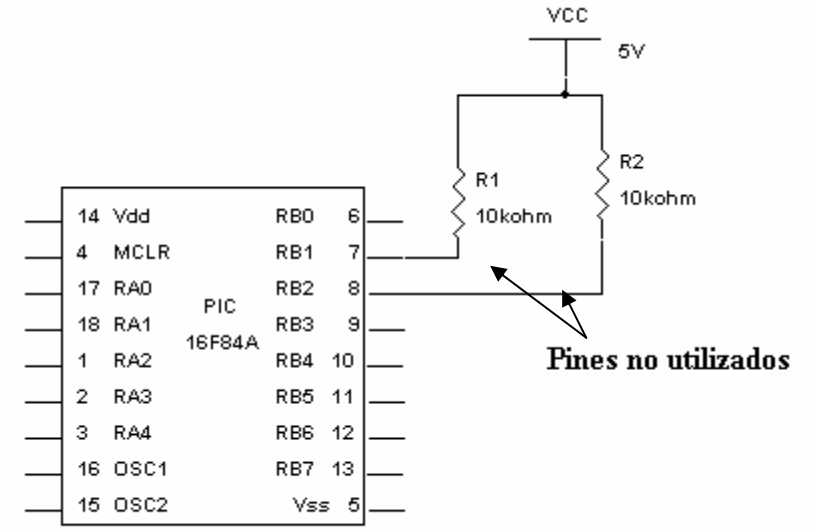


Figura I.6. Conexión de los pines no utilizados a Vcc.

El pin RA4/TOCKI como entrada puede programarse en funcionamiento normal o como entrada del contador/temporizador TMR0. Cuando este pin se programa como entrada digital, funciona como un disparador de Schmitt (Schmitt trigger), 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 de cinco voltios) tal como se observa en la Figura I.7.

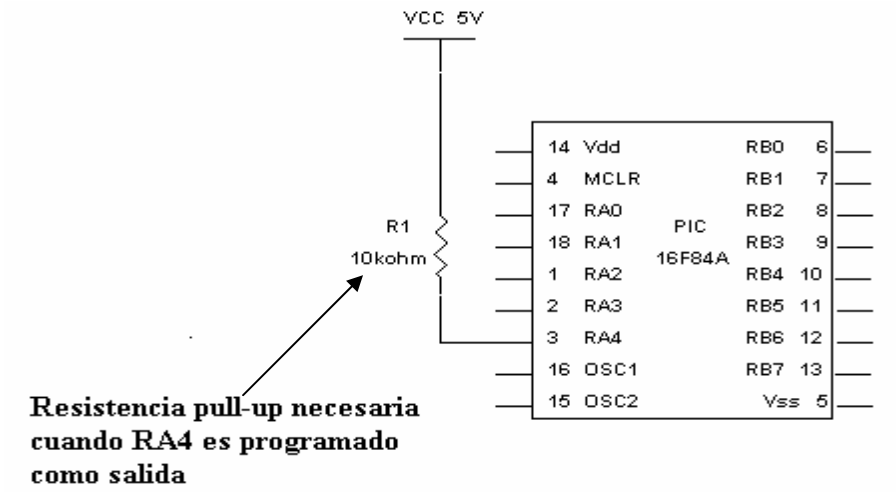


Figura I.7. Resistencia pull-up para la terminal RA4.

La máxima capacidad de corriente en cada una de las terminales de los puertos es:

- 25mA cuando la terminal esta en nivel bajo, es decir, consume corriente. Sin embargo la suma de corrientes por las 5 terminales del puerto A no debe de exceder de 80mA, ni la suma de corriente de las 8 terminales del puerto B no debe de exceder de 150mA.
- 20mA cuando la terminal esta en un nivel alto, es decir, entrega corriente. Sin embargo, la suma de las intensidades por las 5 terminales del puerto A no puede exceder los 50mA, ni la suma de las intensidades por las 8 líneas del puerto B puede exceder de 100mA.

I.3 Arquitectura del microcontrolador PIC16F84.

La arquitectura interna del PIC16F84 se muestra en la Figura I.8 donde se observan todos los bloques que lo forman.

Algunos de estos bloques son:

- Memoria de programa EEPROM o Flash de 1 k x 14 bits.
- Memoria de datos formada por dos áreas. Una RAM donde se alojan 22 registros de propósito específico (SFR) y 68 de propósito general (Conjunto de registros RAM), y otra de tipo EEPROM de 64 bytes.
- Una ALU de 8 bits y un registro de trabajo W del que normalmente recibe un operando y al cual envía el resultado. El otro operando puede provenir del bus de datos o del propio código de la instrucción.
- Diversos recursos conectados al bus de datos, tales como Puertas de Entrada/Salida, Temporizador TMR0, etc.
- Base de tiempos y circuitos auxiliares.

- Direccionamiento de la memoria de programa en base al Contador de Programa de 13 bits ligado a una Pila (Stack) de 8 niveles de profundidad.

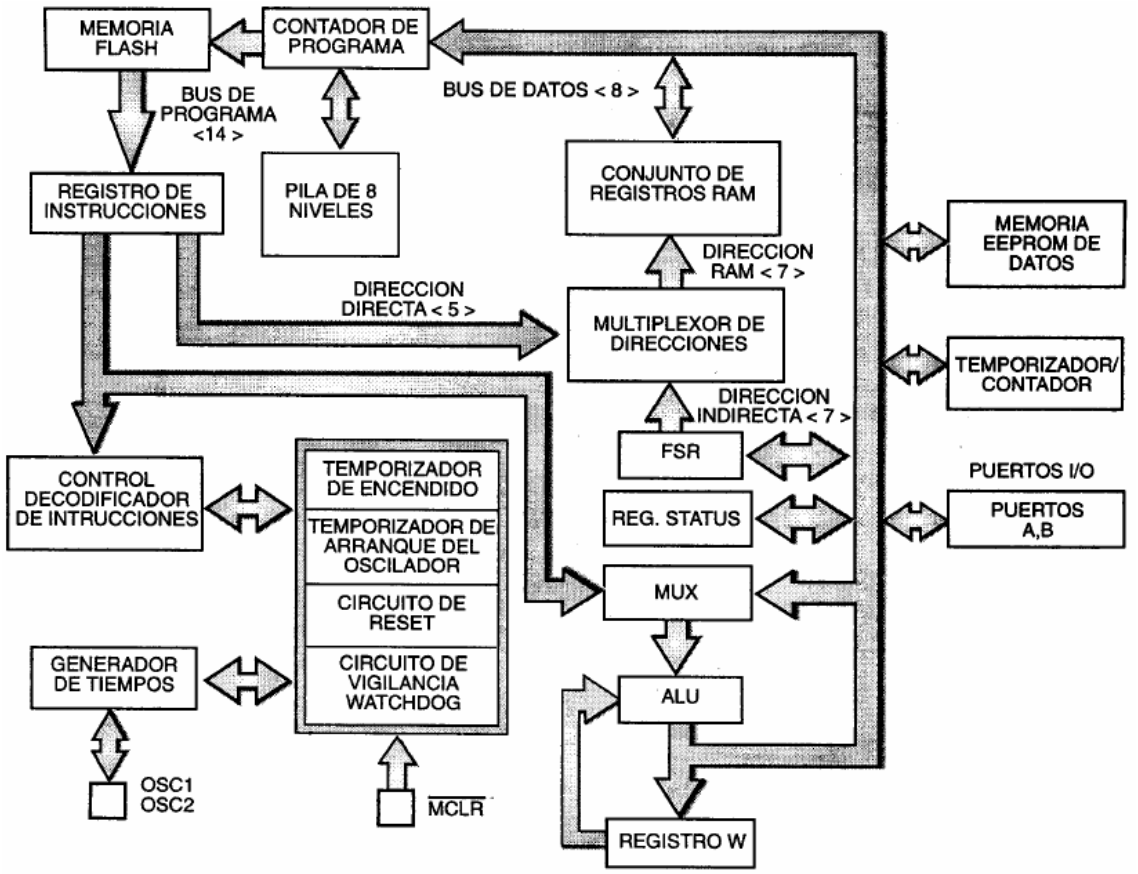


Figura I.8. Arquitectura interna del PIC16F84

I.4 Memoria de Programa.

La memoria de programa es aquella en la que se almacenan todas las instrucciones del programa de control que el micro debe ejecutar. Este programa se graba en la memoria mediante un grabador y se debe de conservar aun cuando el PIC no este polarizado de modo que no se necesite cargar el programa cada vez que se utiliza el PIC. Esto es muy importante ya que in microcontrolador se destina a una tarea y por tanto el programa a ejecutar siempre es el mismo.

El PIC 16F84A tiene una memoria de programa denominada ROM **FLASH** la cual se divide en páginas de 2,048 posiciones tal como se ve en la Figura I.9.

El PIC16F84A sólo tiene implementadas físicamente 1K posiciones (con palabras de 14 bits) es decir de 0000h a 03FFh y el resto no está implementado (*es aquello que se ve en gris*). El PIC16F84 se puede programar varias veces y esta memoria se puede grabar alrededor de 1000 veces. Esto resulta conveniente ya que con un solo chip podemos realizar todas las prácticas planteadas en este trabajo e incluso utilizarlo para algún otro proyecto.

Cuando ocurre un Reset, el contador de programa (PC) apunta a la dirección 0000h, y el micro se inicia nuevamente. Por esta razón, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo (por ejemplo, la configuración de los puertos, etc.).

Ahora, si ocurre una interrupción el contador de programa (PC) apunta a la dirección 0004h, entonces ahí es en donde se escribe la parte del programa necesaria para atender dicha interrupción.

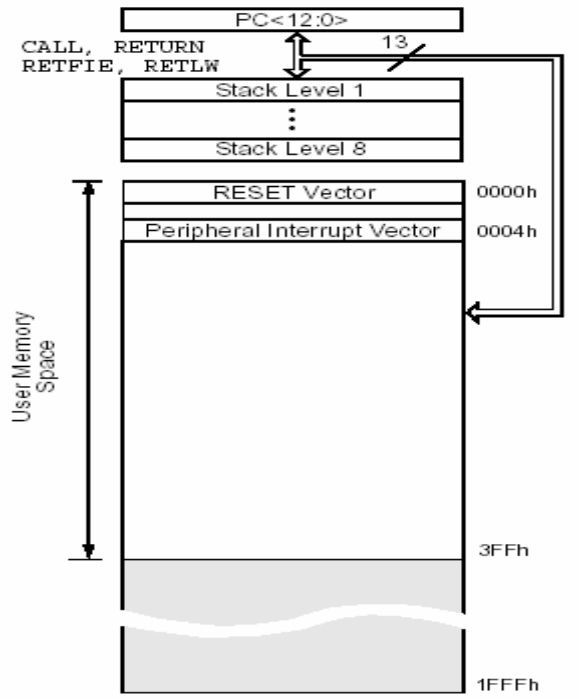


Figura I.9. Memoria de programa del PIC16F84.

I.5 El contador de programa (Program counter) PC.

El contador de programa (PC) es un registro interno del PIC que se utiliza en el direccionamiento de las instrucciones del programa. Este registro contiene la dirección de la siguiente instrucción a ejecutar y se incrementa

automáticamente de forma secuencial es decir una instrucción después de la otra a menos de que se trate de una instrucción de salto.
Al igual que todos los registros específicos que controlan la actividad del procesador, el Contador de Programa está implementado sobre un par de posiciones de la memoria RAM correspondientes a la posición 2 de los dos bancos de la memoria de datos. Este registro se explicara más adelante.

La Pila (stack) es una zona aislada de las memorias de instrucciones y datos. Tiene una estructura LIFO, en la que el último valor guardado es el primero que sale. Tiene 8 niveles de profundidad cada uno con 13 bits. La pila es utilizada solamente, y en forma automática, para guardar las direcciones de retorno de subrutinas e interrupciones, es decir cuando durante la ejecución del programa se encuentra una instrucción de salto (call por ejemplo) se almacena el valor actual del contador (PC) de programa en la posición superior de la pila. Al ocurrir dicho salto el PC toma el nuevo valor de memoria donde se localiza la instrucción a la cual se salto. Para recuperar el contenido de la pila hay que ejecutar la instrucción de retorno (return en este caso) con lo que el contador de programa recupera el valor que tenía antes de saltar y lo incrementa ejecutándose la instrucción siguiente de la instrucción de salto.

Algo que se debe tener en cuenta es la pila o Stack, que consta de 8 posiciones (*o niveles*), esto es como una pila de 8 platos el último en poner es el primero en sacar, si seguimos con este ejemplo, cada plato contiene la dirección y los datos de la instrucción que se está ejecutando, así cuando se efectúa una llamada (CALL) o una interrupción, el PC sabe donde debe regresar (*mediante la instrucción RETURN, RETLW o RETFIE, según el caso*) para continuar con la ejecución del programa. Debido a esto es muy importante que solo se pueden hacer 8 llamadas CALL tomando en cuenta las interrupciones ya que si se hiciera la llamada numero 9 la pila se sobrescribiría sobre el dato almacenado en la primera posición perdiéndose la dirección que se encontraba allí almacenada.

I.6 Registro de Configuración.

Se trata de una posición reservada de la memoria de programa situada en la dirección 2007h y accesible únicamente durante el proceso de grabación. Al escribirse el programa de la aplicación es necesario grabar el contenido de esta posición de acuerdo con las características del sistema. Se trata de una palabra de 14 bits los cuales se muestran en la Tabla I.1:

Bit 13- Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CP	PWRTE	WDTE	FOSC1	FOSC0

Tabla I.1. Bits del registro de configuración.

A continuación se describen cada uno de los bits de los que consta la palabra de configuración:

- **CP:** Bits de protección de código
  CP=1. La memoria de programa se puede leer. No esta protegida
  CP=0: La memoria de programa esta protegida. El programa no se puede leer, evitando copias. Tampoco se puede sobrescribir. Además evita que pueda ser accedida la memoria EEPROM de datos y, finalmente, si se modifica el bit CP de 0 a 1, se borra completamente la EEPROM.
- **PWRTE:** Activación del temporizador Power-up. El temporizador "power-up" retrasa 72 ms la puesta en marcha o Reset que se produce al conectar la alimentación al PIC, para garantizar la estabilidad de la tensión aplicada.
  PWRT=1. Desactivado
  PWRT=0. Activado

- **WDT:** Bit de activación del watchdog.
  WDT=1. Activado el WDT
  WDT=0. Desactivado

- **FOSC1-FOSC0:** Selección del tipo de oscilador
  FOSC<1:0>=11. Oscilador RC
  FOSC<1:0>=10. Oscilador HS (4MHz-20MHz)
  FOSC<1:0>=01. Oscilador XT (4MHz-20MHz)
  FOSC<1:0>=00. Oscilador LP (32KHz-200KHz)

I.7 Memoria de Datos.

En esta memoria se almacenan todos los datos que se manejan en el programa a ejecutar.
La memoria de datos del PIC16F84 dispone de dos zonas diferentes:

1. **Área de RAM estática o SRAM**, donde reside el Banco de Registros Específicos (SFR) y el Banco de Registros de Propósito General (GPR). El primer banco tienen 24 posiciones de tamaño de un byte, aunque dos de ellas no son operativas, y el segundo de 68 posiciones.

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

La zona de memoria **RAM de datos** se halla dividida en dos bancos (banco 0 y banco 1) de 128 bytes cada uno. En el PIC16F84 sólo se hallan implementadas físicamente las 48 primeras posiciones de cada banco.

Las 12 primeras están reservadas a los *Registros de Propósito Específico (SFR)*, que son los encargados del control del procesador y sus recursos. Algunos de dichos registros se hallan repetidos en la misma dirección de los dos bancos, para simplificar su acceso (INDF, STATUS, FSR, PCLATH e INTCON).

Las posiciones apuntadas por la dirección 07h y la apuntada por la 87h no son operativas y se leen como 0.

Los registros restantes de cada banco se destinan a *Registros de Propósito General*, que son usados para guardar datos temporales del programa que se esta ejecutando y en realidad sólo son operativos los 68 del banco 0 porque los del banco 1 se mapean sobre el banco 0, es decir, cuando se apunta a un registro general del banco 1, se accede al mismo del banco 0. Las posiciones de memoria 4Fh a la 7Fh del banco 0 y de la CFh a la FFh no se encuentran implementadas físicamente por lo que se leen como 0.

Para seleccionar el banco a acceder hay que manipular el bit 5 (RP0) del registro STATUS. Si RP0=1 se accede ala banco 1 y si RP0=0 se accede al banco 0. Tras un Reset se accede automáticamente al banco 0. Para seleccionar un registro de propósito general no hay que tener en cuenta el estado del bit RP0, porque al estar mapeado el banco 1 sobre el banco 0, cualquier direccionamiento de un registro del banco 1 corresponde a su homólogo del banco 0.

En la figura I.10 se observa como esta organizada esta memoria.

Cada uno de los registros especiales (FSR) cuenta con 8 bits los cuales se muestran en la Tabla I.2:

A continuación se explica brevemente para que sirve cada registro así como los bits que lo forman.

INDF: este registro existe en el banco 1 y 2 de memoria en las direcciones 00h y 80h respectivamente. No es un registro implementado físicamente. Se utiliza para el direccionamiento indirecto de datos. Este registro utiliza el contenido del registro **FSR** para seleccionar indirectamente la memoria

de datos RAM; la instrucción indicara lo que se bebe de hacer con el registro indicado.

Por ejemplo si el registro FSR=16h y en el programa a ejecutar aparece la instrucción *movwf INDF* se esta indicando que el contenido del registro *w* se carga en la posición de memoria 16h. A este procedimiento se le llama direccionamiento indirecto.

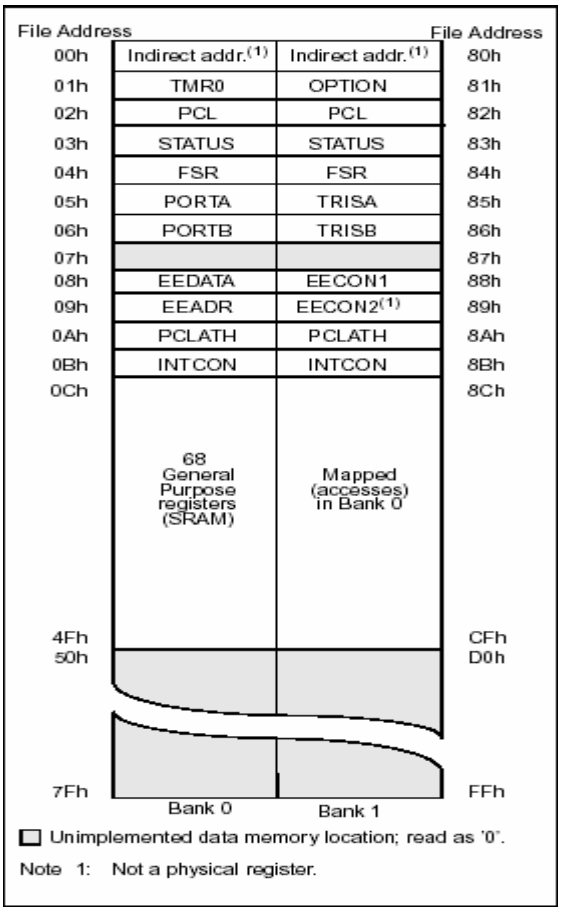


Figura I.10. Organización de la memoria de datos del PIC16F84.

Dirección	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Banco 0									
00h	INDF	Utiliza los registros de FSR para el direccionamiento indirecto de la memoria de datos (no es un registro físico)							
01h	TMR0	Timer / contador de 8 bits							
02 h	PCL	Registro con los 8 bits mas bajos del contador de programa (PC)							
03h	STATUS	IRP	RP1	RP0	/TO	/PD	Z	DC	C
04h	FSR	Registro puntero para el direccionamiento indirecto de la memoria de datos							
05h	PORTA	--------	-------	------	RA4/TOCKI	RA3	RA2	RA1	RA0
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT
07h		Posición no implementada se lee como 0							
08h	EEDATA	Registro de datos EEPROM							
09h	EEADR	Registro de direcciones EEPROM							
0Ah	PCLATH	--------	-------	------	Buffer escrito con los 5 bits mas altos del PC				
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
Banco 1									
80h	INDF	Utiliza los registros de FSR para el direccionamiento indirecto de la memoria de datos (no es un registro físico)							
81h	OPTION	/RBPU	INTDEG	TOSC	TOSE	PSA	PS2	PS1	PS0
82h	PCL	Registro con los 8 bits mas bajos del contador de programa (PC)							
83h	STATUS	IRP	RP1	RP0	/TO	/PD	Z	DC	C
84h	FSR	Registro puntero para el direccionamiento indirecto de la memoria de datos							
85h	TRISA	-------	----------	-------	Registro de configuración de los bits del Puerto A				
86h	TRISB	Registro de configuración de los bits del Puerto b							
87h		Posición no implementada se lee como 0							
88h	EECON1	--------	----------	-------	EEIF	WRERR	WREN	WR	RD
89h	EECON2	Registro de control para la grabación de la memoria EEPROM							
8Ah	PCLATH	--------	----------	-------	Buffer escrito con los 5 bits mas altos del PC				
8Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

Tabla I.2. Bits de los registros especiales de la memoria de programa.

TMR0 (Temporizador/Contador): Temporizador/contador de 8 bits, llamado TMR0 (timer 0), que actúa de dos maneras diferentes:

1. Como contador de impulsos que se aplican desde el exterior a la terminal RA4/T0CKI. Al llegar al valor FF H se desborda el contador y, con el siguiente impulso, pasa a 00 H, advirtiendo esta circunstancia con la activación de un señalizador y/o provocando una interrupción.
2. Como temporizador, cuando se carga en el registro que implementa el recurso un valor inicial que se incrementa con cada ciclo maquina (Fosc/4) hasta que se desborda, o sea, pasa de FF H a 00 H y avisa poniendo el bit señalizador y/o provocando una interrupción. Se utiliza para determinar intervalos de tiempo concretos.

Para que el TMR0 funcione como un contador o temporizador es necesario configurar algunos bits del registro OPTION el cual se describe mas adelante.

En realidad, el PIC16F84 dispone de dos temporizadores, el TMR0 y el Perro Guardián (*Watchdog*). El primero actúa como principal y sobre él recae el control de tiempos y la cuenta de impulsos. El otro vigila que el programa no se cuelgue, y para ello cada cierto tiempo comprueba si el programa se está ejecutando normalmente. En caso contrario, si el control está detenido en un bucle infinito a la espera de algún acontecimiento que no se produce, el Perro Guardián "ladra", lo que se traduce en un Reset que inicializa todo el sistema.

A menudo el TMR0 y el Perro Guardián precisan controlar largos intervalos de tiempo y necesitan aumentar la duración de los impulsos de reloj que les incrementa. Para cubrir este requisito se dispone de un circuito programable denominado Divisor de frecuencia, que divide la frecuencia utilizada por diversos rangos. Para programar el comportamiento del TMR0, el Perro Guardián y el Divisor de frecuencia se utilizan algunos bits del registro OPTION y de la Palabra de Configuración, que se verán más adelante.

El Divisor de frecuencias puede usarse con el TMR0 o con el WDT. Con el TMR0 actúa como *Pre-divisor*, es decir, los impulsos pasan primero por el Divisor y luego se aplican al TMR0, una vez aumentada su duración. Con el WDT actúa después, realizando la función de *Post-divisor*. Los impulsos, que divide por un rango el Divisor de frecuencia, pueden provenir de la señal de reloj interna (Fosc/4) o de los que se aplican a la patilla T0CKI.
El TMR0 puede ser leído y escrito en cualquier momento al estar conectado al bus de datos. Funciona como un contador ascendente de 8 bits. Cuando funciona como temporizador conviene cargarle con el valor de los impulsos que se quiere temporizar restados de 256 que es el valor de desbordamiento. De esta manera, al llegar al número de impulsos deseados se desborda y al pasar por 00 H se activa el señalizador TOIF del registro INTCON y se produce una interrupción.

Para calcular los tiempos a controlar con TMR0 se utilizan la siguiente fórmula.

$$Temporizacion = 4 * (1/f) * Divisor * (256 - C\arg a\_TMR0)$$

Donde:
Temporización: es el tiempo deseado.
F: es la frecuencia del oscilador (XT) utilizado.

Divisor: es el rango de división de frecuencia elegido.
(256-Carga_TMR0): es el número total de impulsos a contar antes de que se desborde.

PCL (Progam counter low byte): Byte bajo del contador de programa ocupa la posición de memoria 02h en el banco 1 y 82h del banco 2. El PIC 16F84 dispone de un contador de programa de 13 bits. Sus bits de menor peso corresponden a los 8 bits del registro PCL. Los 5 bits de más peso del PC (contador de programa), residen en los 5 bits de menos peso del registro PCLATH, que ocupa la posición 0A H de los dos bancos de la memoria RAM. En las instrucciones de salto, los 11 bits de menos peso del PC provienen del código de instrucción y los otros dos de los bits de PCLATH <4:3>, tal y como se muestra en la Figura I.11.

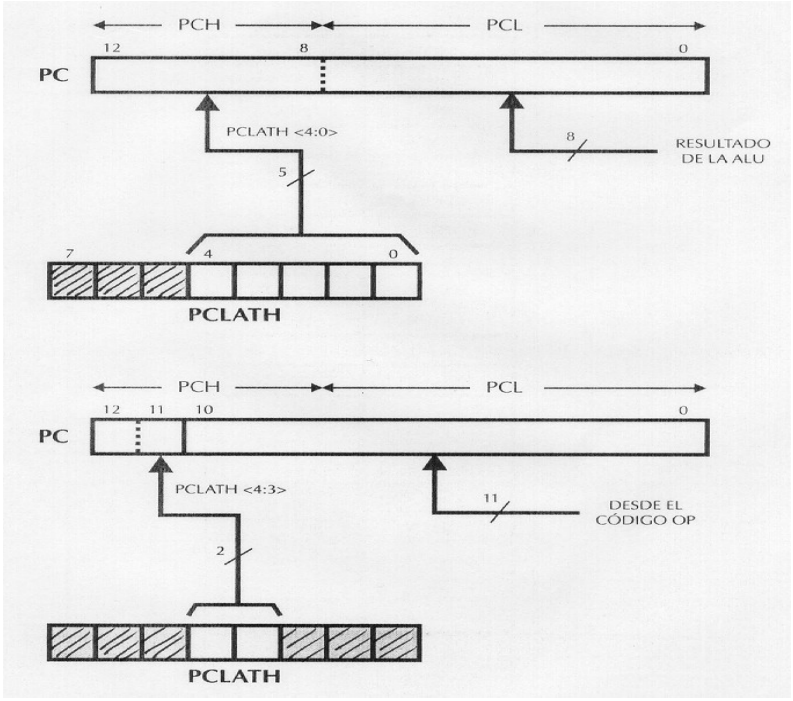


Figura I.11. Bits del contador de programa.

STATUS: Ocupa la posición de memoria 03h del banco 1 y la 83h del banco 2. Sus bits tienen tres misiones distintas:

Se encargan de avisar de la incidencias del resultado de la ALU (C, DC y Z).
Indican el estado de Reset (TO# y PD#).
Seleccionan el banco a acceder en la memoria de datos (IRP, RP0 y RP1).

En la Tabla I.3 se muestran los bits del registro STATUS.

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

Tabla I.3. Bits del registro STATUS.

La misión de cada bit es la siguiente:

- **C (carry bit):** Acarreo en el bit de más peso. En las instrucciones de suma se activa cuando el resultado ha rebasado la capacidad del registro sobre el que trabaja es decir el resultado ha excedido el valor 255 (11111111) que es el maximo valor que se puede representar con los 8 bits.
  C=1: En la suma indica acarreo y en la resta que el resultado ha sido positivo.
  C=0: En la suma indica que no ha habido acarreo y en la resta que el resultado ha sido negativo.

.

- **DC (digit carry):** Acarreo en el 4º bit. En las operaciones aritméticas indica que ha habido acarreo entre los bits 3 y 4.
  DC=1: Acarreo en el 4º bit.
  DC=0: No acarreo en el 4º bit.

- **Z (zero):** Flag de cero. Este flag se activa a "1" cuando el resultado de una operación aritmetica o logica es cero.
  Z=1: El resultado de una instrucción lógico-aritmética ha sido cero.
  Z=0: El resultado de una instrucción lógico-aritmética no ha sido cero.

- **/PD (Power Down):** Flag de bajo consumo. Es un bit de solo lectura, no puede ser escrito por el usuario. Sirve para detectar el modo de bajo consumo.
  /PD=1: Se pone a este valor después de la conexión a la alimentación o al ejecutar *clrwdt*.
  /PD=0: Se pone a este valor al ejecutar *sleep*.

- **/TO (Time Out):** Flag indicador del fin de la temporización del Watchdog. Es un bit de solo lectura. Se pone a "0" cuando el watchdog finaliza la temporización.
  /TO=1: Se pone a este valor después de la conexión a la alimentación o al ejecutar *clrwdt* y *sleep*.
  /TO=0: Se pone a este valor al desbordarse el Perro Guardián (*Watchdog*).

- **RP1-RP0:** Selección de banco en direccionamiento directo. Como el PIC16F84 sólo tiene dos bancos únicamente emplea el bit RP0. Después de un Reset RP0=0.
  RP0=1: Se accede al banco 1.
  RP1=0 Se accede al banco 0.

- **IRP: Selección del banco en direccionamiento indirecto**
  Este bit junto con el de más peso del registro FSR sirven para determinar el banco de la memoria de datos seleccionado. En el PIC16F84 al disponer de dos bancos no se usa este bit.

FSR: Selector de registros para direccionamiento indirecto. Junto con el registro INDF se utiliza para seleccionar indirectamente los otros registros disponibles. Ocupa las posiciones de memoria 04h del banco 0 y 84h del banco 1.

PORTA (puerto A): Registro que ocupa la posición de memoria 05h del banco 0, el cual muestra el estado de los 5 bits de este puerto (Tabla I.4) cada uno de los cuales puede ser configurado como entrada/salida utilizando el registro TRISA.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	---	---	RA4	RA3	RA2	RA1	RA0

Tabla I.4. Bits del registro PORT A.

PORTB (puerto B): Registro que ocupa la posición de memoria 06h del banco 0, el cual muestra el estado de los 8 bits de este puerto (Tabla I.5) cada uno de los cuales puede ser configurado como entrada/salida utilizando el registro TRISB.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

Tabla I.4. Bits del registro PORT B.

EEDATA (EEPROM data register): Registro que ocupa la posición de memoria 08h el cual contiene los bits que se van a escribir o que se han leido de la EEPROM de datos.

EEADR (EEPROM addres register): Registro que ocupa la posición de memoria 09h del banco 0. Contiene la dirección de la EEPROM de datos a la que se va a acceder para leer o escribir.

PCLATH (PC latch high): Registro que ocupa la posición 0Ah del banco 0, el cual permite acceder de forma indirecta a la parte alta del PC

(contador de programa) en algunas instrucciones tal como se describió en el registro PCL.

INTCON (Interrupts control register): Registro de control de interrupciones el cual ocupa las posiciones de memoria 0Bh del banco 0 y 8Bh del banco 1. Los bits de este registro se muestran en la Tabla I.6.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Tabla I.6.Bits del registro INTCON.

La función de cada bit se describe a continuación.

- **GIE: Permiso Global de Interrupciones**
  GIE=0. Prohíbe todas las interrupciones.
  GIE=1. Permite la ejecución de todas las interrupciones, cuyos bits de permiso individuales también las permitan.

Cuando reconoce una interrupción se borra automáticamente para impedir que otra sea atendida y al retornar de la interrupción (mediante la instrucción retfie) se pone a "1" habilitando las interrupciones.

- **EEIE: Permiso de Interrupción por fin de escritura en la EEPROM**
  EEIE=0. Prohíbe que se produzca esta interrupción.
  EEIE=1.Permite que se origine esta interrupción cuando termina la escritura en la EEPROM de datos.

- **T0IE: Permiso de Interrupción por desbordamiento del TMR0**
  TOIE=0.Prohíbe esta interrupción.
  TOIE=1. Permite una interrupción al desbordarse el TMR0.

- **INTE: Permiso de Interrupción externa por activación de la patilla RB0/INT**
  INTE=0. Prohíbe esta interrupción.
  INTE=1. Permite la interrupción al activarse RB0/INT.
- **RBIE: Permiso de Interrupción por cambio de estado en RB7-RB4**
  RBIE=0. Prohíbe esta interrupción
  RBIE=1. Permite esta interrupción.

- **T0IF: Señalizador de desbordamiento del TMR0**
  TOIF=0. Indica que el TMR0 no se ha desbordado.
  TOIF=1. Toma este valor cuando ha ocurrido el desbordamiento.

- **INTF: Señalizador de activación de la patilla RB0/INT**
  INTF=0. No ha habido interrupción externa por el pin RB0/INT.
  INTF=1. Ha ocurrido una interrupción externa por el pin RB0/INT.

- **RBIF: Señalizador de cambio de estado en las patillas RB7-RB4**
  RBIF=0. No ha cambiado el estado de RB7-RB4.
  RBIF=1. Cualquiera de las líneas RB7-RB4 ha cambiado de estado.

OPTION. Este registro ocupa la posición 81h del banco 1 de memoria de datos. La misión principal de este registro es controlar TMR0 y el Divisor de frecuencia. Los bits que forman a este registro se muestran en la Tabla I.7.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
/RBPU	INTDEG	T0CS	TOSE	PSA	PS2	PS1	PS0

Tabla I.7.Bits del registro OPTION.

La función de cada bit se describe a continuación:

- **/RBPU:** habilita las resistencias Pull-Up del puerto B.
  /RBUP=0. Habilita las resistencias pull-up del puerto B.
  /RBUP=1. Deshabilita las resistencias pull-up del puerto B.

- **INTDEG:** Selector de flanco activo que provocará una interrupción externa al aplicarse a la terminal RB0/INT.
  INTDEG=0. Interrupción por flanco de bajada de la terminal RB0/INT.
  INTDEG=1. Interrupción por flanco de subida de la terminal RB0/INT.

- **TOCS:** Bit que sirve para seleccionar la procedencia de los impulsos de reloj, que pueden ser del oscilador interno (Fosc/4) o los que se aplican desde el exterior por la patilla T0CKI.
  TOCS=0. Pulsos de reloj interno Fosc/4 (TMR0 como temporizador).

TOCS=1. Pulsos introducidos a través de la terminal RA4/TOCKI (TMR0 como contador)

- **TOSE:** Este bit sirve para elegir el tipo de flanco activo en los impulsos externos.
  TOSE=0. TMR0 se incrementa con cada flanco ascendente de la señal aplicada por la terminal RA4/TOCKI.
  TOSE=1. TMR0 se incrementa con cada flanco descendente de la señal aplicada por la terminal RA4/TOCKI.

- **PSA:** Asignación al divisor de frecuencia ya sea al WDT o al TMR0.
  PSA=0. El divisor de frecuencias se asigna al TMR0.
  PSA=1. El divisor de frecuencias se asigna al watchdog.

- **PS2-PS0:** Estos bits permiten seleccionar valores del Preescaler con el que actúa el divisor de frecuencia según la Tabla I.8.

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

Tabla I.8. Valores del divisor de frecuencia.

TRISA: Registro que ocupa la posición 85h del banco 1, es utilizado para la configuración de las 5 líneas del puerto A como entradas o salidas. Cada pin se puede configurar independientemente de los demás. Un “0” en el bit correspondiente configura esa línea como salida, mientras que un “1” la configura como entrada.

TRISB: Registro que ocupa la posición 86h del banco 1, es utilizado para la configuración de las 8 líneas del puerto B como entradas o salidas. Cada pin se puede configurar independientemente de los demás. Un “0” en el bit correspondiente configura esa línea como salida, mientras que un “1” la configura como entrada.

EECON1: Registro que ocupa la dirección 88h del banco 1 de la memoria de datos, se utiliza para el control de las operaciones en la EEPROM. Los bits que lo forman se muestran en la Tabla I.9.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	---	---	EEIF	WRERR	WREN	WR	RD

Tabla I.9. Bits del registro EECON1.

La función de cada bit se describe a continuación:

- **RD:** Bit de control de lectura de la EEPROM. Cuando se pone a "1" inicia la lectura de un byte en la EEPROM de datos. Este bit se pone a "0" automáticamente al finalizar la lectura de la posición EEPROM.
  RD=0. No inicia la lectura de la EEPROM o la misma ha terminado.
  RD=1. Inicia la lectura de la EEPROM de datos. Se borra al finalizar la lectura.

- **WR:** Bit de control de la escritura en la EEPRO: AL ponerlo a "1" se inicia la escritura en la EEPROM de datos. Este bit se pone automáticamente a "0" una vez finalizada la escritura de la EEPROM.
  WR=0. No inicia la escritura de la EEPROM de datos o la misma ha terminado.
  WR=1. Inicia la escritura de la EEPROM de datos, se borra automáticamente al finalizar la escritura de la misma.

- **WREN:** Permiso de escritura en la EEPROM.
  WREN=1. Permite la escritura en la EEPROM.
  WREN=0. Prohíbe la escritura en la EEPROM.

- **WRERR:** Señalizador de error en escritura.
  WRERR=1: Se pone a 1 cuando una operación de escritura ha terminado prematuramente.
  WRERR=0. La operación de escritura se ha completado correctamente.

- **EEIF**: Señalizador de final de operación de escritura
  EEIF=1. Cuando este señalizador se pone a 1 indica que la operación se ha completado con éxito. Debe ponerse a "0" por programa.

EEIF=0. La operación de escritura no se ha completado o no ha comenzado.

EECON2: Este registro no esta implementado físicamente por lo que es imposible leerlo. Se emplea como dispositivo de seguridad durante la escritura de la EEPROM, para evitar las interferencias a lo largo del intervalo de tiempo que precisa su desarrollo.

I.8 La memoria de datos EEPROM

La **memoria de datos EEPROM** consta de 64 bytes para almacenar datos que no se pierden al desconectar la alimentación permitiendo almacenar datos en forma permanente. Esta memoria como cualquier otra memoria EEPRO permite realizar dos tipos de operaciones:

- Lectura
- Escritura.

Al escribir en una posición de memoria ya ocupada, se borra automáticamente el dato que ya existía y se almacena el nuevo dato.

Para poder efectuar las operaciones de escritura y lectura de esta memoria es necesario hacer uso de algunos registros los cuales solo se mencionan brevemente puesto que ya fueron explicados en su momento:

- **EEDATA:** Contiene los bytes que se van a escribir o que se han leído de la EEPROM de datos.
- **EEADR:** Contiene la dirección de la EEPROM de datos en la cual se va a leer o escribir un dato
- **EECON1:** Los bits de este registro indican si se va a leer (RD) o a escribir (WR) sobre esta memoria. Estos dos bits se deben de poner a "1" ya que se borran automáticamente cuando la operación de lectura o escritura han terminado.
- **EECON2:** Este registro no esta implementado físicamente y es utilizado como dispositivo de seguridad durante el proceso de escritura de la EEPROM, para evitar interferencias ya que un ciclo de grabación de esta memoria dura alrededor de 10ms lo cual es un tiempo muy largo para la velocidad del procesador.

Lectura de la EEPROM de datos.

Para poder leer una posición de esta memoria es necesario seguir una secuencia de pasos:

1. El registro EEADR debe de contener la posición de memoria a leer.
2. El bit RD del registro EECON1 debe ponerse a “1”.
3. Una vez hecho esto el microcontrolador espera a que finalicé la operación de lectura y pone a “0” el bit RD.
4. Se lee el dato en el registro EEDATA.

El siguiente es un pedazo de programa para la lectura de la memoria EEPROM de datos.

```
        bcf     STATUS, RP0    ; Se trabaja con el Banco 0.
        movlw   Dirección   ; Mueve la dirección a leer en la memoria al registro w.
        movwf   EEADR       ; Se almacena en este registro la Dirección a leer
        bsf     STATUS, RP0    ; Acceso  al Banco 1
        bsf     EECON1, RD     ; Lectura de la EEPROM
Sigue_leyendo           ; etiqueta
        btfsc   EECON1,RD   ;Si ha terminado la lectura RD=0
        goto    Sigue_leyendo   ;Si RD=0 se ignora esta instruccion
        bcf     STATUS, RP0    ; Acceso al Banco 0
        movf    EEDATA, W   ; Se mueve el contenido del registro EEDATA
                            ; al registro w.
```

Escritura en la EEPROM de datos.

Al igual que para la lectura es necesario seguir una secuencia de pasos para poder escribir en la memoria EEPROM de datos.

1. Se almacena en el registro EEDATA el dato a grabar.
2. Se carga en el registro EEADR la posición de la memoria a escribir
3. Se deshabilitan las interrupciones del microcontrolador.
4. Se ejecuta la secuencia que el fabricante marca como necesaria para la escritura de cada byte.
5. El bit WR del registro EECON1 debe ponerse a “1”. Para comenzar la escritura.
6. Al terminar la escritura los bits del registro EECON1 WR y EEIF se ponen automáticamente a “0” y a “1” respectivamente.
7. Se debe poneer a “0” el bit EEIF del registro EECON1

El siguiente es un pedazo de programa para poder efectuar la escritura en la memoria EEPROM:

```
CBLOCK
Int_iniciales
ENDC
```

```
        bsf     STATUS,RP0      ;Acceso al banco 1
        movlw   Dato    ; Mueve el Dato a escribir en la memoria al registro w.
        movwf   EEDATA      ; Se almacena el byte del Dato en este registro
        movlw direccion     ;la posición de memoria  en la cual se va
        movwf  EEADR        ; a escribir se guarda en este registro
        movwf   INTCON,w    ;Se almacena el estado de las interrupciones
        movwf   Int_iniciales     ; en el registro Int_iniciales
        bsf     STATUS,RP0  ;Acceso al banco 1
        bcf     INTCON,GIE      ;Se deshabilitan las interrupciones
        bsf     EECON1,WREN     ; Habilita la escritura
        movlw   0x55            ; *Esta parte el fabricante la marca como
        movwf   EECON2          ; *necesaria para poder realizar
        movlw   0xAA            ; *la escritura de cada byte
        movwf   EECON2          ; *de la memoria EEPROM.
        bsf     EECON1,WR       ; Inicia la escritura en la EEPROM
        bsf     INTCON,GIE      ; Habilita las interrupciones
Fin_escribir  ; etiqueta
        btfsc   EECON1,WR   ; Si se ha terminado de escribir en la EEPROM
WR=0.
        goto    Fin_escribir    ; Si WR=0 se ignora esta instrucción
        bcf     EECON1,WREN         ; Desautoriza la escritura en EEPROM.
        bcf     EECON1,EEIF         ; Limpia este flag.
        bcf     STATUS,RP0          ; Acceso al Banco 0.
        movf    Int_iniciales,w ; Restaura el valor anterior de INTCON.
        movwf   INTCON
```

I.9 Instrucciones del microcontrolador.

El microcontrolador al ser un sistema digital programable necesita de un juego de instrucciones (comandos) para poder realizar programas que satisfagan las necesidades planteadas.

Cada uno de estos comandos o nemónicos representan las instrucciones en lenguaje ensamblador, cada una de las cuales son traducidas a código maquina por el programa ensamblador.

Cada nemónico esta escrito con las iniciales en ingles de la instrucción a ejecutar. Todas las instrucciones tienen una longitud de 14 bits y todos los datos manejados son de 8 bits.

Es importante que tener en claro las notaciones que se deberán tomar en cuenta para poder interpretar las instrucciones las cuales son:

Notación para los números.

- **Decimal** : D’100’ ó d’100’ ó .100
- **Hexadecimal** : H’64’ ó h’64’ ó 0x64 ó 64
- **Octal** : O’144’
- **Binario** : B’01101100’ ó b’01101100’
- **ASCII** : A’C’ ó ‘C’

Notación para registros y literales.

- **w:** Registro W, similar al acumulador, es el registro de trabajo.
- **f:** Campo de 5 bits (fffff), contiene la dirección del banco de registros, que ocupa el banco 0 del área de datos. **Direcciona uno de esos registros.**
- **k:** Representa una constante de 8 bits.
- **d:** Bit del código OP de la instrucción. Selecciona el destino donde se guarda el resultado de una operación. Si d=0, el destino es W, y si d=1 el destino es f.
- **b** : Determina la posición de un bit dentro de un registro de 8 bits, (*o sea, tomará valores entre 0 y 7*)

Símbolos.

- **[]:** Opciones.
- **():** Contenido.
- $\rightarrow$: Se asigna a ...
- **<>:** Campo de bits de un registro.
- **E** : Pertenece al conjunto ...
- **Label:** Nombre de la etiqueta.
- **TOS:** Cima de la pila con 8 niveles en la gama media.
- **PC:** Contador de programa que direcciona la memoria de instrucciones.

Flags.

Los Flags o banderas son marcadores, representados por bits dentro del registro STATUS, y son:

- **Z:** Flag de cero, se pone a 1 cuando una operación lógica o aritmética da 0 (cero) como resultado. En cualquier otro caso se pone a 0.

- **C:** Flag de Carry, se pone a 1 cuando la operación que le afecta sobrepasa el nivel de representación del procesador, en nuestro caso es de 8 BIT's , de esta manera si sumamos a b'11111111' un b'00000011' el resultado sería b'00000010' y el BIT de Carry pasaría a 1.
  C=1: En la suma indica acarreo y en la resta que el resultado ha sido positivo.
  C=0: En la suma indica que no ha habido acarreo y en la resta que el resultado ha sido negativo.

- **DC:** Flag de carry del nibble inferior, este se comporta igual que el BIT de Carry, solo que el límite de representación son los 4 bits inferiores, de esta manera si tenemos 0b00001111 y sumamos 0b00000111, el resultado será 0b00010110 y el BIT de DC se pone a 1, el BIT de Carry estará a 0 al no superarse los 8 bits y el de Z a 0 al ser el número diferente de 0.

*El set de instrucciones del microcontrolador PI16F84A***.**

El set de instrucciones del microcontrolador PIC 16F84A consta de 35 instrucciones las cuales se muestran en la tabla I.10 en la cual cada instrucción se encuentra clasificada según su función.

Como ya se menciono las instrucciones son representadas por un código de 14 bits y a continuación se explica como esta dividido dicho código y que representa.

Instrucciones para operar con registros de 1 byte.

Estas instrucciones permiten operar con registros de longitud de 1 byte almacenados en la memoria RAM de datos (de la dirección 00h a 7Fh).

El formato general para este tipo de instrucciones se muestra en la Tabla I.11:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
Código de operación						d	Dirección del registro "f" (00h a 7Fh)						

Tabla I.11. Instrucciones para operar con registros de 1 byte.

En donde se pueden observar los 14 bits de los que constan las instrucciones. Del bit 0 al bit 6 se indica la dirección de memoria de datos en la cual se ubica el registro "f" al cual hace referencia la instrucción. El bit 7 indica el lugar donde se almacena el resultado de la operación, si d=0

se guarda en “w” y si d=1 se guarda en el registro “f”. Del bit 8 al bit 13 se tiene el código de operación en lenguaje máquina correspondiente al nemónico del que se trate.

Nemónico/operando	Descripción	Flags afectados
Instrucciones para operar con registros de 1 byte		
addwf f,d	Sumar w y f y el resultado se almacena en d	C,DC,Z
andwf f,d	AND entre w y f, el resultado se almacena en d	Z
clrf f	Pone a 0 el registro f	Z
clrw	Pone a 0 w	Z
comf f,d	Complementar f, el resultado se almacena en d	Z
decf f,d	Decrementa f y el resultado se almacena en d	Z
decfsz f,b	Decrementar f, si es 0 salta la siguiente instrucción	
incf f,d	Incrementa f y el resultado se almacena en d	Z
incfsz f,b	incrementar f, si es 0 salta la siguiente instrucción	
iorwf f,d	OR entre w y f, el resultado se almacena en d	Z
movf f,d	Mover f a d	Z
movwf f	Mueve el dato de w a f	
nop	No operación	
rlf f,d	Rota a la izquierda a través del carry y el resultado se almacena en d	C
rrf f,d	Rota a la derecha a través del carry y el resultado se almacena en d	C
subwf f,d	Resta f-w y el resultado se almacena en d	C,DC,Z
swapf f,d	Intercambia los nibbles de f y se almacena en d	
xorlwf f,d	XOR entre w y f, el resultado se almacena en d	Z
Instrucciones que operan con un bit de un registro		
bcf f,b	Pone a 0 el bit ‘b’ del registro f	
bsf f,b	Pone a 1 el bit ‘b’ del registro f	
btfsc f,b	Checa el bit ‘b’ de ‘f’, salta si es 0	
btfss f,b	Checa el bit ‘b’ de ‘f’, salta si es 1	
Instrucciones de control y operación con una literal (constante)		
addlw k	Sumar k con w y el resultado se almacena en w	C,DC,Z
andlw k	AND entre k y w, el resultado se almacena en w	Z
call k	Llamada de la subrutina k	
clrwdt	Borrar el timer del watchdog	/TO,/PD
goto k	Salta a la dirección k	
iorlw k	OR entre w y k, el resultado se almacena en w	Z
movlw k	Mueve la literal k a w	
retfie	Retorno de una interrupción	
retlw k	Retornar y cargar w con el valor de k	
return	Retorno de subrutina	
sleep	Entra en modo de bajo consumo	/TO,/PD
sublw k	Resta k-w y el resultado se almacena en w	C,DC,Z
xorlw k	XOR entre w y k, el resultado se almacena en w	Z

Notas:
Si d=0 el resultado se almacena en w.
Si d=1 el resultado se almacena en el registro f

Tabla I.10. El set de instrucciones del microcontrolador PIC 16F84A

Instrucciones que operan con un bit de un registro.

Este tipo de instrucciones opera únicamente con un bit específico de un registro al cual se hace referencia.

El formato general para este tipo de instrucciones se muestra en la Tabla I.12

Del bit 0 al bit 6 se indica la dirección de memoria de datos en la cual se ubica el registro “f” al cual hace referencia la instrucción. Del bit 7 al bit 9 se indica la ubicación del bit con el que se va a operar, el cual es uno de los 8 bits que forman al registro “f”. Del bit 10 al bit 13 se tiene el código de operación en lenguaje máquina correspondiente al nemónico del que se trate.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
Código de operación				b			Dirección del registro “f” (00h a 7Fh)						

Tabla I.12. Instrucciones que operan con un bit de un registro.

Instrucciones de control y operación con una literal (constante).

Las instrucciones de operación con una literal permiten la operación entre un valor almacenado en un registro y un valor constante “k”.
Las instrucciones de control direccional el flujo de un programa.
El formato general para este tipo de instrucciones se muestra en la Tabla I.13

13	12	11	10	9	8	7	6	5	4	3	2	1	0
Código de operación						k (valor constante de 8 bits)							

Tabla I.13. Instrucciones de control y operación con una literal (constante).

Del bit 0 al bit 7 se especifica el valor binario de la constante “k” (0<k<255). Del bit 10 al bit 13 se tiene el código de operación en lenguaje máquina correspondiente al nemónico del que se trate.

Las instrucciones **CALL** y **GOTO** usan otro formato el cual se muestra en la Tabla I.14

13	12	11	10	9	8	7	6	5	4	3	2	1	0
Código de operación			k (valor constante de 11 bits)										

Tabla I.14. Formato de las instrucciones CALL y GOTO.

Del bit 0 al bit 10 se especifica el valor binario de la constante “k” (0<k<2047). Del bit 11 al bit 13 se tiene el código de operación en lenguaje máquina correspondiente al nemónico del que se trate.

A continuación se describen a detalle cada una de las instrucciones del PIC.

ADDLW Suma la literal k con w

Sintaxis: [label] addlw k

Operandos: $0 \leq k \leq 255$

Operación: (w)+(k) → (w)

Flags afectados: C, DC, Z

Código de operación:

11	111x	kkkk	kkkk

Descripción: suma el contenido del registro 'w' y el valor de 'k', almacena el resultado en 'w'.

Ejemplo: addlw b'11111111'

Antes: w=b'00000011' y C=¿?

Después: w=b'00000010' y C=1

ADDWF Suma w con el registro f

Sintaxis: [label] addwf f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación: (w)+(f) → (destino)

Flags afectados: C, DC, Z

Código de operación:

00	0111	dfff	ffff

Descripción: suma el contenido del registro 'w' y el contenido del registro 'f' y almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: addwf reg,1

Antes: w=0x17, reg=0xC2

Después: w=0x17, reg=0xD9

ANDLW w AND k

Sintaxis: [label] andlw k

Operandos: $0 \leq k \leq 255$

Operación: (w)AND(k) → (w)

Flags afectados: Z

Código de operación:

11	1001	kkkk	kkkk

Descripción: Hace la operación lógica AND entre el contenido del registro 'w' y el valor de 'k', el resultado se almacena en 'w'.

Ejemplo: andlw b'11111111'

Antes: w=b'10001011' y Z=¿?

Después: w=b'10001011' y Z=0

ANDWF w AND f.

Sintaxis: [label] andwf f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación(w)AND(f) → (destino)

Flags afectados: Z

Código de operación:

00	0101	dfff	ffff

Descripción: Hace la operación lógica AND entre el contenido del registro 'w' y el contenido del registro 'f' y almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: andwf reg,0

Antes: w=0x00, reg=0xA2, Z=¿?

Después: w=0x00, reg=0xA2, Z=1

BCF Pone a cero un bit.

Sintaxis: [label] bcf f,b

Operandos: $0 \leq f \leq 127$, $0 \leq b \leq 7$

Operación: 0 →(f<b>)

Flags afectados: Ninguno

Código de operación:

01	00bb	bfff	ffff

Descripción: Pone a cero el bit 'b' del registro 'f'.

Ejemplo: bcf reg,5

Antes: reg=b'11111111'

Después: reg=b'11011111'

BSF Pone a uno un bit.

Sintaxis: [label] bsf f,b

Operandos: $0 \leq f \leq 127$, $0 \leq b \leq 7$

Operación: 1 →(f<b>)

Flags afectados: Ninguno

Código de operación:

01	01bb	bfff	ffff

Descripción: Pone a uno el bit 'b' del registro 'f'.

Ejemplo: bsf reg,5

Antes: reg=b'00000000'

Después: reg=b'00100000'

BTFSC Test del bit 'b' del registro 'f' salta si es cero.

Sintaxis: [label] btfsc f,b

Operandos: $0 \leq f \leq 127$, $0 \leq b \leq 7$

Operación: Salta si (f<b>)=0

Flags afectados: Ninguno

Código de operación:

01	10bb	bfff	ffff

Descripción: Si el bit 'b' del registro 'f' es "0" se salta la instrucción siguiente ignorándola, si dicho bit es "1" se ejecuta la siguiente instrucción. Cuando salta necesita de 2 ciclos maquina en ejecutarse.

Ejemplo:
```
checa   btfsc  reg,0
Uno     goto   fue_uno
Cero    instrucción x
```

BTFSS Test del bit 'b' del registro 'f' salta si es uno.

Sintaxis: [label] btfss f,b

Operandos: $0 \leq f \leq 127$, $0 \leq b \leq 7$

Operación: Salta si (f<b>)=1

Flags afectados: Ninguno

Código de operación:

01	11bb	bfff	ffff

Descripción: Si el bit 'b' del registro 'f' es "1" se salta la instrucción siguiente ignorándola, si dicho bit es "0" se ejecuta la siguiente instrucción. Cuando salta necesita de 2 ciclos maquina en ejecutarse.

Ejemplo:
```
checa   btfss  reg,7
Cero    goto   fue_cero
Uno     instrucción x
```

CALL Lamada a subrutina.

Sintaxis: [label] bcf f,b

Operandos: $0 \leq k \leq 2047$

Operación: (PC) →TOS, k→PC

Flags afectados: Ninguno

Código de operación:

10	0kkk	kkkk	kkkk

Descripción: Salta a una subrutina, guardando el valor actual del PC en la pila y el PC adquiere el valor de la dirección a la cual se salta. Consume 2 ciclos maquina.

Ejemplo: origen call destino

Antes: (PC)= origen

Después:(PC)= destino, TOS=origen

CLRF Borra el registro 'f'.

Sintaxis: [label] clrf f

Operandos: $0 \leq f \leq 127$

Operación: 0x00 →(f), 1 →Z

Flags afectados: Z

Código de operación:

00	0001	1fff	ffff

Descripción: El contenido del registro 'f' se borra y el flag Z se pone a "1".

Ejemplo: clrf reg

Antes: reg=b'11101011'; Z= ¿?

Después: reg=b'00000000'; Z=1

CLRW Borra el registro w.

Sintaxis: [label] clrw

Operandos: Ninguno

Operación: 0x00 →(w), 1=>Z

Flags afectados: Z

Código de operación:

00	0001	0xxx	xxxx

Descripción: El registro de trabajo 'w' se carga con el valor 0x00 y el flag Z se pone a "1".

Ejemplo: clrw

Antes: w= 0x5A

Después: w=0x00; Z=1

CLRWDT Borra el WDT.

Sintaxis: [label] clrwdt

Operandos: Ninguno

Operación: 0x00 → WDT, 1 →/TO, 1 →/PD

Flags afectados: /TO, /PD

Código de operación:

00	0000	0110	0100

Descripción: Se borra el timer de watchdog (WDT). Los bits /TO, /PD del registro STATUS se ponen a "1".

Ejemplo: clrwdt

Antes: (Timer WDT)= ¿?

Después: (Timer WDT)= 0x00, /TO=1, /PD=1

COMF Complementa a f.

Sintaxis: [label] comf f,d

Operandos: $0 \le f \le 127$, $d \in [0,1]$

Operación: (/f) →(destino)

Flags afectados: Z

Código de operación:

00	1001	dfff	ffff

Descripción: Complementa el valor del registro 'f' y almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: comf reg,0

Antes: reg=b'01001101'

Después: reg=b'01001101', w=b'10110010', Z=0

DECF Decrementa a f.

Sintaxis: [label] decf f,d

Operandos: $0 \le f \le 127$, $d \in [0,1]$

Operación: (f)-1 → (destino)

Flags afectados: Z

Código de operación:

00	0011	dfff	ffff

Descripción: Se decrementa en una unidad el contenido del registro 'f' y almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: decf contador

Antes: contador=0x01

Después: contador=0x00, Z=1

DECFSZ Decrementa a f y salta si el resultado es cero.

Sintaxis: [label] decfsz f,d

Operandos: $0 \le f \le 127$, $d \in [0,1]$

Operación: (f)-1 →(destino), Salta si R=0

Flags afectados: Ninguno

Código de operación:

00	1011	dfff	ffff

Descripción: Se decrementa en una unidad el contenido del registro 'f', almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1. Si el resultado del decremento es cero se ignora la siguiente instrucción.

Ejemplo:

```
checa       decfsz  reg,0
            goto    no_es_cero
Es cero     instrucción x
No_es_cero  instrucción y
```

GOTO Salto incondicional.

Sintaxis: [label] goto k

Operandos: $0 \le k \le 2047$

Operación: k → PC<10:0>

Flags afectados: Ninguno

Código de operación:

10	1kkk	kkkk	kkkk

Descripción: Salto incondicional utilizado para dirigirse a una dirección determinada.

Ejemplo: origen goto destino

Antes: (PC)= origen

Después: (PC)= destino

INCF Decrementa a f.

Sintaxis: [label] incf f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación: (f)+1 → (destino)

Flags afectados: Z

Código de operación:

00	1010	dfff	ffff

Descripción: Se incrementa en una unidad el contenido del registro 'f' y almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: incf contador

Antes: contador=0xFF, Z= ¿?

Después: contador=0x00, Z=1

INCFSZ Incrementa a f y salta si el resultado es cero.

Sintaxis: [label] incfsz f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación: (f)+1 → (destino), Salta si R=0

Flags afectados: Ninguno

Código de operación:

00	1111	dfff	ffff

Descripción: Se incrementa en una unidad el contenido del registro 'f', almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1. Si el resultado del incremento es cero se ignora la siguiente instrucción.

Ejemplo:
```
checa      incfsz reg,0
           goto   no_es_cero
Es cero    instrucción x
No_es_cero instrucción y
```

IORLW w OR k.

Sintaxis: [label] iorlw k

Operandos: $0 \leq k \leq 255$

Operación: (w)OR(k) → (w)

Flags afectados: Z

Código de operación:

11	1000	kkkk	kkkk

Descripción: Se realiza la operación logica OR entre el contenido del registro 'w' y la constante 'k' y el resultado se guarda en el registro de trabajo 'w'.

Ejemplo: iorlw b'00110111'

Antes: w=b'10011010'

Después: w=b'10111111, Z=0

IORWF w OR f.

Sintaxis: [label] iorwf f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación: (w)OR(f) → (destino)

Flags afectados: Z

Código de operación:

00	0100	dfff	ffff

Descripción: Se realiza la operación logica OR entre el contenido del registro 'w' y el contenido del registro 'f' y almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: iorwf reg,0

Antes: w=0x91, reg=0x13, Z= ¿?

Después: w=0x93, reg=0x13, Z=0

MOVLW Mueve el valor de k a w.

Sintaxis: [label] movlw k

Operandos: $0 \leq k \leq 255$

Operación: (k) → (w)

Flags afectados: Ninguno

Código de operación:

11	00xx	kkkk	kkkk

Descripción: El valor del literal 'k' pasa al registro 'w'.

Ejemplo: movlw b'00110111'

Antes: w=b'00110111'

Después: w=b'00110111

MOVF Mueve a f.

Sintaxis: [label] movf f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación: (f) → (destino)

Flags afectados: Z

Código de operación:

00	1000	dfff	ffff

Descripción: El contenido del registro 'f' se mueve al destino 'd'. Si d=0 se almacena en 'w' y si d=1 en el registro 'f'.

Ejemplo: movf reg,0

Antes: w= ¿?, reg=0x13, Z= ¿?

Después: w=0x13, reg=0x13, Z=0

MOVWF Mueve el valor de w a f.

Sintaxis: [label] movwf f

Operandos: $0 \leq k \leq 127$

Operación: (w) → (f)

Flags afectados: Ninguno

Código de operación:

00	0000	1fff	ffff

Descripción: El contenido del registro 'w' pasa al registro 'f'.

Ejemplo: movwf reg

Antes: w=0xFF, reg= ¿?

Después: w=0xFF, reg=0xFF

NOP No operación.

Sintaxis: [label] nop

Operandos: Ninguno

Operación: No operar

Flags afectados: Ninguno

Código de operación:

00	0000	0xx0	0000

Descripción: No realiza ninguna operación lo único que hace es consumir un ciclo maquina sin hacer nada.

Ejemplo: nop

RETFIE Retorno de interrupción.

Sintaxis: [label] retfie

Operandos: Ninguno

Operación: TOS → (PC), 1 → GIE

Flags afectados: Ninguno

Código de operación:

00	0000	0000	1001

Descripción: Carga el PC con el valor que se encuentra en la parte alta de la pila (TOS) que es la dirección de retorno de la interrupcion. Se vuelven a habilitar las interrupciones.

Ejemplo: retfie

Antes: (PC)= ¿?, GIE=0

Después: (PC)= (TOS), GIE=1

RETLW Retorno y carga w con el valor de 'k'.

Sintaxis: [label] retlw k

Operandos: $0 \leq k \leq 255$

Operación: k → w, TOS → (PC)

Flags afectados: Ninguno

Código de operación:

11	01xx	kkkk	kkkk

Descripción: Se carga el registro 'w' con la constante 'k'. El PC se carga con el contenido de la cima de la pila (TOS) efectuando así un retorno de subrutina.

Ejemplo: aquí retlw 0x37

Antes: (PC)= aqui, w= ¿?

Después: (PC)= (TOS), w=0x37

RETURN Retorno de subrutina.

Sintaxis: [label] return

Operandos: Ninguno

Operación: TOS → (PC)

Flags afectados: Ninguno

Código de operación:

00	0000	0000	1000

Descripción: Carga el PC con el valor que se encuentra en la parte alta de la pila (TOS) que es la dirección de retorno de la subrutina.

Ejemplo: aquí return

Antes: (PC)= aquí

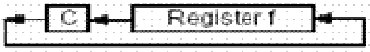
Después: (PC)= (TOS)

RLF Rota a la izquierda con el carry.

Sintaxis: [label] rlf f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación: Rota los bits de 'f' a la izquierda.



Flags afectados: C

Código de operación:

00	1101	dfff	ffff

Descripción: Los bits del registro 'f' se rotan a la izquierda, el bit mas significativo pasa al carry y el bit contenido en el carry pasa al bit menos significativo. Almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: rlf reg,0

Antes: reg=b'10101011', C=0, w= ¿?

Después: reg=b'10101011', C=1, w=b'01010110'

RRF Rota a la derecha con el carry.

Sintaxis: [label] rrf f,d

Operandos: $0 \le f \le 127$, $d \in [0,1]$

Operación: Rota los bits de 'f' a la derecha.

Flags afectados: C

Código de operación:

00	1100	dfff	ffff

Descripción: Los bits del registro 'f' se rotan a la derecha, el bit menos significativo pasa al carry y el bit contenido en el carry pasa al bit mas significativo. Almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: rrf reg,1

Antes: reg=b'10101011', C=0

Después: reg=b'01010101', C=1,

SLEEP Pasa a modo de bajo consumo.

Sintaxis: [label] sleep

Operandos: Ninguno

Operación: 0x00 → WDT, 1 → /TO, 0 → WDT Prescaler, 0 → /PD

Flags afectados: /PD, /TO

Código de operación:

00	0000	0110	0011

Descripción: El micro pasa a modo de bajo consumo con la parada del oscilador y del Timer 0. Se puede salir de este estado por:

- Activación del pin /MCLR (reset).
- Por una interrupción que no sea TMR0.

Ejemplo: sleep

Antes: WDT= ¿?, /TO= ¿?, /PD= ¿?

Después: WDT= 0x00, /TO= 1, /PD= 0

SUBLW Resta k-w.

Sintaxis: [label] sublw k

Operandos: $0 \le k \le 255$

Operación: (k)-(w) → (w)

Flags afectados: C, DC, Z

Código de operación:

11	110x	kkkk	kkkk

Descripción: Resta a la constante 'k' el contenido del registro 'w' usando complemento a 2 y el resultado se almacena en w.

Ejemplos: sublw 0x0A

Antes 1: w=0x04, C= ¿?, Z= ¿?

Después1 : w=0x06, C=1, Z=0

Antes 2: w=0x0B, C= ¿?, Z= ¿?

Después 2: w=0xFF, C=0, Z=0

Antes 3: w=0x0A, C= ¿?, Z= ¿?

Después 3: w=0x00, C=1, Z=1

SUBWF Resta f-w.

Sintaxis: [label] subwf f,d

Operandos: $0 \le f \le 127$, $d \in [0,1]$

Operación: (f)-(w) → (destino)

Flags afectados: C, DC, Z

Código de operación:

00	0010	dfff	ffff

Descripción: Resta al contenido del registro 'f' el contenido del registro 'w' usando complemento a 2 y el resultado se almacena en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplos: subwf reg,1

Antes 1: reg= 0x03, w=0x02, C= ¿?, Z= ¿?

Después1: reg= 0x01, w=0x02, C=1, Z=0

Antes 2: reg= 0x03, w=0x03, C= ¿?, Z= ¿?

Después 2 reg= 0x00, w=0x03, C=1, Z=1

Antes 3: reg= 0x01, w=0x02, C= ¿?, Z= ¿?

Después 3 w=0xFF, C=0, Z=0

SWAPF Intercambia los nibbles de f.

Sintaxis: [label] swapf f,d

Operandos: $0 \le f \le 127$, $d \in [0,1]$

Operación: (f<3:0>) → (d<7:4>), (d<7:4>) → (f<3:0>)

Flags afectados: Ninguno

Código de operación:

00	1110	dfff	ffff

Descripción: Los 4 bits de mas peso del registro 'f' se intercambian con los 4 bits de menos peso del mismo registro y el resultado se almacena en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: swapf reg,0

Antes: reg= b'11110000', w= ¿?

Después: reg= b'11110000', w=b'00001111'

XORLW w XOR k.

Sintaxis: [label] xorlw k

Operandos: $0 \le k \le 255$

Operación: (w)XOR(k) → (w)

Flags afectados: Z

Código de operación:

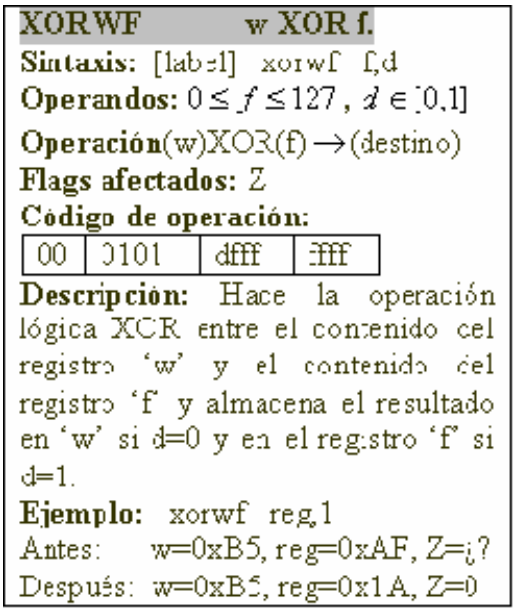
11	1010	kkkk	kkkk

Descripción: Hace la operación lógica XOR entre el contenido del registro 'w' y el valor de 'k', el resultado se almacena en 'w'.

Ejemplo: xorlw b'00101010'

Antes: Z=¿?, w=b'10001011'

Después: Z=0, w=b'10100001'



XORWF w XOR f.

Sintaxis: [label] xorwf f,d

Operandos: $0 \leq f \leq 127$, $d \in [0,1]$

Operación (w)XOR(f) → (destino)

Flags afectados: Z

Código de operación:

00	0101	dfff	ffff

Descripción: Hace la operación lógica XOR entre el contenido del registro 'w' y el contenido del registro 'f' y almacena el resultado en 'w' si d=0 y en el registro 'f' si d=1.

Ejemplo: xorwf reg,1

Antes: w=0xB5, reg=0xAF, Z=¿?

Después: w=0xB5, reg=0x1A, Z=0

I.10 Modos de Direccionamiento.

Las instrucciones del PIC 16F84A pueden especificar los datos u operandos mediante cuatro modos de direccionamiento:

- *Direccionamiento inmediato.* El valor del dato (constante) lo contiene el mismo código de operación, el cual se guarda en el registro w para su posterior procesamiento. Ejemplo: movlw constante
- *Direccionamiento Directo.* La dirección de la memoria RAM la contiene el mismo código de operación: movwf registro_x
- *Direccionamiento de bit.:* La dirección del dato es un bit. Ejemplo: bcf STATUS,RP0.
- *Direccionamiento indirecto.* La dirección del dato se encuentra en el registro INDF. Cada vez que se utiliza este registro se utiliza el contenido del registro FSR para direccionar al operando por ejemplo si el registro FSR=16h y en el programa a ejecutar aparece la instrucción *movwf INDF* se esta indicando que el contenido del registro *w* se carga en la posición de memoria 16h. A este procedimiento se le llama direccionamiento indirecto.

Apéndice II “Subrutina del Bus 12C”.

```
; Estas subrutinas permiten realizar las tareas básicas de control del bus serie I2C,
; por parte de un solo microcontrolador maestro.
; ZONA DE DATOS
;
        CBLOCK
        I2C_ContadorBits        ; Cuenta los bits a transmitir o a recibir.
        I2C_Dato                ; Dato a transmitir o recibido.
        I2C_Flags               ; Guarda la información del estado del bus I2C.
        ENDC

#DEFINE I2C_UltimoByteLeer      I2C_Flags,0
;   - (I2C_UltimoByteLeer)=0, NO es el último byte a leer por el maestro.
;   - (I2C_UltimoByteLeer)=1, SÍ es el último byte a leer por el maestro.

; La definición de las líneas SCL y SDA del bus I2C se puede cambiar según las
; necesidades del hardware.

#DEFINE SCL     PORTA,3                 ; Línea SCL del bus I2C.
#DEFINE SDA     PORTA,4                 ; Línea SDA del bus I2C.
;
; Subrutina "SDA_Bajo" ------------------------------------------------------------------------
---
;
SDA_Bajo
        bsf     STATUS,RP0      ; Configura la línea SDA como salida.
        bcf     SDA
        bcf     STATUS,RP0
        bcf     SDA             ; SDA en bajo.
        return
;
; Subrutina "SDA_AltaImpedancia" --------------------------------------------------------
;
SDA_AltaImpedancia
        bsf     STATUS,RP0      ; Configura la línea SDA entrada.
        bsf     SDA             ; Lo pone en alta impedancia y, gracias a la
        bcf     STATUS,RP0      ; Rp de esta línea, se mantiene a nivel alto.
        return
;
; Subrutina "SCL_Bajo" ------------------------------------------------------------
;
SCL_Bajo
        bsf     STATUS,RP0
        bcf     SCL             ; Configura la línea SCL como salida.
        bcf     STATUS,RP0
        bcf     SCL             ; La línea de reloj SCL en bajo.
        return
```

```
; Subrutina "SCL_AltaImpedancia" -------------------------------------------------------
;
SCL_AltaImpedancia
        bsf     STATUS,RP0          ; Configura la línea SCL entrada.
        bsf     SCL                 ; Lo pone en alta impedancia y, gracias a la Rp
        bcf     STATUS,RP0          ; de esta línea, se mantiene a nivel alto.
SCL_EsperaNivelAlto
        btfss   SCL                 ; Si algún esclavo mantiene esta línea en bajo
        goto    SCL_EsperaNivelAlto         ; hay que esperar.
        return
;
; Subrutina "I2C_EnviaStart" -----------------------------------------------------------
;
; Esta subrutina envía una condición de Start o inicio.
;
I2C_EnviaStart
        call    SDA_AltaImpedancia          ; Línea SDA en alto.
        call    SCL_AltaImpedancia; Línea SCL en alto.
        call    Retardo_4micros             ; Tiempo tBUF del protocolo.
        call    SDA_Bajo            ; Flanco de bajada de SDA mientras SCL está alto.
        call    Retardo_4micros             ; Tiempo tHD;STA del protocolo.
        call    SCL_Bajo            ; Flanco de bajada del reloj SCL.
        call    Retardo_4micros
        return
;
; Subrutina "I2C_EnviaStop" ------------------------------------------------------------
;
; Esta subrutina envía un condición de Stop o parada.
;
I2C_EnviaStop
        call    SDA_Bajo
        call    SCL_AltaImpedancia; Flanco de subida de SCL.
        call    Retardo_4micros             ; Tiempo tSU;STO del protocolo.
        call    SDA_AltaImpedancia          ; Flanco de subida de SDA.
        call    Retardo_4micros             ; Tiempo tBUF del protocolo.
        return
;
; Subrutina "I2C_EnviaByte" ------------------------------------------------------------
;
; El microcontrolador maestro transmite un byte por el bus I2C, comenzando por el bit
; MSB. El byte a transmitir debe estar cargado previamente en el registro de trabajo W.
; De la subrutina ejecutada anteriormente I2C_EnviaStart o esta misma I2C_EnviaByte,
; la línea SCL se debe encontrar a nivel bajo al menos durante 5 µs.
;
I2C_EnviaByte
        movwf I2C_Dato                      ; Almacena el byte a transmitir.
        movlw 0x08                  ; A transmitir 8 bits.
        movwf I2C_ContadorBits
```

```
I2C_EnviaBit
        rlf     I2C_Dato,F          ; Chequea el bit, llevándolo previamente al Carry.
        btfsc   STATUS,C
        goto    I2C_EnviaUno
I2C_EnviaCero
        call    SDA_Bajo            ; Si es "0" envía un nivel bajo.
        goto    I2C_FlancoSCL
I2C_EnviaUno
        call    SDA_AltaImpedancia          ; Si es "1" lo activará a alto.
I2C_FlancoSCL
        call    SCL_AltaImpedancia; Flanco de subida del SCL.
        call    Retardo_4micros             ; Tiempo tHIGH del protocolo.
        call    SCL_Bajo            ; Termina el semiperiodo positivo del reloj.
        call    Retardo_4micros             ; Tiempo tHD;DAT del protocolo.
        decfsz  I2C_ContadorBits,F          ; Lazo para los ocho bits.
        goto    I2C_EnviaBit
;
        call    SDA_AltaImpedancia          ; Libera la línea de datos.
        call    SCL_AltaImpedancia; Pulso en alto de reloj para que el esclavo
        call    Retardo_4micros             ; pueda enviar el bit ACK.
        call    SCL_Bajo
        call    Retardo_4micros
        return
;
; Subrutina "I2C_LeeByte" ---------------------------------------------------------
;
; El microcontrolador maestro lee un byte desde el esclavo conectado al bus I2C. El dato
; recibido se carga en el registro I2C_Dato y lo envía a la subrutina superior a través
; del registro W. Se empieza a leer por el bit de mayor peso MSB.
; De alguna de las subrutinas ejecutadas anteriormente I2C_EnviaStart, I2C_EnviaByte
; o esta misma I2C_LeeByte, la línea SCL lleva en bajo al menos 5 µs.

I2C_LeeByte
        movlw 0x08                  ; A recibir 8 bits.
        movwf I2C_ContadorBits
        call    SDA_AltaImpedancia          ; Deja libre la línea de datos.
I2C_LeeBit
        call    SCL_AltaImpedancia; Flanco de subida del reloj.
        bcf     STATUS,C            ; En principio supone que es "0".
        btfsc   SDA                 ; Lee el bit
        bsf     STATUS,C            ; Si es "1" carga 1 en el Carry.
        rlf     I2C_Dato,F          ; Lo introduce en el registro.
        call    SCL_Bajo            ; Termina el semiperiodo positivo del reloj.
        call    Retardo_4micros             ; Tiempo tHD;DAT del protocolo.
        decfsz  I2C_ContadorBits,F          ; Lazo para los 8 bits.
        goto    I2C_LeeBit
```

```
; Chequea si este es el último byte a leer para enviar o no el bit de reconocimiento
; ACK en consecuencia.
;
        btfss   I2C_UltimoByteLeer ; Si es el último, no debe enviar
                                   ; el bit de reconocimiento ACK.
        call    SDA_Bajo           ; Envía el bit de reconocimiento ACK
                                   ; porque todavía no es el último byte a leer.
        call    SCL_AltaImpedancia; Pulso en alto del SCL para transmitir el
        call    Retardo_4micros            ; bit ACK de reconocimiento. Este es
tHIGH.
        call    SCL_Bajo           ; Pulso de bajada del SCL.
        call    Retardo_4micros
        movf    I2C_Dato,W         ; El resultado se manda en el registro de
        return                     ; de trabajo W.
```

Apéndice III “Subrutina delPCF8574”.

```
; Estas subrutinas permiten realizar las tareas de manejo del expansor de bus I2C
PCF8574
; que convierte un bus de 8 líneas paralelo a bus serie I2C y viceversa.
; Como es posible que haya más de un expansor en el mismo circuito, se deja a la
subrutina
; de llamada a éstas que determine las direcciones de escritura y lectura mediante las
; etiquetas "PCF8574_DireccionLectura" y "PCF8574_DireccionEscritura", atendiendo
a las
; características del hardware utilizado.
;
; ZONA DE DATOS  CBLOCK
        PCF8574_Dato                ; Aquí guarda el resultado del dato leído o que
        ENDC                        ; se va a escribir.
; Subrutina "PCF8574_Lee" ----------------------------------------------------------
;
; Lee el periférico conectado al bus paralelo de 8 líneas del PCF8574 que actúa como
; entrada y el resultado lo proporciona en el registro de trabajo W.

PCF8574_Lee
        call    I2C_EnviaStart              ; Envía la condición de Start.
        movlw PCF8574_DireccionLectura  ; Apunta al expansor de lectura.
        call    I2C_EnviaByte
        bsf     I2C_UltimoByteLeer
        call    I2C_LeeByte         ; Lee el puerto.
        movwf PCF8574_Dato                  ; Lo guarda en el registro correspondiente.
        call    I2C_EnviaStop               ; Acaba de leer.
        movf    PCF8574_Dato,W              ; Recupera el valor del dato leído.
        return

; Subrutina "PCF8574_Escribe" ------------------------------------------------------
;
; Escribe en el periférico conectado bus paralelo de 8 líneas del expansor PCF8574 que
; actúa como salida, el dato que le llega por el registro de trabajo W. El dato escrito
; es recuperado finalmente en el registro W.
;
PCF8574_Escribe
        movwf PCF8574_Dato              ; Guarda el dato a escribir.
        call    I2C_EnviaStart              ; Envia condición de Start.
        movlw PCF8574_DireccionEscritura        ; Apunta al expansor que actúa
como salida.
        call    I2C_EnviaByte
        movf    PCF8574_Dato,W              ; Recupera el valor del dato a enviar.
        call    I2C_EnviaByte               ; Envía el dato.
        call    I2C_EnviaStop               ; Termina.
        movf    PCF8574_Dato,W              ; Recupera el valor del dato escrito.
        return
```

BIBLIOGRAFIA ESCRITA.

- Palacios Municio Enrique, Remiro Domínguez Fernando, López Pérez Lucas. Microcontrolador PIC16F84. Desarrollo de proyectos. Editorial Alfaomega.

- *PIC16F84. Data Sheet. Microchip Technology Inc.*

- *J.* Mª. Angulo Usategui, E. Martín Cuenca, I. Angulo Martínez. Microcontroladores PIC diseño práctico de aplicaciones. Editorial Mc Graw Hill.

- Vallejo Horacio. Proyectos con microcontroladores PIC y PICAXE. Colección "Club saber electrónica No 7". Editorial Quark.

- Vallejo Horacio. Curso de PICS para estudiantes y aficionados. Colección "Club saber electrónica No 20". Editorial Quark.

- Vallejo Horacio. Microcontroladores PIC. Programación y desarrollo. Colección "Club saber electrónica No 24". Editorial Quark.

- *J.* Mª. Angulo Usategui, E. Martín Cuenca, I. Angulo Martínez. Microcontroladores PIC. La Solución en un Chip. Ed. Paraninfo. (1997)

- *4N25. Data Sheet. Motorola*

- *MPASM USER'S GUIDE*

- *PCF8574.Data Sheet. Philps semiconductors.*

BIBLIOGRAFÍA ELECTRÓNICA.

- Microchip. http://www.microchip.com
- MPLAB. http://www.microchip.com/10/Tools/mTools/MPLAB/index.htm
- IC-PROG. www.ic-prog.com.
- Philips semiconductors. www.semiconductors.philips.com
- Parallax. http://www.parallaxinc.com
- El Rincón del Pic. http://members.es.tripod.de/~InfoE/infop.htm
- Rei Project: Mod Chip: Algunos proyectos. http://chip.aeug.org
- Microcontroladores: Información, Herramientas y Programador. http://www.geocities.com/TheTropics/2174/micro.html
- Microcontroladores, autómatas, electrónica, etc http://www.jmengual.com/
- Pic Programming. Getting Started: 4 pasos para empezar con los Pic. http://www.pp.clinet.fi/~newmedia/pic/index.html