**UNIVERSIDAD DE EL SALVADOR**

**FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE**

**DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA.**

**TRABAJO DE GRADUACIÓN DENOMINADO:**

"DISEÑO DE GUÍAS DE TRABAJO Y CONSTRUCCIÓN DE EQUIPO DIDÁCTICO PARA LA IMPLANTACIÓN DE PRÁCTICAS DE LABORATORIO CON MICRO CONTROLADORES EN LA CARRERA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS DE LA FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE."

**PARA OPTAR AL GRADO DE:**
**INGENIERO DE SISTEMA INFORMÁTICOS**

**PRESENTAN:**
**FRANCIA ESCOBAR, ROBERTO ANTONIO**
**GARCÍA, JUAN CARLOS**
**UMAÑA ORDOÑEZ, JORGE ARTURO**

**DOCENTE DIRECTOR**
**ING. JOSE FRANCISCO ANDALUZ**

**NOVIEMBRE, 2007.**

**SANTA ANA EL SALVADOR CENTRO AMÉRICA**

# UNIVERSIDAD DE EL SALVADOR

**RECTOR**

MÁSTER RUFINO ANTONIO QUEZADA SÁNCHEZ

**VICERRECTOR ACADÉMICO**

MÁSTER MIGUEL ÁNGEL PÉREZ RAMOS

**VICE RECTOR ADMINISTRATIVO**

MÁSTER ÓSCAR NOÉ NAVARRETE

**SECRETARIO GENERAL**

LICENCIADO DOUGLAS VLADIMIR ALFARO CHÁVEZ

**FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE**

**DECANO**

LIC. JORGE MAURICIO RIVERA

**VICE DECANO**

LIC. ELADIO ZACARÍAS ORTEZ

**SECRETARIO**

LIC. VÍCTOR HUGO MERINO QUEZADA

**JEFE DE DEPARTAMENTO DE INGENIERÍA**

ING. RENÉ ERNESTO MARTÍNEZ BERMÚDEZ

# AGRADECIMIENTOS

**A DIOS TODOPODEROSO**

Por permitir que llegara hasta el final de la carrera, por no dejarme solo en este camino y siempre levantarme cuando necesite de su apoyo y fuerza para continuar adelante.

**A MI MADRE ÁNGELA VICTORIA ESCOBAR DE FRANCIA**

Por su apoyo, paciencia y ser un pilar en mi vida; sin la cual no hubiese podido culminar la carrera., le dedico este triunfo con las palabras con las que siempre me ha dado confianza y fuerza de seguir adelante "se triunfa cuando se persevera".

**A MI PADRE JOSÉ ANTONIO FRANCIA ESCOBAR**

Que su ejemplo formo en mi la idea de siempre mirar más adelante, seguir luchando y creer que siempre es posible superarse cada día más; gracias por su inmenso apoyo desde todos los puntos de mi carrera y mi vida, como padre, docente, asesor y amigo.

**A MIS HERMANOS JOSÉ ALEXANDER Y CLAUDIA LISSETTE**

Por brindarme su apoyo, paciencia y cariño a lo largo de mi formación como profesional, gracias hermanos por siempre creer en mí y darme su confianza cuando más lo he necesitado.

**A ROXANA PATRICIA PALENCIA**

Iniciamos juntos un camino al inicio de nuestras carreras y le agradezco por ser un apoyo constante en mi vida, por estar ahí siempre en las buenas y en las malas.

**A MIS COMPAÑEROS**

Por brindarme su confianza en este proyecto y por toda la ayuda que mutuamente ha logrado culminar este trabajo de Grado.

**A MIS FAMILIARES**

Por creer siempre que era posible terminar y lograr este triunfo.

**A MIS AMIGOS**

Que me han apoyado con su confianza y cariño durante todo este proceso, que siempre tuvieron palabras de aliento, consejos y regaños; a todos ellos les agradezco inmensamente.

*ROBERTO ANTONIO FRANCIA ESCOBAR*

# AGRADECIMIENTOS


## A DIOS TODOPODEROSO

Por permitirme lograr una meta más en mi vida, por darme las fuerzas para seguir adelante y no dejarme vencer nunca, por darme infinitas bendiciones y sabiduría para hacer lo que me propongo siempre.

## A MI MADRE ANA ELIZABETH GARCÍA MINEROS

Por su inmenso apoyo y paciencia, por creer en mí y en mis sueños, porque sin su comprensión no podría haber logrado esta meta, por todo el esfuerzo que hizo para ayudarme a salir adelante, por sus consejos que me alientan a seguir luchando.

## A MI HERMANA FLOR DE MARÍA GARCÍA

Con cariño por su apoyo y confianza en mí, por motivarme a salir adelante durante toda mi carrera.

## A MIS COMPAÑEROS

Por toda la paciencia que tuvieron y por confiar en mí.

## A MIS FAMILIARES

Por su confianza en mi capacidad, por su estimulo para la culminación de mi carrera.

## A MIS AMIGOS

Que desinteresadamente me ayudaron y apoyaron durante todo este proceso, que con su buena voluntad, por hacer que cada pedazo de tiempo fuera ameno. No voy a olvidar sus consejos, enseñanzas, amistad y cariño me motivaron a seguir.

*JUAN CARLOS GARCÍA.*

# AGRADECIMIENTOS


## A DIOS TODOPODEROSO

Por mantenerme en el camino correcto y darme toda la ayuda necesaria para no abandonar este proyecto, parte tan importante en mi futuro, y por dejarme estar cerca de mis seres queridos dándome siempre su apoyo.

## A MIS PADRES

Que sin ellos esto nunca hubiera podido completarse, que me han apoyado durante toda mi vida estando siempre a mi lado. Gracias a ellos nunca me he sentido solo.

## A MIS HERMANOS

Que han estado presentes para apoyarme y acompañarme en todo momento. Han luchado conmigo durante todo el camino.

## A MI NOVIA

Siempre ha sido la perfecta compañía haciendo muchas veces a un lado sus proyectos para dedicar tiempo a los míos. Dando siempre lo mejor de si.

## A MIS FAMILIARES

Que me han facilitado las cosas con sus conocimientos, experiencias y oraciones. He aprendido mucho de todos y cada uno de ellos.

## A MIS AMIGOS

Que han estado conmigo en los momentos felices y tristes de mi vida. Han sido parte importante en este proceso, siempre he podido contar con ellos.

## A MIS COMPAÑEROS DE TESIS

Gracias a sus esfuerzos y cualidades hemos podido lograrlo.

JORGE ARTURO UMAÑA ORDÓÑEZ

# Contenido

# Capitulo 1

Generalidades

## 1.1 INTRODUCCIÓN.

Los Micro controladores se han convertido en una parte integral de nuestras vidas, hasta el punto que la mayoría de los productos electrónicos con los que tenemos contacto a diario tienen por lo menos un Micro controlador dentro de ellos. Los Electrodomésticos hacen uso de ellos para el control de un motor, así como unidad central de procesamiento. Los teléfonos celulares no serian posibles sin ellos.

Las extensas áreas de aplicación de los Micro controladores, que se pueden considerar ilimitadas, exigen a los estudiantes de la carrera de Ingeniería de Sistemas Informáticos estar familiarizados con el funcionamiento y programación de los Micro controladores. Pero aprender a manejar y programar Micro controladores sólo se consigue desarrollando prácticamente diseños reales.

Es por eso, que surge la motivación de diseñar y elaborar guías de trabajo para desarrollar prácticas dentro de la asignatura de Microprogramación, así como proporcionar el equipo adecuado para la realización de dichas prácticas. De esta forma los alumnos de la carrera de Ingeniería de Sistemas Informáticos tendrán la posibilidad de adquirir estos conocimientos que le serán de mucha utilidad en el transcurso de su vida.

En el presente documento se muestran las generalidades del trabajo de grado, tales como objetivos, el planteamiento del problema, la justificación y alcances.

## 1.2 OBJETIVOS

### 1.2.1 OBJETIVO GENERAL.

Elaborar un diseño y construcción de guías de trabajo y equipo didáctico para la implementación de prácticas de laboratorio con Micro controladores (Circuitos Integrados Programables PICs) para la asignatura de Microprogramación que se imparte en la carrera de Ingeniería de Sistemas Informáticos de la Facultad Multidisciplinaria de Occidente.

### 1.2.2 OBJETIVOS ESPECÍFICOS

- Realizar una investigación detallada de los Micro controladores y modelos que se utilizan en la enseñanza.

- Diseñar e implementar de guías de trabajo y material didáctico de instructoría, orientado al estudio y pruebas de Micro controladores.

- Diseñar y construir equipo didáctico para realización de prácticas básicas en el laboratorio.

- Elaborar un documento que sirva para ser utilizado por la comunidad educativa como una referencia teórica básica que facilite el aprendizaje y la formación acerca de los Micro controladores.

- Implementar las prácticas de laboratorios de Micro controladores para contribuir a que los estudiantes de la asignatura puedan desarrollar, dirigir y participar en proyectos de investigación y desarrollo tecnológico en el área de la electrónica digital con Micro controladores.

- Impulsar el uso de software en el diseño de sistemas digitales con Micro controladores.

## 1.3 PLANTEAMIENTO DEL PROBLEMA

El avance de la tecnología ha tomado auge a nivel mundial, originando una mayor demanda en la formación de profesionales, para poder incursionar en el mercado productivo, por lo que se hace necesario poseer conocimientos prácticos para tener un mayor desenvolvimiento en las organizaciones.

Los estudiantes universitarios son los profesionales del mañana, es importante que esos futuros profesionales deben de ser bien formados para lograr así uno de los principales objetivos de nuestra alma mater, por ello es importante que se brinden todos los medios posibles para el satisfactorio aprendizaje de los educandos, actualmente el Departamento de Ingeniería de la Facultad Multidisciplinaria de Occidente no cuenta con los recursos necesarios para realizar los laboratorios prácticos acerca de Micro controladores tema correspondiente al hardware de la carrera de ingeniería en sistemas que allí se sirve.

Con los cambios tecnológicos, crece la necesidad de proveer medios y equipo necesarios con el cual los estudiantes puedan realizar sus laboratorios prácticos de la asignatura de microprogramación de la especialidad del área de hardware de la carrera de ingeniería en sistemas informáticos.

Con lo descrito anteriormente se observan los siguientes problemas:

- El Departamento de Ingeniería no cuenta con los medios y equipos para poder realizar los laboratorios prácticos acerca del uso de Micro controladores y sus aplicaciones que apoyen los conceptos teóricos de la asignatura de Microprogramación de la especialidad en el área de hardware de la carrera Ingeniería de Sistemas Informáticos.

- A partir del año 2004 se contó con un Auxiliar de Cátedra que solamente cubrió las asignaturas de Sistemas Digitales I y Arquitectura de Computadores (no en Microprogramación), limitándose a la elaboración de guías de prácticas para las asignaturas antes mencionadas con las que se complementaban algunos de los conocimientos básicos teóricos expuestos en clase, en contenidos referentes al hardware, pero no existen los recursos didácticos y las guías de trabajo para poder llevar acabo los laboratorios prácticos de los Micro controladores y sus aplicaciones en el programa de estudio de la asignatura.

- La aportación de la asignatura al perfil de los egresados, se ve limitada en su participación para el desarrollo de las capacidades y competencias en el perfil de un Ingeniero informático, ocasionando que los profesionales se desempeñen en el ámbito laboral con ciertas deficiencias en el área de hardware; ya que no existen, cursos prácticos con Micro controladores, dentro de la facultad multidisciplinaria de occidente que da como resultado un déficit en el área técnica práctica de la carrera de Ingeniería de Sistemas Informáticos.

Debido a los problemas antes mencionados, se presenta una propuesta que permita al Departamento de Ingeniería y Arquitectura implantar un Laboratorio práctico de Hardware acerca de los Micro controladores, que cubra las necesidades de clases prácticas en la asignatura de microprogramación de la especialidad de dicha carrera.

## 1.4 JUSTIFICACIÓN.

Con el ’’Diseño de guías de trabajo y construcción de equipo didáctico para la implementación de prácticas de laboratorio con Micro controladores para la asignatura de Microprogramación, que se imparte en la carrera de Ingeniería de Sistemas Informáticos de la Facultad Multidisciplinaria de Occidente’’ se busca solucionar en gran medida muchas de las deficiencias en la asimilación de los conocimientos de las distintas aplicaciones en el mundo real de este tipo de dispositivo electrónico; ya que se aborda únicamente de forma teórica y simulaciones basadas en software que actualmente presentan los estudiantes de las carreras de ingeniería de la Facultad Multidisciplinaria de Occidente, no tomándose en cuenta las practicas de laboratorio con Micro controladores reales.

Con esta propuesta se persigue:

- Desarrollar la creatividad del estudiante evidenciando la integración de los nuevos saberes y la reutilización de los previos, a través de la utilización de los recursos didácticos.

- Propiciar experiencias educativas que favorezcan la producción del conocimiento en el mediano y largo plazo.

- La utilización de los recursos como herramientas fuertemente motivadoras para el aprendizaje de los sistemas digitales con Micro controladores.

Es importante que el alumno tenga una visión y experiencia en el trabajo con Micro controladores, ya que es común encontrarse con ellos en todo sistema automatizado en la industria, que en un momento determinado de la aplicación de la carrera pueda necesitar un análisis técnico y resolución de problemáticas.

Otra razón importante para el diseño de guías de trabajo y construcción de equipo didáctico para la implementación de prácticas de laboratorio con Micro controladores para la asignatura de Microprogramación, es la responsabilidad de la facultad de mantenerse a la vanguardia de la enseñanza con el uso de la tecnología en la educación.

## 1.5 ALCANCES.

- Investigación sobre los Micro controladores y establecer los modelos que puedan aportar mas facilidad de enseñanza didáctica y aplicación a prácticas de laboratorio.

- Diseño y elaboración de guías para alumnos, instructores y material didáctico.

- Equipo de práctica para el desarrollo de guías de trabajo; como lo son:

    - 5 equipos básicos de entrenamiento

        - Diseño.

        - Documentación.

        - Hardware (Equipo construido).

    - 5 quemadores o grabadoras con diseño DM

        - Diseño.

        - Documentación.

        - Hardware (Equipo construido).

- Manual de usuario y guías de trabajo sobre el software para el uso de los equipos:
    - MPLAB
    - WINPIC
    - PROTEUS

## 1.6 BENEFICIOS ESPERADOS.

Los resultados esperados al finalizar el proyecto de ''Diseño de guías de trabajo y construcción de equipo didáctico para la implementación de prácticas de laboratorio con Micro controladores para la asignatura de Microprogramación, que se imparte en la carrera de Ingeniería de Sistemas Informáticos de la Facultad Multidisciplinaria de Occidente'', para las diferentes entidades de la Facultad son:

**a) Para el Departamento de Ingeniería y Arquitectura:**

- Diseño completo de toda la documentación necesaria para la implantación de un laboratorio para el aprendizaje practico del uso de los Micro controladores y sus aplicaciones, específicamente un conjunto de guías de trabajo para desarrollar practicas de los laboratorios acerca de estos, normas utilizadas dentro del laboratorio, sus precauciones y procedimientos para la utilización del equipo didáctico.

- Diseño y creación de equipo didáctico (Grabadores y Entrenador) a utilizar para desarrollar las guías de los laboratorios prácticos de los Micro controladores; que apoye al proceso de enseñanza - aprendizaje de los estudiantes de la materia de Microprogramación en el momento que lo necesiten.

**b) Para la Jefatura del Departamento de Ingeniería y Arquitectura:**

- Toda la documentación necesaria para la realización de las prácticas de laboratorio de Micro controladores para el aprendizaje practico del uso de los Micro controladores y sus aplicaciones.

**c) Para los estudiantes del Departamento de Ingeniería y Arquitectura.**

- La utilización de los recursos (Grabadores, Entornos de desarrollo, Simuladores) como herramientas fuertemente motivadoras para el aprendizaje de los sistemas digitales con Micro controladores; que es atractiva y que apoye eficientemente el proceso de Enseñanza - Aprendizaje de los estudiantes para una mayor comprensión de temas en la asignatura de microprogramación.

**d) Para las autoridades de la Facultad Multidisciplinaria de Occidente.**

- Toda la documentación necesaria para la realización de las prácticas de laboratorio de Micro controladores para el aprendizaje practico del uso de los Micro controladores y sus aplicaciones. Que sirva de apoyo al Proceso de Enseñanza Aprendizaje de los estudiantes del Departamento de Ingeniería y Arquitectura, que colabore con la Misión y la Visión de la Universidad de El Salvador.

## 1.7 LIMITANTES.

- La disponibilidad de recursos de Micro controladores dentro del país no es inmediata por lo tanto su adquisición se dificulta.

- La carencia de recursos en los establecimientos se traduce en precios de venta elevados, lo que conlleva a comprarlos en el exterior, aun cuando el precio de venta en otros países es bajo a de tener que cancelarse gastos de envió e impuestos para su ingreso al país.

# Capitulo 2

Marco Teórico

## 2.1 CONTROLADOR Y MICRO CONTROLADOR.

Recibe el nombre de controlador el dispositivo que se emplea para el gobierno de uno o varios procesos. Por ejemplo, el controlador que regula el funcionamiento de un horno dispone de un sensor que mide constantemente su temperatura interna y, cuando traspasa los límites prefijados, genera las señales adecuadas que accionan los efectores que intentan llevar el valor de la temperatura dentro del rango estipulado.

Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado frecuentemente. Hace tres décadas, los controladores se construían exclusivamente con componentes de lógica discreta, posteriormente se emplearon los microprocesadores, que se rodeaban con chips de memoria y E/S sobre una tarjeta de circuito impreso. En la actualidad, todos los elementos del controlador se han podido incluir en un chip, el cual recibe el nombre de Micro controlador. Realmente consiste en un sencillo pero completo computador contenido en el corazón (chip) de un circuito integrado.

Los '**PIC'** son una familia de Micro controladores tipo RISC fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instruments.

El nombre actual no es un acrónimo. En realidad, el nombre completo es **PICmicro**, aunque generalmente se utiliza como ***P****eripheral* ***I****nterface* ***C****ontroller* (Controlador de Interfaz Periférico).

El PIC original se diseñó para ser usado con la nueva UCP de 16 bits CP16000. Siendo en general una buena UCP, ésta tenía malas prestaciones de

E/S, y el PIC de 8 bits se desarrolló en 1975 para mejorar el rendimiento del sistema quitando peso de E/S a la UCP. El PIC utilizaba micro código simple almacenado en ROM para realizar estas tareas; y aunque el término no se usaba por aquel entonces, se trata de un diseño RISC que ejecuta una instrucción cada 4 ciclos del oscilador.

En 1985, dicha división de microelectrónica de General Instruments se convirtió en una filial y el nuevo propietario canceló casi todos los desarrollos, que para esas fechas la mayoría estaban obsoletos. El PIC, sin embargo, se mejoró con EPROM para conseguir un controlador de canal programable. Hoy en día multitud de PICs vienen con varios periféricos incluidos (módulos de comunicación serie, UARTs, núcleos de control de motores, etc.) y con memoria de programa desde 512 a 32.000 palabras (una *palabra* corresponde a una instrucción en ensamblador, y puede ser 12, 14 o 16 bits, dependiendo de la familia específica de PICmicro).

Un Micro controlador es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de un ordenador: CPU, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado. Aunque sus prestaciones son limitadas, además de dicha integración, su característica principal es su alto nivel de especialización. Aunque los hay del tamaño de un sello de correos, lo normal es que sean incluso más pequeños, ya que, lógicamente, forman parte del dispositivo que controlan.

Es un microprocesador optimizado para ser utilizado para controlar equipos electrónicos. Los Micro controladores representan la inmensa mayoría de los chips de ordenadores vendidos, sobre un 50% son controladores "simples" y el restante corresponde a dsPICs más especializados. Mientras se

pueden tener uno o dos microprocesadores de propósito general en casa (Usted está usando uno para leer esto), usted tiene probablemente distribuido entre los electrodomésticos de su hogar una o dos docenas de Micro controladores. Pueden encontrarse en casi cualquier dispositivo eléctrico como automóviles, lavadoras, hornos microondas, teléfonos, etc.


**El diagrama del sistema de un micro controlador**

Los dispositivos de entrada pueden ser un teclado, un interruptor, un sensor, etc.

Los dispositivos de salida pueden ser Leds, pequeños parlantes, zumbadores, interruptores de potencia (transistores, opto acopladores), u otros dispositivos como relés, luces, un secador de pelo. Aquí tienes una representación en bloques del Micro controlador, para que te des una idea, y puedes ver que lo adaptamos tal y cual es un ordenador, con su fuente de alimentación, un circuito de reloj y el chip Micro controlador, el cual dispone de su CPU, sus memorias, y por supuesto, sus puertos de comunicación listos para conectarse al mundo exterior.

Un Micro controlador difiere de una CPU normal, debido a que es más fácil convertirla en un ordenador en funcionamiento, con un mínimo de chips externos de apoyo. La idea es que el chip se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por otros chips.

Por ejemplo, un Micro controlador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria RAM y ROM/EPROM/EEPROM, significando que para hacerlo funcionar, todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los Micro controladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidores de analógico a digital, temporizadores, UARTs y buses de interfaz serie especializados, como $I^2C$ y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados. Los modernos Micro controladores frecuentemente incluyen un lenguaje de programación integrado, como el BASIC que se utiliza bastante con este propósito.

Los Micro controladores negocian la velocidad y la flexibilidad para facilitar su uso. Debido a que se utiliza bastante sitio en el chip para incluir funcionalidad, como los dispositivos de entrada/salida o la memoria que incluye el Micro controlador, se ha de prescindir de cualquier otra circuitería.

No obstante a que se sigan desarrollando y modernizando los Micro controladores existen una serie de dispositivos que ayudan de diversas formas el trabajo del Micro controlador y le dan ventajas y facilidad de eliminarle carga la mismo como son: los puertos externos 8255, controlador de teclado display, memorias externas y otros dispositivos como demultiplexores, Conversores BCD siete segmentos, buffer y otros mas que harán el trabajo en conjunto con el micro y será muy fácil la implementación de la aplicación ya que  a veces necesitamos de mas puertos o de mas capacidad.

## 2.2 ¿POR QUE UNA PEQUEÑA COMPUTADORA?

A es este chip se le conoce también como microcomputadora, porque tiene todos los componentes y recursos necesarios para serlo, es decir:

- Una CPU (Central Processor Unit o Unidad de Procesamiento Central) quien interpreta las instrucciones de programa.

- Una memoria PROM (Programmable Read Only Memory o Memoria Programable Solamente para Lectura) el cual memoriza permanentemente las instrucciones de programa. Otros modelos de Micro controladores tienen memoria de programa de tipo EEPROM y otros de tipo FLASH

- Una memoria RAM (Random Access Memory o Memoria de Acceso Aleatorio) utilizada para memorizar las variables utilizadas para el programa.

- Una serie de LINEAS de E/S para controlar dispositivos externos o recibir pulsos de sensores, switches, etc.

- Una serie de dispositivos auxiliares para su funcionamiento, como puede ser generador de clock, bus, contador, etc.

## 2.3 VARIACIONES DEL PIC.

### a) PICs modernos

Los viejos PICs con memoria PROM o EPROM se están renovando gradualmente por chips con memoria Flash. Así mismo, el juego de instrucciones original de 12 bits del PIC1650 y sus descendientes directos ha sido suplantado por juegos de instrucciones de 14 y 16 bits. Microchip todavía vende versiones PROM y EPROM de la mayoría de los PICs para soporte de aplicaciones antiguas o grandes pedidos.

### b) Clones del PIC

Por todos lados surgen compañías que ofrecen versiones del PIC más baratas o mejoradas. La mayoría suelen desaparecer rápidamente. Una de ellas que va perdurando es Ubicorn (antiguamente Scenix) que vende clones del PIC que funcionan mucho más rápido que el original. OpenCores tiene un núcleo del PIC16F84 escrito en Verilog.

### c) PICs wireless

El Micro controlador **rfPIC** integra todas las prestaciones del PICmicro de Microchip con la capacidad de comunicación wireless UHF para aplicaciones RF de baja potencia. Estos dispositivos ofrecen un diseño muy comprimido para ajustarse a los cada vez más demandando requerimientos de miniaturización en aparatos electrónicos.

### d) PICs para procesado de señal (dsPICs)

Los dsPICs (Digital Signal Controller). Son el último lanzamiento de Microchip, comenzando a producirlos a gran escala a finales de 2004. Son los primeros PICs con bus de datos inherente de 16 bits. Incorporan todas las posibilidades de los anteriores PICs y añaden varias operaciones de DSP (Digital Signal Processor) Implementadas en hardware, como multiplicación con suma de acumulador.

## 2.4 LOS MICRO CONTROLADORES MÁS COMUNES EN USO SON:

<table>
<tr><th>Fabricante</th><th colspan="3">Tipo</th></tr>
<tr><td>Atmel</td><td colspan="3">AVR</td></tr>
<tr><td rowspan="2">Free scale (antes Motorola)</td><td>8 bits</td><td>16 bits</td><td>32 bits</td></tr>
<tr><td>68HC05<br>68HC08<br>68HC11</td><td>68HC12<br>68HC16</td><td>683xx</td></tr>
<tr><td>Hitachi</td><td colspan="3">H8</td></tr>
<tr><td>Holtek</td><td colspan="3">HT8</td></tr>
<tr><td>Intel</td><td colspan="2">8 bits</td><td>16 bits</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">8XC42<br>MCS51<br>8xC251</td><td>MCS96<br>MXS296</td></tr>
<tr><td>National Semiconductor</td><td colspan="3">COP8</td></tr>
<tr><td rowspan="2">Microchip</td><td colspan="3">8 bits</td><td>16 bits</td></tr>
<tr><td>10f2xx<br>12Cxx<br>12Fxx</td><td>16Cxx<br>16Fxx</td><td>18Cxx<br>18Fxx</td><td>24F<br>24H<br>dsPIC30<br>dsPIC33</td></tr>
<tr><td>NEC</td><td colspan="3">78K</td></tr>
<tr><td>Parallax</td><td colspan="3">BASIC Stamp<br>SX<br>Propeller</td></tr>
</table>

| | |
|---|---|
| ST | ST 62<br><br>ST 7 |
| Texas Instruments | TMS370 |
| Zilog | Z8<br><br>Z86E02 |
| Genérico | Algunas arquitecturas de Micro controlador están disponibles por tal cantidad de vendedores y en tantas variedades, que podrían tener, con total corrección, su propia categoría. Entre ellos encontramos, principalmente, las variantes de 8051 y Z80. |

***PICS MÁS COMÚNMENTE USADOS.***

- PIC12C508/509 (encapsulamiento reducido de 8 pines, oscilador interno, popular en pequeños diseños como el hipad remote)

- PIC16F84 (Considerado obsoleto, pero imposible de descartar y muy popular)

- PIC16F84A (Buena actualización del anterior, algunas versiones funcionan a 20 MHz)

- PIC12F629/675

- PIC16F628

- La familia PIC16F87X (los hermanos mayores del PIC16F84, con cantidad de mejoras incluidas en hardware. Bastante común en proyectos de aficionados)

- PIC18F452

## 2.5 JUEGO DE INSTRUCCIONES Y ENTORNO DE PROGRAMACIÓN

El PIC usa un juego de instrucciones tipo RISC, cuyo número puede variar desde 35 para PICs de gama baja a 70 para los de gama alta. Las instrucciones se clasifican *entre las que realizan operaciones entre el acumulador y una constante,* entre el acumulador y una posición de memoria, instrucciones de condicionamiento y de salto/retorno, implementación de interrupciones y una para pasar a modo de bajo consumo llamada sleep.

Microchip proporciona un entorno de desarrollo freeware llamado *MPLAB* que incluye un simulador software y un ensamblador. Otras empresas desarrollan compiladores C y BASIC. Microchip también vende compiladores para los PICs de gama alta ("C18" para la serie F18 y "C30" para los dsPICs) y se puede descargar una edición para estudiantes del C18 que inhabilita algunas opciones después de un tiempo de evaluación.

Para Pascal existe un compilador de código abierto, JAL, lo mismo que PicForth para el lenguaje Forth. GPUTILS es una colección de herramientas distribuidas bajo licencia GNU que incluye ensamblador y enlazador, y funciona en Linux, MacOS y Microsoft Windows. GPSIM es otra herramienta libre que permite simular diversos dispositivos hardware conectados al PIC.

Los viejos PICs con memoria PROM o EPROM se están renovando gradualmente por chips con memoria Flash. Así mismo, el juego de instrucciones original de 12 bits del PIC1650 y sus descendientes directos ha sido suplantado por juegos de instrucciones de 14 y 16 bits. Microchip todavía vende versiones PROM y EPROM de la mayoría de los PICs para soporte de aplicaciones antiguas o grandes pedidos.

## 2.6 ELEMENTOS QUE CONFIGURAN UN CONTROLADOR.

Un Micro controlador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador.

Un Micro controlador dispone normalmente de los siguientes componentes:

- Núcleo de arquitectura tipo Harvard de 8/16 bits.
- Memoria Flash y ROM desde 256 bytes a 256 Kb
- Puertos de entrada / salida (I/O)
- Temporizadores de 8/16 bits.
- Función sleep (dormir) de bajo consumo.
- Comunicación serial ( USART , AUSART , EUSART )
- Conversores ADC de 10/12 bits.
- Comparadores de tensión.
- Módulos PWM
- Comunicaciones vía I2C y SPI
- Memoria EEPROM interna.

- Soporte bus USB.
- Soporte red Ethernet.
- Soporte bus CAN.
- Soporte LIN.
- Soporte de IrDA.
- LIN controller support
- Capacidad de DSP (digital signal processing - procesamiento digital de señales) en dsPICs
- Procesador o UCP (Unidad Central de Proceso).
- Memoria RAM para Contener los datos.
- Memoria para el programa tipo ROM/PROM/EPROM.
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, Puertas Serie y Paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema.

## 2.7 VENTAJAS DE LOS PRODUCTOS QUE INCORPORAN UN MICRO CONTROLADOR.

- **Aumento de prestaciones**: un mayor control sobre un determinado elemento representa una mejora considerable en el mismo.

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

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

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

El Micro controlador es en definitiva un circuito integrado que incluye todos los componentes de un computador. Debido a su reducido tamaño es posible montar el controlador en el propio dispositivo al que gobierna. En este caso el controlador recibe el nombre de controlador empotrado (embedded controller).

## 2.8 DIFERENCIA ENTRE MICROPROCESADOR Y MICRO CONTROLADOR.

Es muy habitual confundir los términos de Micro controlador y microprocesador, cayendo así en un error de cierta magnitud. Un Micro controlador es un sistema completo, con unas prestaciones limitadas que no pueden modificarse y que puede llevar a cabo las tareas para las que ha sido programado de forma autónoma. Un microprocesador, en cambio, es simplemente un componente que conforma el Micro controlador, que lleva a cabo ciertas tareas que analizaremos más adelante y que, en conjunto con otros componentes, forman un Micro controlador.

Debe quedar clara por tanto la diferencia entre Micro controlador y microprocesador: a modo de resumen, el primero es un sistema autónomo e independiente, mientras que el segundo es una parte, cabe decir que esencial, que forma parte de un sistema mayor.

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

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

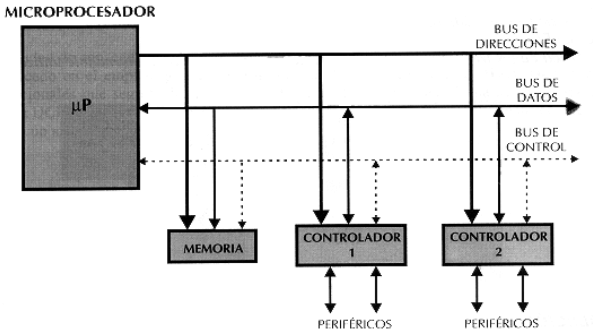



**Figura 1. Estructura de un sistema abierto basado en un microprocesador. La disponibilidad de los buses en el exterior permite que se configure a la medida de la aplicación.**

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

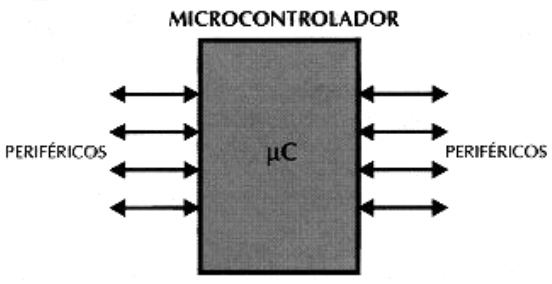



**Figura 2. El Micro controlador es un sistema cerrado. Todas las partes del computador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.**

## 2.9 APLICACIONES DE LOS MICRO CONTROLADORES.

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

Algunos fabricantes de Micro controladores superan el millón de unidades de un modelo determinado producidas en una semana. Este dato puede dar una idea de la masiva utilización de estos componentes.

Los Micro controladores están siendo empleados en multitud de sistemas presentes en nuestra vida diaria, como pueden ser juguetes, horno microondas, frigoríficos, televisores, computadoras, impresoras, módems, el sistema de arranque de nuestro coche, etc. Y otras aplicaciones con las que seguramente no estaremos tan familiarizados como instrumentación electrónica, control de sistemas en una nave espacial, etc. Una aplicación típica podría emplear varios Micro controladores para controlar pequeñas partes del sistema. Estos pequeños controladores podrían comunicarse entre ellos y con un procesador central, probablemente más potente, para compartir la información y coordinar sus acciones, como, de hecho, ocurre ya habitualmente en cualquier PC.

## 2.10 EL MERCADO DE LOS MICRO CONTROLADORES.

Aunque en el mercado de la microinformática la mayor atención la acaparan los desarrollos de los microprocesadores, lo cierto es que se venden cientos de Micro controladores por cada uno de aquéllos.

Existe una gran diversidad de micros controladores. Quizá la clasificación más importante sea entre Micro controladores de 4, 8, 16 ó 32 bits. Aunque las prestaciones de los Micro controladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los Micro controladores de 8 bits dominan el mercado y los de 4 bits se resisten a desaparecer. La razón de esta tendencia es que los Micro controladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace absurdo emplear micros más potentes y consecuentemente más caros. Uno de los sectores que más tira del mercado del Micro controlador es el mercado automovilístico. De hecho, algunas de las familias de Micro controladores actuales se desarrollaron pensando en este sector, siendo modificadas posteriormente para adaptarse a sistemas más genéricos. El mercado del automóvil es además uno de los más exigentes: los componentes electrónicos deben operar bajo condiciones extremas de vibraciones, choques, ruido, etc. y seguir siendo fiables. El fallo de cualquier componente en un automóvil puede ser el origen de un accidente.

En cuanto a las técnicas de fabricación, cabe decir que prácticamente la totalidad de los Micro controladores actuales se fabrican con tecnología CMOS 4 (Complementary Metal Oxide Semiconductor). Esta tecnología supera a las técnicas anteriores por su bajo consumo y alta inmunidad al ruido.

## 2.11 LA DISTRIBUCIÓN DE LAS VENTAS SEGÚN SU APLICACIÓN:

- Una tercera parte se absorbe en las aplicaciones relacionadas con los computadores y sus periféricos.

- La cuarta parte se utiliza en las aplicaciones de consumo (electrodomésticos, juegos, TV, vídeo, etc.)

- El 16% de las ventas mundiales se destinó al área de las comunicaciones.

- Otro 16% fue empleado en aplicaciones industriales.

- El resto de los Micro controladores vendidos en el mundo, aproximadamente un 10% fueron adquiridos por las industrias de automoción.

También los modernos Micro controladores de 32 bits van afianzando sus posiciones en el mercado, siendo las áreas de más interés el procesamiento de imágenes, las comunicaciones, las aplicaciones militares, los procesos industriales y el control de los dispositivos de almacenamiento masivo de datos.

## 2.12 ¿QUÉ MICRO CONTROLADOR EMPLEAR?

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

a) **Costes.** Como es lógico, los fabricantes de Micro controladores compiten duramente para vender sus productos. Y no les va demasiado mal ya que sin hacer demasiado ruido venden 10 veces más Micro controladores que microprocesadores. Para que nos hagamos una idea, para el fabricante que usa el Micro controlador en su producto una diferencia de precio en el Micro controlador de algunos centavos es importante (el consumidor deberá pagar además el coste del empaquetado, el de los otros componentes, el diseño del hardware y el desarrollo del software). Si el fabricante desea reducir costes debe tener en cuenta las herramientas de apoyo con que va a contar: emuladores, simuladores, ensambladores, compiladores, etc. Es habitual que muchos de ellos siempre se decanten por Micro controladores pertenecientes a una única familia.

b) **Aplicación.** Antes de seleccionar un Micro controlador es imprescindible analizar los requisitos de la aplicación.

c) **Procesamiento de datos:** puede ser necesario que el Micro controlador realice cálculos críticos en un tiempo limitado. En ese caso debemos asegurarnos de seleccionar un dispositivo suficientemente rápido para ello. Por otro lado, habrá que tener en

cuenta la precisión de los datos a manejar: si no es suficiente con un Micro controlador de 8 bits, puede ser necesario acudir a Micro controlador de 16 ó 32 bits, o incluso a hardware de coma flotante. Una alternativa más barata y quizá suficiente es usar librerías para manejar los datos de alta precisión. -

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

e) **Consumo:** algunos productos que incorporan micros controladores están alimentados con baterías y su funcionamiento puede ser tan vital como activar una alarma antirrobo. Lo más conveniente en un caso como éste puede ser que el Micro controlador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla.

f) **Memoria:** para detectar las necesidades de memoria de nuestra aplicación debemos separarla en memoria volátil (RAM), memoria no volátil (ROM, EPROM, etc.) y memoria no volátil modificable (EEPROM). Este último tipo de memoria puede ser útil para incluir información específica de la aplicación como un número de serie o parámetros de calibración. El tipo de memoria a emplear vendrá determinado por el volumen de ventas previsto del producto: de menor a mayor volumen será conveniente emplear EPROM, OTP y ROM. En cuanto a la cantidad de memoria necesaria puede ser

imprescindible realizar una versión preliminar, aunque sea en seudo-código, de la aplicación y a partir de ella hacer una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

g) **Ancho de palabra:** el criterio de diseño debe ser seleccionar el Micro controlador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Usar un Micro controlador de 4 bits supondrá una reducción en los costes importante, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un byte. Los Micro controladores de 16 y 32 bits, debido a su elevado coste, deben reservarse para aplicaciones que requieran sus altas prestaciones (Entrada/Salida potente o espacio de direccionamiento muy elevado).

h) **Diseño de la placa:** la selección de un Micro controlador concreto condicionará el diseño de la placa de circuitos. Debe tenerse en cuenta que quizá usar un Micro controlador barato encarezca el resto de componentes del diseño.

Los micros controladores más populares se encuentran, sin duda, entre las mejores elecciones:

- 8048 (Intel). Es el padre de los micros controladores actuales, el primero de todos. Su precio, disponibilidad y herramientas de desarrollo hacen que todavía sea muy popular.

- 8051 (Intel y otros). Es sin duda el Micro controlador más popular. Fácil de programar, pero potente. Está bien

documentado y posee cientos de variantes e incontables herramientas de desarrollo.

- 80186, 80188 y 80386 EX (Intel). Versiones en Micro controlador de los populares microprocesadores 8086 y 8088. Su principal ventaja es que permiten aprovechar las herramientas de desarrollo para PC.

- 68HC11 (Motorola y Toshiba). Es un Micro controlador de 8 bits potente y popular con gran cantidad de variantes.

- 683xx (Motorola). Surgido a partir de la popular familia 68k, a la que se incorporan algunos periféricos. Son Micro controladores de altísimas prestaciones.

- PIC (Microchip). Familia de micros controladores que gana popularidad día a día. Fueron los primeros micros controladores RISC.

Es preciso resaltar en este punto que existen innumerables familias de Micro controladores, cada una de las cuales posee un gran número de variantes.

## 2.13 RECURSOS COMUNES A TODOS LOS MICRO CONTROLADORES.

Al estar todos los micros controladores integrados en un chip, su estructura fundamental y sus características básicas son muy parecidas. Todos deben disponer de los bloques esenciales Procesador, memoria de datos y de instrucciones, líneas de E/S, oscilador de reloj y módulos controladores de periféricos. Sin embargo, cada fabricante intenta enfatizar los recursos más idóneos para las aplicaciones a las que se destinan preferentemente.

En este apartado se hace un recorrido de todos los recursos que se hallan en todos los Micro controladores describiendo las diversas alternativas y opciones que pueden encontrarse según el modelo seleccionado.

### 2.13.1 ARQUITECTURA BÁSICA.

Aunque inicialmente todos los Micro controladores adoptaron la arquitectura clásica de von Neumann, en el momento presente se impone la arquitectura Harvard. La arquitectura de von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control).

## 2.13.2 ARQUITECTURA INTERNA DEL PIC:

Hay dos arquitecturas conocidas; la clásica de ***von Neumann***, y la ***arquitectura Harvard***, veamos como son:

***Arquitectura Von Neumann*** Dispone de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control).


**Arquitectura Von Neumann**

***Arquitectura Harvard*** Dispone de dos memorias independientes, una que contiene sólo instrucciones, y otra que contiene sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias, ésta es la estructura para los PICs.

Otra aportación frecuente que aumenta el rendimiento del computador es el fomento del paralelismo implícito, que consiste en la segmentación del procesador (pipe-line), descomponiéndolo en etapas para poder procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez.


**Los Micro controladores PIC responden a la arquitectura Harvard.**

### 2.13.2.1 EL PROCESADOR O UCP.

Es el elemento más importante del Micro controlador y determina sus principales características, tanto a nivel hardware como software.

Se encarga de direccionar la memoria de instrucciones, recibir el código OP de la instrucción en curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado.

Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales.

a) **CISC (Computadores de Juego de Instrucciones Complejo):** Un gran número de procesadores usados en los Micro controladores están basados en la filosofía Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución. Una ventaja de los procesadores **CISC** es que

ofrecen al programador instrucciones complejas que actúan como macros.

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

c) **SISC (Computadores de Juego de Instrucciones Específico):** En el Micro controlador destinado a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es "específico", o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista.

### 2.13.2.2 MEMORIA.

En los Micro controladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo **ROM**, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo **RAM**, volátil, y se destina a guardar las variables y los datos.

Hay dos peculiaridades que diferencian a los Micro controladores de los computadores personales:

- No existen sistemas de almacenamiento masivo como disco duro o disquetes.

- Como el Micro controlador sólo se destina a una tarea en la memoria **ROM**, sólo hay que almacenar un único programa de trabajo.

La **RAM** en estos dispositivos es de poca capacidad pues sólo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la **RAM** pues se ejecuta directamente desde la **ROM**.

Los usuarios de computadores personales están habituados a manejar Megabytes de memoria, pero, los diseñadores con Micro controladores trabajan

con capacidades de **ROM** comprendidas entre 512 bytes y 8 k bytes y de **RAM** comprendidas entre 20 y 512 bytes.

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

**ROM** con máscara.

Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip. El elevado coste del diseño de la máscara sólo hace aconsejable el empleo de los Micro controladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

**OTP.**

El Micro controlador contiene una memoria no volátil de sólo lectura "programable una sola vez" por el usuario. OTP (One Time Programmable). Es el usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde un PC.

La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas.

Tanto en este tipo de memoria como en la EPROM, se suele usar la encriptación mediante fusibles para proteger el código contenido.

**EPROM.**

Los Micro controladores que disponen de memoria **EPROM** (Erasable Programmable Read OnIy Memory) pueden borrarse y grabarse muchas veces.

La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde un PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la **EPROM** a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los Micro controladores con memoria OTP que están hechos con material plástico.

**EEPROM.**

Se trata de memorias de sólo lectura, programables y borrables eléctricamente **EEPROM** (Electrical Erasable Programmable Read Only Memory). Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. Es muy cómoda y rápida la operación de grabado y la de borrado. No disponen de ventana de cristal en la superficie.

Los Micro controladores dotados de memoria **EEPROM** una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

El número de veces que puede grabarse y borrarse una memoria **EEPROM** es finito, por lo que no es recomendable una reprogramación continúa. Son muy idóneos para la enseñanza y la Ingeniería de diseño.

Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria **EEPROM** en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que adecuan el dispositivo a las condiciones del entorno. Este tipo de memoria es relativamente lenta.

**FLASH.**

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una **ROM** y una **RAM** pero consume menos y es más pequeña.

A diferencia de la **ROM**, la memoria **FLASH** es programable en el circuito. Es más rápida y de mayor densidad que la **EEPROM**.

La alternativa **FLASH** está recomendada frente a la **EEPROM** cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado.

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

### 2.13.2.3 PUERTAS DE ENTRADA Y SALIDA.

La principal utilidad de las patitas que posee la cápsula que contiene un Micro controlador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores.

Por ellas enviamos o introducimos señales digitales TTL (5V) de forma que podemos comunicar el Micro controlador con el exterior según los controladores de periféricos que posea cada modelo de Micro controlador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control

### 2.13.2.4 RELOJ PRINCIPAL.

Todos los Micro controladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

Generalmente, el circuito de reloj está incorporado en el Micro controlador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C.

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía.

## 2.14 RECURSOS ESPECIALES.

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

Los principales recursos específicos que incorporan los Micro controladores son:

- Temporizadores o "Timers".
- Perro guardián o "Watchdog".
- Protección ante fallo de alimentación o "Brownout".
- Estado de reposo o de bajo consumo.
- Conversor A/D.
- Conversor D/A.
- Comparador analógico.
- Modulador de anchura de impulsos o PWM.
- Puertas de E/S digitales.
- Puertas de comunicación.

### 2.14.1 TEMPORIZADORES O "TIMERS".

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).

Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patitas del Micro controlador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

### 2.14.2 PERRO GUARDIÁN O "WATCHDOG".

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema. Pero un Micro controlador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, "ladrará y ladrará" hasta provocar el reset.

### 2.14.3 PROTECCIÓN ANTE FALLO DE ALIMENTACIÓN O "BROWNOUT"

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

### 2.14.4 ESTADO DE REPOSO Ó DE BAJO CONSUMO.

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

### 2.14.5 CONVERSOR A/D (CAD).

Los Micro controladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patitas del circuito integrado.

### 2.14.6 CONVERSOR D/A (CDA).

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

### 2.14.7 COMPARADOR ANALÓGICO.

Algunos modelos de Micro controladores disponen internamente de un Amplificador Operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por una de las patitas de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.

También hay modelos de Micro controladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia que se pueden aplicar en los comparadores.

### 2.14.8 MODULADOR DE ANCHURA DE IMPULSOS O PWM.

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

### 2.14.9 PUERTOS DE E/S DIGITALES.

Todos los Micro controladores destinan algunas de sus patitas a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos.

Las líneas digitales de los Puertos pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

## 2.14.10 PUERTOS DE COMUNICACIÓN.

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

- **UART**, adaptador de comunicación serie asíncrona.
- **USART,** adaptador de comunicación serie síncrona y asíncrona
- Puerta paralela esclava para poder conectarse con los buses de otros microprocesadores.
- **USB (Universal Serial Bus)**, que es un moderno bus serie para los PC.
- **Bus $I^2C$,** que es un interfaz serie de dos hilos desarrollado por Philips.
- **CAN (Controller Área Network)**, para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE.UU. se usa el J185O.

## 2.15 HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES.

Uno de los factores que más importancia tiene a la hora de seleccionar un Micro controlador entre todos los demás es el soporte tanto software como hardware de que dispone. Un buen conjunto de herramientas de desarrollo puede ser decisivo en la elección, ya que pueden suponer una ayuda inestimable en el desarrollo del proyecto.

Las principales herramientas de ayuda al desarrollo de sistemas basados en Micro controladores son:

### 2.15.1 DESARROLLO DEL SOFTWARE:

#### 2.15.1.1 ENSAMBLADOR.

La programación en lenguaje ensamblador puede resultar un tanto ardua para el principiante, pero permite desarrollar programas muy eficientes, ya que otorga al programador el dominio absoluto del sistema. Los fabricantes suelen proporcionar el programa ensamblador de forma gratuita y en cualquier caso siempre se puede encontrar una versión gratuita para los Micro controladores más populares.

### 2.15.1.2 COMPILADOR.

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

### 2.15.1.3 DEPURACIÓN.

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

### 2.15.1.4 SIMULADOR.

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

### 2.15.1.5 PLACAS DE EVALUACIÓN.

Se trata de pequeños sistemas con un Micro controlador ya montado y que suelen conectarse a un PC desde el que se cargan los programas que se ejecutan en el Micro controlador. Las placas suelen incluir visualizadores LCD, teclados, Leds, fácil acceso a los pines de E/S, etc. El sistema operativo de la placa recibe el nombre de programa monitor.

El programa monitor de algunas placas de evaluación, aparte de permitir cargar programas y datos en la memoria del Micro controlador, puede permitir en cualquier momento realizar ejecución paso a paso, monitorizar el estado del Micro controlador o modificar los valores almacenados los registros o en la memoria.

### 2.15.1.6 EMULADORES EN CIRCUITO.

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

## 2.16 DISEÑO DE PROYECTOS.

Frente a un problema técnico, hay que buscar una solución de forma barata y sencilla, en este proceso de búsqueda de soluciones, los Micro controladores PIC pueden ayudarnos a realizar soluciones sencillas, rápidas y baratas.

Partiremos de un planteamiento teniendo presente todas las variables que afectan al sistema, desarrollaremos la idea y la implementaremos con las herramientas adecuadas.

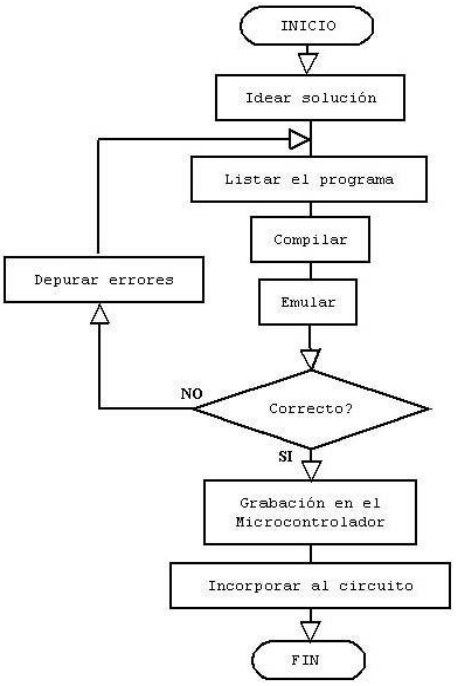



**Diagrama de flujo del desarrollo de proyectos con Micro controlador**

## 2.17 FAMILIA DEL PIC 16F87X

### 2.17.1 PRINCIPALES CARACTERÍSTICAS.

- Procesador de arquitectura RISC avanzada.
- Juego de solo 35 instrucciones con 14 bits de longitud. Todas ellas se ejecutan en un ciclo de instrucción, menos las de salto que tardan 2.
- Hasta 8K palabras de 14 bits para la memoria de programa, tipo FLASH en los modelos 16F876 y 16F877 y 4KB de memoria para los PIC 16F873 y 16F874.
- Hasta 368 Bytes de memoria de datos RAM.
- Hasta 256 Bytes de memoria de datos EEPROM.
- Pines de salida compatibles para el PIC 16C73/74/76/77.
- Hasta 14 fuentes de interrupción internas y externas.
- Pila de 8 niveles.
- Modos de direccionamiento directo e indirecto.
- Power-on reset (pop).
- Temporizador Power-on (POP) y Oscilador Temporizador Start-Up.

- Perro Guardián (WDT).
- Código de protección programable.
- Modo SLEEP de bajo consumo.
- Programación serie en circuito con dos pines, solo necesita 5V para programarlo en ese modo.
- Voltaje de alimentación comprendido entre 2 y 5,5V.
- Bajo consumo: <2 mA valor para 5V y de 4 Mhz 20 µA para 3V y 32 M> 1 µA en standby.

### 2.17.2 DISPOSITIVOS PERIFÉRICOS.

- Timer0: Temporizador-contador de 8 bits con Preescaler de 8 bits.
- Timer1: Temporizador-contador de 16 bits con Preescaler que puede incrementarse en modo sleep de forma externa por un cristal/clock.
- Timer2: Temporizador-contador de 8 bits con Preescaler y postscaler.
- Dos módulos de captura, comparación PWM (Modulación de Ancho de Pulsos).

- Conversor A/D de 10 bits.

- Puerto serie síncrono Máster (MSSP) con SPI e $I^2C$ (Máster/Slave).

- USART/SCI (Universal Synchronous Asynchronous Receiver Transmitter) con 9 bits.

- Puerto Paralelo Esclavo (PSP) solo en encapsulados con 40 pines.

### 2.17.3 DIFERENCIAS ENTRE LOS MODELOS DE 28 Y LOS DE 40 PINES.

El PIC 16F873 y el 876 tienen 28 pines, mientras que el PIC 16F874 y 877 tienen 40. Las diferencias que existen entre estos son mínimas y se detallan a continuación:

- Los modelos de 40 pines disponen de 5 puertos de E/S: A, B, C, D y E, mientras que los de 28 pines solo tienen 3 puertos: A, B y C.

- Los modelos de 40 pines tienen 8 canales de entrada al Conversor A/D, mientras que los de 28 solo tienen 5 cabales.

- Solo poseen el Puerto Paralelo Esclavo los PIC 16F87X de 40 pines.

### 2.17.4 PIC 16F87X.

Bajo el nombre de esta subfamilia de micro controladores, actualmente encontramos cuatro modelos: el PIC 16F873/4/6 y 7. Estos controladores disponen de una memoria de programa FLASH de 4K a 8K palabras de 14 bits, considerablemente superior frente al PIC 16F84 en el que solo disponíamos de 1K palabras de 14 bits.

De los micro controladores indicados, el 16F873 y el 16F876 son de 28 pines, mientras que el 16F874 y el 16F877 tienen 40 pines, lo que les permite disponer de hasta 33 líneas de E/S. En su arquitectura además incorporan:

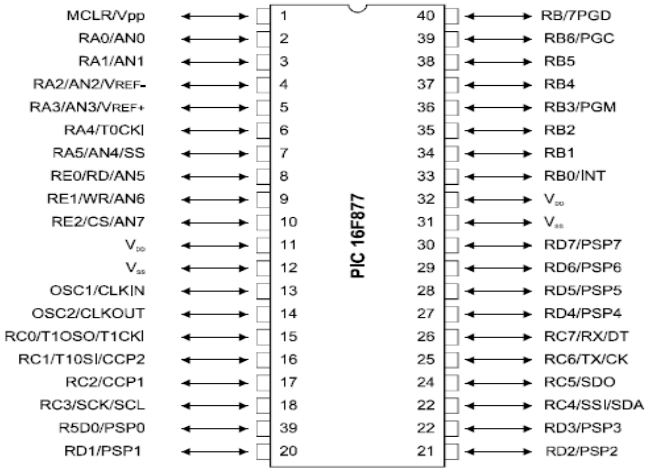
- Varios Timers.
- USART.
- Bus I2C.

## 2.17.5 COMPARACIÓN ENTRE LOS PIC DE LA FAMILIA 16F87X.

| Características | 16F873 | 16F874 | 16F876 | 16F877 |
|---|---|---|---|---|
| Frecuencia Máxima. | DC-20Mhz | DX-20Mhz | DX-20Mhz | DX-20Mhz |
| Memoria de programa FLASH (Palabra de 14 bits). | 4K | 4K | 8K | 8K |
| Posiciones RAM de datos. | 192 | 192 | 368 | 368 |
| Posiciones EEPROM de datos. | 128 | 128 | 256 | 256 |
| Puertos E/S. | A, B y C | A, B, C, D y E | A, B y C | A, B, C, D y E |
| N$^{o}$ de Pines. | 28 | 40 | 28 | 40 |
| Interrupciones | 13 | 14 | 13 | 14 |
| Timers | 3 | 3 | 3 | 3 |
| Módulos CCP | 2 | 2 | 2 | 2 |
| Comunicaciones serie | MSSP, USART | MSSP, USART | MSSP, USART | MSSP, USART |
| Comunicación paralelo | - | PSP | - | PSP |
| Líneas de entrada en convertidor A/D de 10 bits | 5 | 8 | 5 | 8 |
| Juego de instrucciones | 35 | 35 | 35 | 35 |
| Longitud de la instrucción | 14 bits | 14 bits | 14 bits | 14 bits |

## 2.17.6 PIC 16F877.

- Puertos programables de E/S.
- Timers/Counters.
- Puertos de captura/comparación de datos.
- Moduladores de ancho de pulso (PWM).
- Conversor Analógico/Digital de 10 bits.
- Puerto serie síncrono.
- USART.
- Parallel Slave Port (Puerto Paralelo Esclavo).



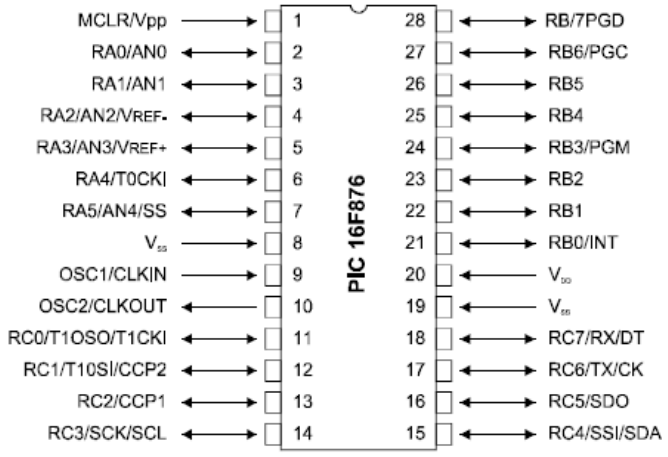

**MICRO CONTROLADOR PIC 16F877**

Es un circuito de 40 pines. Salvo 7 pines, todos los demás están asociados al menos con los puertos E/S. Los 7 pines son:

- 4 pines para alimentación, están duplicados.
- 1 pin para MCLR y Vpp que es una tensión de 12 a 14 V. Usado cuando esta en modo programación.
- pines relacionados con entradas o salidas de reloj, en función del tipo de reloj utilizado.

## 2.17.7 PIC 16F876.

El 16F876 contiene el mismo núcleo que el 16F877, pero dispone de 28 pines, lo cual supone eliminar algunos de los periféricos de los presentes en su hermano mayor.

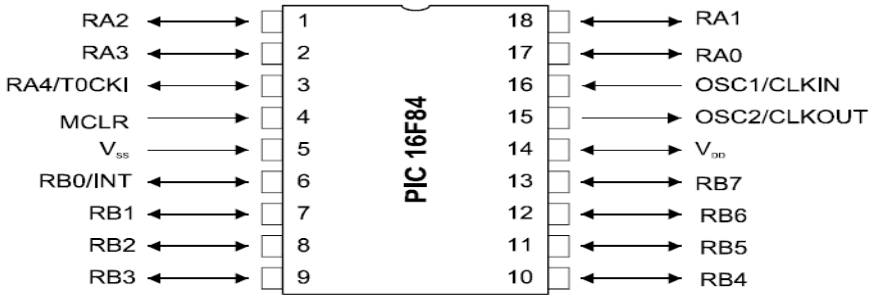



**MICRO CONTROLADOR PIC 16F876**

Tal y como se muestra en la figura, se puede observar que carece de los puertos D (8 bits) y E (3 bits), y en cuanto a periféricos, no existe el SPP (Slave Parallel Port) y el Conversor A/D dispone de 5 entradas (frente a 8 del 16F877). También se dispone de una entrada menos de alimentación (un único pin para Vdd).

## 2.17.8 PIC 16F84

El 16F84 es un micro controlador de la gama media al igual que el 16F877, pero que carece de varios de los periféricos de éste a cambio de ser un circuito más económico y más pequeño. Carece de Conversor A/D, y dispone de solo 11 líneas de entrada/salida.




**MICRO CONTROLADOR PIC 16F84**

## 2.18 PROGRAMACIÓN DE MICRO CONTROLADORES.

Una parte importante al momento de tratar con Micro controladores es su programación, es aquí donde debemos conocer software y hardware esencial para esta tarea. Al hablar de hardware nos referimos a los dispositivos necesarios para grabar la información dentro del Micro controlador, y cuando hablamos de software es acerca de las aplicaciones que nos permiten transformar la información de los programas en un lenguaje legible para el Micro controlador, es por esto que es necesario conocer las diferentes alternativas existentes, y así poder elegir en base a nuestra comodidad y conveniencia. En este apartado conoceremos primero lo concerniente al software para luego mostrar las opciones en hardware.

### 2.18.1 SISTEMAS DE NUMERACIÓN.

La memoria en una computadora está compuesta de números. La memoria de la computadora no almacena estos números en decimal (base 10). Porque se simplifica mucho el hardware, las computadoras almacenan toda la información en binario (base 2). Primero haremos una revisión del sistema decimal.

#### 2.18.1.1 DECIMAL.

Los números con base 10 están compuestos de 10 posibles dígitos (0-9). Cada digito de un número tiene una potencia de 10 asociada con él, basada en su posición en el número. Por ejemplo:

$$234 = 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

### 2.18.1.2 BINARIO.

Los números en base dos están compuestos de dos posibles dígitos (0 y 1). Cada digito de un número tiene una potencia de 2 asociada con él basada en su posición en el número. Por ejemplo:

$$110012 = 1 \times 24 + 1 \times 23 + 0 \times 22 + 0 \times 21 + 1 \times 20$$

$$= 16 + 8 + 1$$

$$= 25$$

Esto muestra como los números binarios se pueden convertir a decimal.

El Cuadro muestra como se representan los primeros números en binario.

| Decimal | Binario | Decimal | Binario |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | 10 | 1010 |
| 3 | 0011 | 11 | 1011 |

| | | | |
|---|---|---|---|
| 4 | 0100 | 12 | 1100 |
| 5 | 0101 | 13 | 1101 |
| 6 | 0110 | 14 | 1110 |
| 7 | 0111 | 15 | 1111 |

**Suma binaria (c es acarreo)**

| No hay acarreo antes | | | | Si hay acarreo antes | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| +0 | +1 | +0 | +1 | +0 | +1 | +0 | +1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| | | | c | | c | c | c |

A continuación se muestra como se suman los dígitos binarios individuales (bits). Ejemplo:

$$\begin{array}{r} 11011_2 \\ +10001_2 \\ \hline 101100_2 \end{array}$$

Si uno considera la siguiente división decimal:

$$1234 \div 10 = 123 \text{ r } 4$$

Podemos ver que esta división suprime el dígito más a la derecha del número y desplaza los otros dígitos una posición a la derecha. Dividiendo por dos hacemos una operación similar, pero para los dígitos binarios de un número. Consideremos la siguiente división binaria[1]:

$$1101_2 \div 10_2 = 110_2 \text{ r } 1$$

Este hecho se puede usar para convertir un numero decimal a su representación equivalente en binario como muestra la Figura. Este método encuentra primero el bit del extremo derecho, llamado bit menos significativo (lbs.). El bit del extremo izquierdo es llamado bit más significativo (msb).

La unidad básica de memoria esta compuesta de 8 bits y es llamado byte.

[1] El subíndice 2 se usa para mostrar que el numero esta representado en binario no en decimal.

**Conversión a decimal**

| Decimal | Binario |
|---|---|
| 25 ÷ 2 = 12 r 1 | 11001 ÷ 10 = 1100 r 1 |
| 12 ÷ 2 = 6 r 0 | 1100 ÷ 10 = 110 r 0 |
| 6 ÷ 2 = 3 r 0 | 110 ÷ 10 = 11 r 0 |
| 3 ÷ 2 = 1 r 1 | 11 ÷ 10 = 1 r 1 |
| 1 ÷ 2 = 0 r 1 | 1 ÷ 10 = 0 r 1 |
| Así $25_{10} = 11001_2$ | |

### 2.18.1.3 HEXADECIMAL.

Los números hexadecimales tienen base 16. Los hexadecimales (o hex) se pueden usar como una representación resumida de los números binarios. Los números hexadecimales tienen 16 dígitos posibles. Esto crea un problema ya que no hay símbolos para estos dígitos adicionales después del nueve. Por convención se usan letras para estos dígitos adicionales. Los 16 dígitos hexadecimales son: 0-9 y luego A, B, C, D, E, F. El digito A equivale a 10 en decimal, B es 11 etc. Cada digito de un numero hexadecimal tiene una potencia de 16 asociada con el. Por ejemplo:

$$2BD_{16} = 2 \times 16^2 + 11 \times 16^1 + 13 \times 16^0$$

$$= 512 + 176 + 13$$

$$= 701$$

Para convertir de decimal a hex use la misma idea que la usada para la conversión binaria excepto que se divide por 16. Vea la Figura  para un ejemplo.

La razón por la cual los hexadecimales son útiles es que hay una manera fácil para convertir entre HEX y binario. Los números binarios se tornan largos y molestos rápidamente. Los HEX son una manera mucho mas compacta de representar los números binarios.

$$589 \div 16 = 36 \text{ r } 13$$

$$36 \div 16 = 2 \text{ r } 4$$

$2 \div 16 = 0 \text{ r } 2$

Así $589 = 24D_{16}$

Para convertir un número hexadecimal a binario simplemente convierta cada digito hexadecimal a un número binario de 4 bits. Por ejemplo, $24D_{16}$ se convierta a $0010\ 0100\ 1101_2$. Observe que ¡los ceros delanteros son importantes! Si los ceros del dígito de la mitad de $24D_{16}$ no se usan el resultado es erróneo. Convertir de binario a HEX es así de fácil. Uno hace el proceso inverso, convierte cada segmento de 4 bits a hexadecimal comenzando desde el extremo derecho, no desde el izquierdo, del número binario. Esto asegura que el segmento de 4 bits es correcto[2]. Ejemplo:

$0110\ 0000\ 0101\ 1010\ 0111\ 1110_2$

$6 \quad 0 \quad 5 \quad A \quad 7 \quad E_{16}$

Un número de 4 bits es llamado nibble. Así cada dígito hexadecimal corresponde a un nibble. Dos nibble conforman un byte y por lo tanto un byte puede ser representado por dos dígitos hexadecimales. Los valores de un byte van de 0 a 11111111 en binario, 0 a FF en HEX y 0 a 255 en decimal.

[2] Si no es claro porque el punto de inicio hace la diferencia, intente convertir el ejemplo comenzando desde la izquierda.

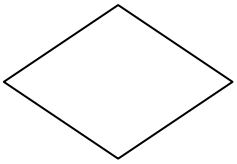
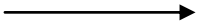
## 2.18.2 DIAGRAMAS DE FLUJO.

Un diagrama de flujo es una representación gráfica de un algoritmo o de una parte del mismo. Los diagramas de flujo ayudan en la comprensión de la operación de las estructuras de control (*Si, Mientras*).

La ventaja de utilizar un algoritmo es que lo puede construir independientemente de un lenguaje de programación, pues al momento de llevarlo a código se lo puede hacer en cualquier lenguaje.

Dichos diagramas se construyen utilizando ciertos símbolos de uso especial como son rectángulos, diamantes, óvalos, y pequeños círculos, estos símbolos están conectados entre sí por flechas, conocidas como *líneas de flujo*. A continuación se detallarán estos símbolos.

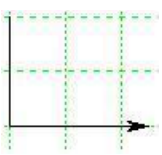
| **Nombre** | **Símbolo** | **Función** |
|---|---|---|
| Terminal |  | Representa el inicio y fin de un programa. También puede representar una parada o interrupción |
| Entrada / salida |  | Cualquier tipo de introducción de datos en la memoria desde los periféricos o registro de información procesada en un periférico. |

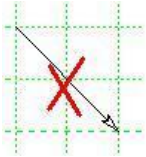
| | | |
|---|---|---|
| Proceso |  | Cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transformaciones. |
| Decisión |  | Indica operaciones lógicas o de comparación entre datos (normalmente dos) y en función del resultado de la misma determina (normalmente si y no) cual de los distintos caminos alternativos del programa se debe seguir. |
| Conector |  | Sirve para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en la misma pagina del diagrama |
| Indicador de dirección o línea de flujo |  | Indica el sentido de la ejecución de las operaciones |

| | | |
|---|---|---|
| Salida |  | Se utiliza en ocasiones en lugar del símbolo de salida. El dibujo representa un pedazo de hoja. Es usado para mostrar datos o resultados. |

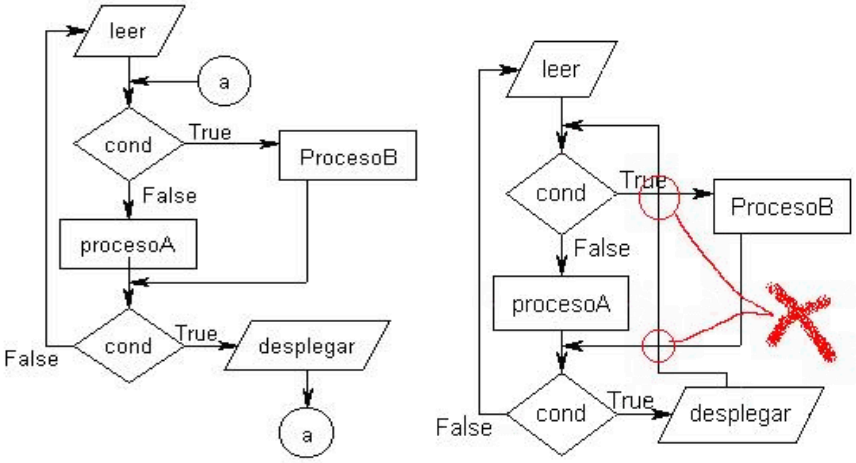
### 2.18.2.1 REGLAS DE LOS DIAGRAMAS DE FLUJO.

- Debe de indicar claramente dónde **inicia** y dónde **termina** el diagrama.
- Cualquier camino del diagrama debe de llevarle siempre a la terminal de fin.
- Organizar los símbolos de tal forma que siga visualmente el flujo de arriba hacia abajo y de izquierda a derecha.
- Las líneas deben ser verticales u horizontales, nunca diagonales.



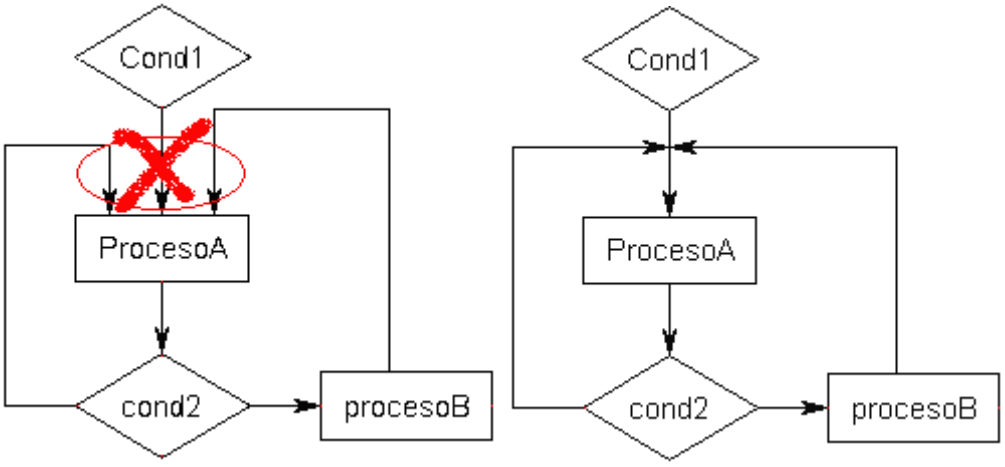


- No cruzar las líneas de flujo empleando los conectores adecuados sin hacer uso excesivo de ellos.




- No fraccionar el diagrama con el uso excesivo de conectores.

- Solo debe llegar una sola línea de flujo a un símbolo. Pero pueden llegar muchas líneas de flujo a otras líneas.




- Las líneas de flujo deben de entrar a un símbolo por la parte superior y/o izquierda y salir de él por la parte inferior y/o derecha.

- Usar lógica positiva, es decir, realizar procesos cuando es verdadera la condición y expresar las condiciones de manera clara (por ej., "no es a =/= de b" ==> "a=b").

### 2.18.3 LENGUAJE ENSAMBLADOR.

El lenguaje ensamblador es utilizado para programar los PICs. Los elementos básicos del lenguaje ensamblador son:

- ***Etiquetas***
- ***Instrucciones***
- ***Operandos***
- ***Directivas***
- ***Comentarios***

Para la programación se utiliza una cierta tabulación que se debe respetar, además utilizar una tabulación adecuada hace a los programas más claros y legibles. Las etiquetas se escriben en la primera columna de cualquier línea, las instrucciones y directivas en la segunda, y por último, en la tercera columna los operandos. Los comentarios de pueden escribir en cualquier parte del programa.

### 2.18.3.1 ETIQUETAS.

Una etiqueta es una palabra utilizada para designar alguna línea o sección del programa, se pueden utilizar para saltar de una parte hacia esa etiqueta. Es importante que las etiquetas empiecen con una letra o con un guión bajo "_". La longitud de una etiqueta puede ser de hasta 32 caracteres y como ya se dijo se deben escribir en la primer columna.

### 2.18.3.2 INSTRUCCIONES.

Las instrucciones son las operaciones que realiza el Micro controlador, así que estas ya están definidas para cada familia de PIC. El 16F877A así como todos los PICs de gama media utiliza un conjunto de 35 instrucciones que están definidas en la hoja de datos del PIC.

### 2.18.3.3 OPERANDOS.

Son los elementos que emplea la instrucción que se está ejecutando.

Usualmente los operandos son los registros, las variables o las constantes.

### 2.18.3.4 DIRECTIVAS.

Las directivas son similares a las instrucciones, pero a diferencia de estas, las directivas son propias del lenguaje ensamblador e independientes de Micro controlador que se utilice. Las directivas representan algunas características del lenguaje ensamblador, se utilizan para especificar el procesador empleado así como la configuración de este, también para asignar ubicaciones de memoria, entre otras cosas.

### 2.18.3.5 COMENTARIOS.

Los comentarios son palabras, frases y oraciones que se pueden escribir en el código para hacer el programa más claro y legible. Los comentarios de pueden escribir en cualquier parte del código pero siempre deben empezar con punto y coma “;”.

El siguiente ejemplo es un programa simple escrito en ensamblador donde se muestran los elementos básicos del lenguaje:

```
; Primer programa de prueba. Inicializa el Puerto B y pone todos los
; pines del puerto en un estado lógico "1"
; Fecha: 17.01.07   Autor: Jorge A. Bojórquez   micropic.wordpress.com

        list        p=16f628a       ; Declaración del procesador
        include     p16f628a.inc    ;
        __config    0x3F38          ; Declaración de la configuración

                                    ; Inicio del programa
        org     0x00                ; Vector de Inicio
        goto    Inicio              ; Ir a la etiqueta 'Inicio'

Inicio  bsf     STATUS,RP0          ; Seleccionar el banco de memoria 1
        clrf    PORTB               ; Configurar puerto B como salida
        bcf     STATUS,RP0          ; Seleccionar el banco de memoria 0

        movlw   0xFF                ; Cargar al acumulador W el valor 0xFF
        movwf   PORTB               ; Pone todos los pines del Puerto B en "1"

Ciclo   goto    Ciclo

        end
```

Este ejemplo se escribió usando MPLAB[3], se puede ver que MPLAB reco noce la sintaxis del lenguaje ensamblador y utiliza diferentes colores para los diferentes elementos del código. Las directivas se muestran en azul, las instrucciones en azul y en negritas, los comentarios son mostrados en verde y por ultimo las etiquetas y los operandos se muestran en color lila.

La primera directiva “list” sirve para especificar el PIC seleccionado, en el caso del ejemplo fue el 16F826a. La directiva “include” se utiliza para incluir un archivo externo en el programa. Con la directiva “__config” se establece la configuración del PIC, el tipo de reloj usado, la configuración del Watchdog, el reset interno, etc. “org” define la dirección de memoria a partir de la cual el programa se guarda y “end” es la directiva que marca el final del programa, esta directiva es muy importante y nunca debe faltar en el código, además todo lo que se escriba después de esa directiva será ignorado.

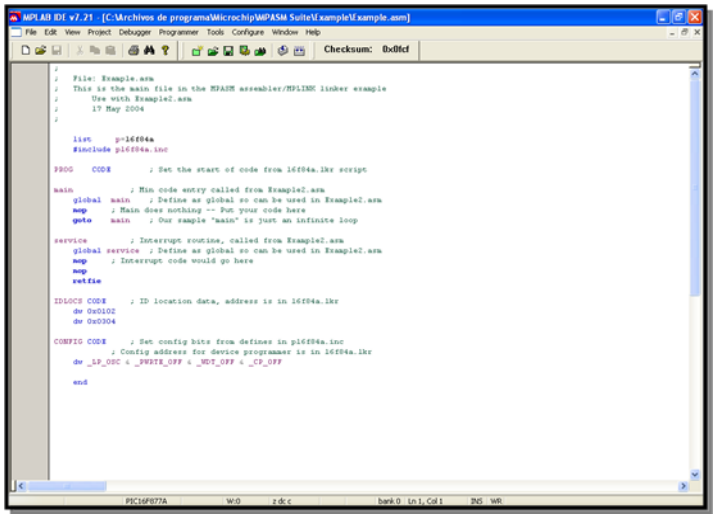
[3] Estudiado posteriormente.

Este es un programa muy simple, comienza con la directiva “org”, después la instrucción “goto” dirige el flujo del programa hacia la etiqueta “inicio”. El siguiente paso es seleccionar el banco de memoria 1 (bsf STATUS, RPO) para poder configurar el puerto B como salida (clrf PORTB), después volvemos al banco de memoria 0 (bcf STATUS, RPO) y por ultimo ponemos todos los pines del puerto B a “1” (movlw 0xFF, movwf PORTB) y con eso el programa prácticamente está finalizado. Se incluye la etiqueta “ciclo” y la ultima instrucción “goto” para que el programa se quede realizando un ciclo en ese punto (goto ciclo). Al final se incluye la directiva “end” que es necesaria para que el programa ensamblador sepa que no hay más instrucciones que interpretar.

### 2.18.4 ENTORNOS DE DESARROLLO Y SOFTWARE DE GRABACIÓN.

#### 2.18.4.1 MPLAB.

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



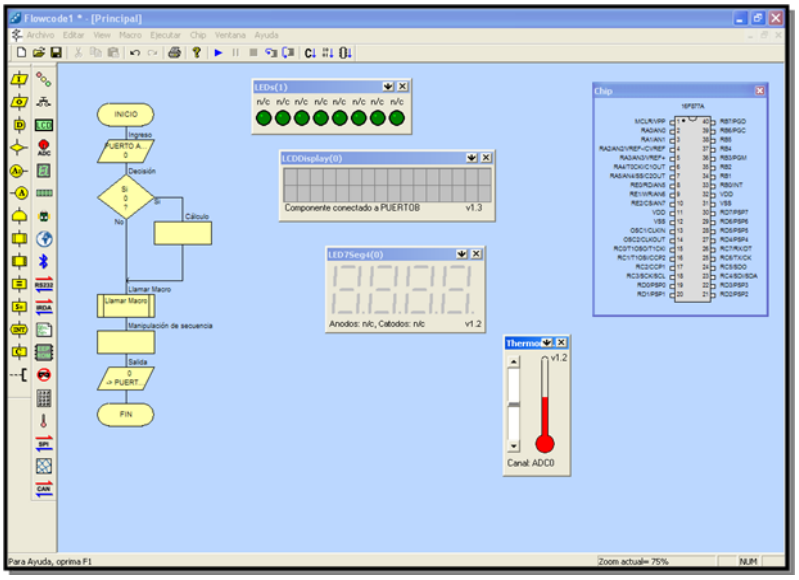

**MPLAB es un software gratuito que se encuentra disponible en la página de Microchip.**

### 2.18.4.2 FLOWCODE.

Flowcode es uno de los programas gráficos para programación de Micro controladores más avanzado. Permite crear complejos sistemas electrónicos en minutos incluso a personas con poca experiencia. Esto lo realiza por medio de dos simples pasos, el usuario arrastra los símbolos de diagramas de flujo hacia el área de trabajo, y llena los cuadros de dialogo cuando aparecen. Luego Flowcode compila el diagrama de flujo en código fuente que luego es descargado al Micro controlador, el cual ejecuta el programa.

Flowcode está destinado a usuarios familiarizados con:

- Los fundamentos básicos de la lógica digital.
- Lo que es un PICmicro.
- Lo que hace un PICmicro.
- Que los PICmicro necesitan una entrada de reloj.
- Que los PICmicro tienen circuitos internos como, por ejemplo, cronómetros, que pueden utilizar los programas.
- Los diagramas de flujo y su funcionamiento.
- Conocimientos básicos de Windows™ como copiar, pegar, etc.
- Circuitos electrónicos simples como, por ejemplo, LED, interruptores, transistores utilizados como interruptores, etc.



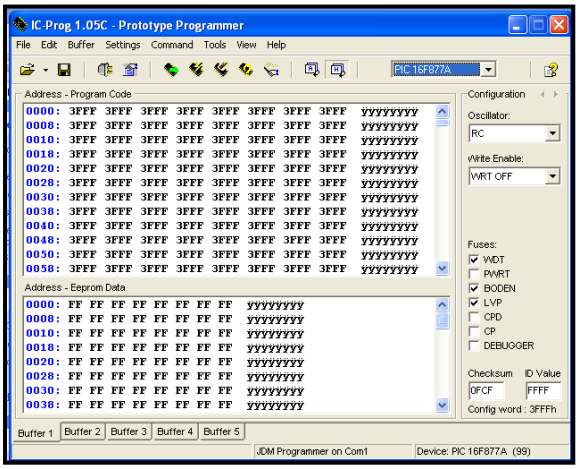

**Flowcode es uno de los programas gráficos para programación de Micro controladores más avanzado**

### 2.18.4.3 IC-PROG

El IC-Prog es un programa que funciona bajo Windows para controlar un programador de Micro controladores PIC. Para operar este programa se necesitan conocimientos básicos de Windows y de electrónica.

Para que el programa funcione se deberá conectar a la computadora un programador, y configurar correctamente tanto a éste como al programa. Por favor note que, debido a la variedad de programadores y sus diferencias el programa puede no funcionar con ciertas combinaciones de computadoras y equipos programadores. El IC-Prog requiere Windows 95, 98, ME, NT, o 2000 y un coprocesador interno o externo para funcionar.

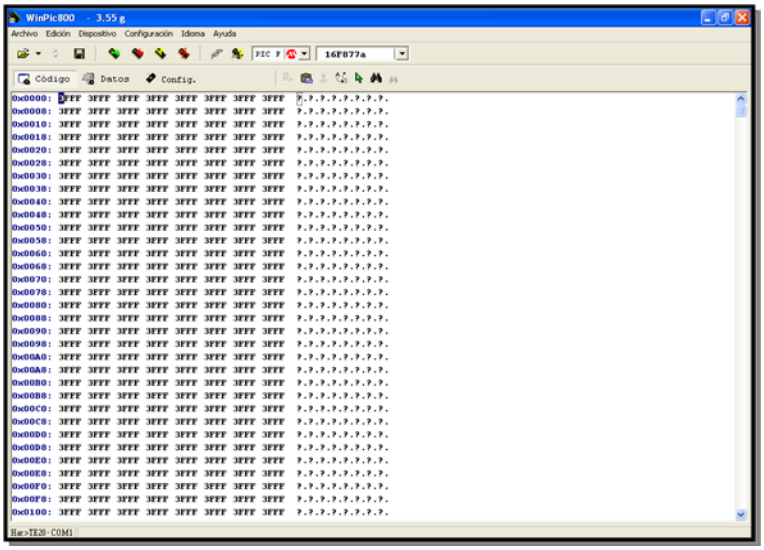
El IC-Prog es un programa registrado aunque es de libre distribución. Se permite su copia y distribución en la medida en que no se lo modifique de manera alguna, se cobre por su uso o se utilice para fines ilegales. El IC-Prog ha sido designado como una aplicación de programación universal para todos los programadores.




**El IC-Prog es un programa que funciona bajo Windows para controlar un programador de Micro controladores**

### 2.18.4.4 WINPIC800.

WinPic800 al igual que IC-Prog es un software diseñado para programar Micro controladores, usando un dispositivo grabador de chips el cual puede ser conectado por medio de un puerto serial, paralelo o USB. WinPic800 soporta una gran variedad de Micro controladores PIC, así como también muchos equipos grabadores. WinPic800 posee una sencilla interfaz gráfica la cual nos permite visualizar el programa (en Hex) que deseamos introducir en el Micro controlador, de la misma forma nos permite ver la información contenida dentro de un Micro controlador grabado previamente. WinPic800 es freeware, por lo tanto es gratuito y es posible distribuirlo son ningún inconveniente.




**WinPic800 al igual que IC-Prog es un software diseñado para programar Micro controladores**

## 2.18.5 PROGRAMADORES (HARDWARE DE GRABACIÓN).

Para transferir el código de una computadora al PIC normalmente se usa un dispositivo llamado programador. La mayoría de PICs que Microchip distribuye hoy en día incorporan ICSP (*In Circuit Serial Programming*, programación serie incorporada) o LVP (*Low Voltage Programming*, programación a bajo voltaje), lo que permite programar el PIC directamente en el circuito destino. Para la ICSP se usan los pines RB6 y RB7 como reloj y datos y el MCLR para activar el modo programación aplicando un voltaje de unos 11 voltios. Existen muchos programadores de PICs, desde los más simples que dejan al software los detalles de comunicaciones, a los más complejos, que pueden verificar el dispositivo a diversas tensiones de alimentación e implementan en hardware casi todas las funcionalidades. Muchos de estos programadores complejos incluyen ellos mismos PICs pre programados como interfaz para enviar las órdenes al PIC que se desea programar. Uno de los programadores más simples es el TE20, que utiliza la línea TX del puerto RS232 como alimentación y las líneas DTR y CTR para mandar o recibir datos cuando el Micro controlador está en modo programación. El software de programación puede ser el ICprog o WinPic800 muy comunes entre la gente que utiliza este tipo de Micro controladores.

Los siguientes son algunos de los programadores más comunes:

### 2.18.5.1 PRO MATE II.

El programador PRO MATE II le permite programar su software dentro de cualquier Micro controlador PIC de manera fácil y rápida. PRO MATE II utiliza el software de desarrollo MPLAB IDE y opera como un dispositivo independiente o en conjunto con una computadora personal. Cabe mencionar que este dispositivo no se encuentra más en producción. Estas son algunas de sus características:

- Soporta ICSP (In-Circuit Serial Programming).
- Conexión por medio de cable serial.
- Completa línea de sockets intercambiables para el soporte de los diversos PICs.
- Compatibilidad completa con el software MPLAB IDE.


**PRO MATE II le permite programar su software dentro de cualquier Micro controlador**

### 2.18.5.2 PICSTART PLUS.

El PICSTART Plus es un programador de bajo costo. Se conecta por medio de un cable serial RS-232 y opera con es software de desarrollo MPLAB IDE. El programador soporta la mayoría de Micro controladores PIC de Microchip. Estas son algunas de sus características:

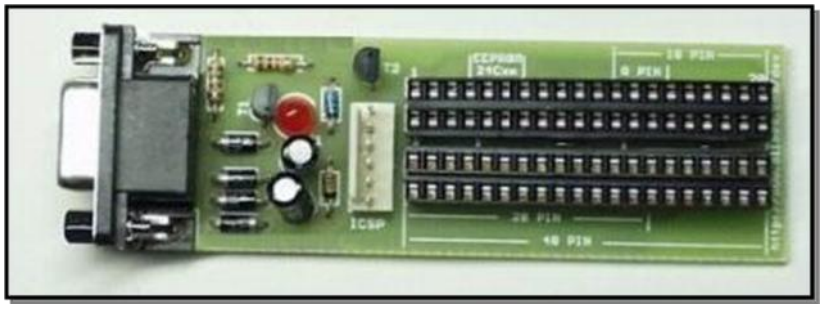
- Opera con cualquier computadora personal con sistema operativo Microsoft Windows.
- Lee, programa y verifica los datos del PIC.
- Utiliza cable serial RS-232 y fuente de poder de 9V.
- Soporta los proyectos de MPLAB y los descarga automáticamente al Micro controlador.


**PICSTART Plus es un programador de bajo costo**

### 2.18.5.3 PROGRAMADOR JDM.

Este es un programador no propietario, es decir, puede ser fabricado por cualquiera. Funciona por medio de el cable serial RS-232 de el cual toma las señales necesarias, incluso la alimentación eléctrica por lo tanto no necesita fuente externa de alimentación.

Soporta PICs de 8, 18, 28 y 40 pines, posee conector ICSP (In Circuit Serial Programming). Es compatible con cualquiera del software de programación existente. Cabe mencionar que este dispositivo puedo no funcionar con algunas laptops debido a la baja potencia de los puertos de estas.




**PROGRAMADOR JDM**

### 2.18.6 LOS ENTRENADORES.

Los entrenadores son equipos diseñados para facilitar el aprendizaje con los PIC, permite conectar un Micro controlador a diferentes periféricos como Leds, pantalla LCD, interruptores, etc. De esa forma es posible demostrar las diferentes aplicaciones de los Micro controladores de una forma fácil y segura. Existen varias versiones de entrenadores, algunos permiten trabajar con la mayoría de los Micro controladores de Microchip Technology. A continuación mencionamos dos de los entrenadores más completos con sus respectivas características.

#### 2.18.6.1 E-BLOCKS EB006 MULTIPROGRAMADOR

Este equipo actúa como el corazón de cualquier sistema de E-Blocks (módulos de expansión). Provee hasta cinco puertos de salida y de poder regulados de 5V (hasta un máximo de 400mA) provenientes de una sola fuente de voltaje, para conectar, dar poder y controlar otros módulos de E-Blocks.

El E-Block EB006 Multiprogramador y Modulo de Proceso USB puede programar Micro controladores PIC Microchip de 8, 14, 18, 28 y 40 pines usando el flexible software de programación incluido, y provee acceso a todas las líneas de entrada y salida del Micro controlador. Conectando módulos E-Block adicionales, esta tarjeta puede ser expandida convirtiéndose en una solución configurable para propósitos de desarrollo y aprendizaje.

1. Conector de poder.

2. Conector USB.

3. Botón de Reset.

4. Puerto E – de hasta 3 líneas.

5. Puerto A – de hasta 5 líneas.

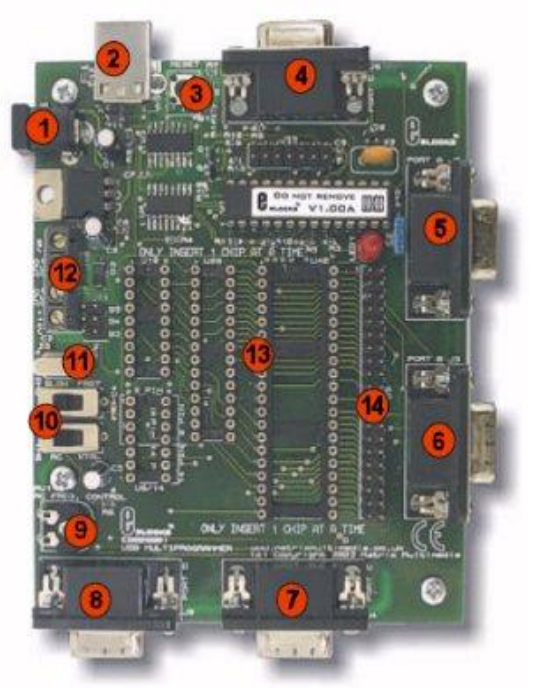
6. Puerto B – de hasta 8 líneas.

7. Puerto C – de hasta 8 líneas.

8. Puerto D – de hasta 8 líneas.

9. Control de velocidad de reloj RC.

10. Botón de selección de oscilador.




11. Cristal ubicado en zócalo – Este puede ser removido para diferentes aplicaciones.

12. Terminales con tornillos para conexiones de poder – Incluye salida de 5V.

13. Zócalo para Micro controladores PICmicro – Soporta dispositivos de 8, 14, 18, 28 y 40 pines. Solo es posible conectar un Micro controlador a la vez.

### 2.18.6.2 PICDEM 2 PLUS

El PICDEM 2Plus es una tarjeta simple que demuestra las capacidades de muchos de los Micro controladores de 18, 28 y 40 pines de las familias PIC16XXXX y PIC18XXXX. Trabaja junto con una computadora personal por medio de un cable serial RS-232, necesita fuente de alimentación externa de 9V o puede trabajar con una batería de 9V. El entrenador PICDEM 2 Plus posee las siguientes características:

1. Sockets para PICs de 18, 28 y 40 pines. (A pesar de poseer tres sockets, es permitido trabajar con un solo PIC a la vez.

2. Entrada directa de 9V, 100 mA AC/DC con adaptador de pared o una batería de 9V.

3. Conector RS-232 para conexión directa.

4. Conector de expansión.

5. Potenciómetro para dispositivos con entradas analógicas.

6. Tres interruptores para estimulación externa y reset.

7. LED indicador de encendido.

8. Cuatro LED conectados a PORTB.

9. Jumper J6 desconecta los LED de PORTB.

10. Oscilador de cristal de 4 MHz

11. Agujeros disponibles para la conexión de cristales.

12. Cristal de 32.768 kHz para la operación del reloj Timer1.

13. Jumper J7 desconecta el oscilador RC incluido.

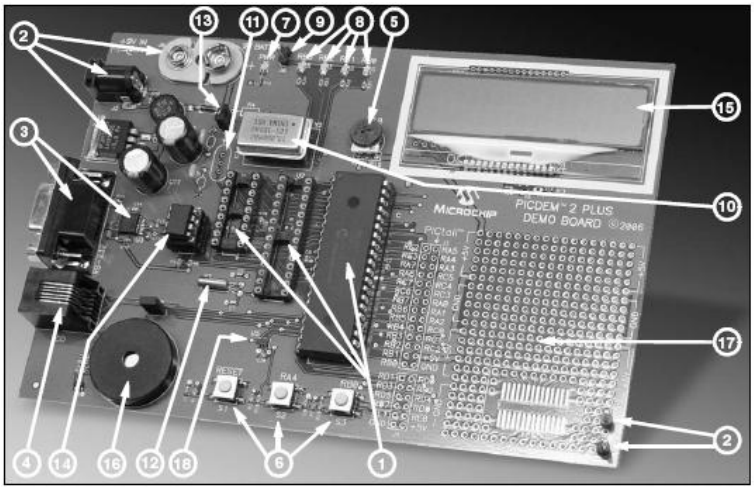
14. EEPROM Serial 32K x 8.

15. Pantalla LCD.

16. Timbre.

17. Área para prototipos.

18. Sensor térmico Microchip TC74.




**El PICDEM 2Plus**

# Capitulo 3

Análisis de la información recolectada en la Facultad Multidisciplinaria de Occidente e Instituciones Educativas con Laboratorios de Electrónica

## 3.1 INTRODUCCIÓN

La información que se presenta en este capitulo sobrelleva criterios congruentes en la aplicación de prácticas de laboratorio con Micro controladores en la asignatura de Microprogramación.

Entre los puntos que se tocan en este capítulo se encuentran: los antecedentes de la situación actual, el diagnóstico de la problemática, la metodología de la investigación en donde se establece la lógica seguida en el proceso de recopilación de información y se describen los métodos utilizados para obtenerla. Luego se identifican las fuentes de estudio, el ámbito de la investigación, el método estadístico, y el cálculo de la muestra de la población.

Posteriormente se presenta la tabulación y el análisis de datos, junto con los resultados obtenidos.

## 3.2 ANTECEDENTES DE LA SITUACIÓN ACTUAL.

La carrera de Ingeniería de Sistemas Informáticos ha sido un proyecto en la FMO impulsado por los catedráticos del Departamento de Ingeniería y Arquitectura, con el propósito de brindar una educación en el área de informática a los estudiantes de la región occidental y que éstos posean una opción más de estudio en la Facultad Multidisciplinaria de Occidente.

Desde el comienzo de la carrera en la FMO, se le ha dado más importancia a las prácticas de asignaturas relacionadas con el manejo de paquetes de software y al área de programación. Los recursos disponibles para desarrollar la carrera influyeron en gran parte a esa orientación. Pero actualmente el campo laboral en nuestro medio exige que un Ingeniero de Sistemas Informáticos posea conocimientos prácticos en el área de hardware.

A partir del año 2000 la carrera ha evolucionado de gran manera, con el desarrollo de cursos extracurriculares, orientados a estudiantes de tercer año de la carrera; estos cursos comprenden el mantenimiento de computadoras y el área de redes de computadoras.

En el año 2004 las asignaturas de Sistemas Digitales y Arquitectura de Computadores cuentan con un Auxiliar de Cátedra, encargado de la realización de las prácticas de laboratorio.

Para la realización de las prácticas de las asignaturas de Sistemas Digitales y Arquitectura de Computadores se cuenta con un espacio en el Edificio de Usos Múltiples de la FMO, el cual está destinado para la

implementación de un laboratorio de cómputo, ya que cuenta con la infraestructura para el montaje de la red y del equipo informático.

El Departamento de Ingeniería y Arquitectura cuenta con un local donde se proporciona mantenimiento al equipo informático y que es llamado Laboratorio de Hardware pero que no tiene los requisitos necesarios para desarrollar prácticas de las asignaturas relacionadas con el hardware.

## 3.3 METODOLOGÍA DE LA INVESTIGACIÓN

Existen diferentes tipos de investigación entre las cuales tenemos:

La investigación exploratoria: El objetivo es examinar un tema o problema de investigación poco estudiado, y nos sirve para familiarizarnos con fenómenos relativamente conocidos.

La investigación Descriptiva: La finalidad es buscar y especificar las propiedades importantes de personas, grupos, comunidades o cualquier otro fenómeno que sea sometido a análisis.

La investigación Correlacional: El propósito es medir el grado de relación que exista entre dos o más conceptos o variables, lo que se pretende en este tipo de investigación es determinar como se puede comportar un concepto o una variable conociendo el comportamiento de otras variables relacionadas.

La investigación Explicativa: Se dirige a responder a las causas de los eventos físicos o sociales. Como su mismo nombre lo indica, se centra en explicar porque ocurre un fenómeno y en que condiciones se da este.

Partiendo del hecho que una investigación puede incluir elementos de los cuatro tipos de estudios, es decir abarcar fines exploratorios, descriptivos, correlaciónales y explicativos; tomando en cuenta que su clasificación depende del conocimiento actual del tema de investigación, y del enfoque que el investigador pretenda darle, se determina que el presente estudio puede definirse como una investigación social “Exploratoria – Descriptiva”, la cual tiene las siguientes características:

1. No esta orientada a la comprobación de una hipótesis.
2. El problema se caracteriza a partir de la situación actual para determinar su diagnóstico y posterior determinación de requerimientos.
3. No pierde su carácter exploratorio, aun con investigaciones parecidas; no orientadas específicamente a la aplicación de un laboratorio de especialización.

### 3.3.1 ELEMENTOS DE LA INVESTIGACIÓN

Los elementos que serán investigados en el siguiente capítulo son:

La comunidad estudiantil de la Facultad Multidisciplinaria de Occidente que cursa la carrera de Ingeniería de Sistemas Informáticos, a los cuales se les aplicara una encuesta como involucrados en la implementación, Instituciones educativas que cuentan con laboratorios de especialización de electrónica.

### 3.3.2 FUENTES DE INFORMACIÓN

En primera instancia es importante acudir a fuentes de información que proporcionen datos objetivos y concretos que den apoyo en la investigación de campo, como también para identificar otros elementos que pueden ser incorporados en la investigación, ya sea teórica o en visitas técnicas. Para ello se determinan dos tipos de fuentes: fuentes de información primaria y fuentes de información secundaria.

#### 3.3.2.1 FUENTES PRIMARIAS

Comprenden aquellas entidades y/o personas que proporcionan información teórica por medio de documentos técnicos que son de interés para el desarrollo del trabajo, además de la información directa que ellos conocen del trabajo que ejecutan; entre las fuentes de investigación primaria que serán parte primordial de esta investigación tenemos:

a) La comunidad estudiantil y docente del Departamento de Ingeniería de la Facultad Multidisciplinaria de Occidente; como involucrados en la propuesta.

b) Los sitios Web oficiales que proporcionan las Hojas de Datos técnicos de los Micro controladores que se utilizaran en el diseño de las guías de Laboratorio y prácticas propuestas.

c) Las Instituciones educativas que cuenten con Laboratorios de Electrónica, que nos proporcionen datos en las visitas técnicas.

### 3.3.2.2 FUENTES SECUNDARIAS

Son importantes como referencia, para la ubicación de las fuentes de investigación primaria, ya que contienen los listados de dichas fuentes, donde se resumen los temas más importantes de que tratan, algunos ejemplos de estas fuentes son las siguientes: bibliotecas que proveen libros, tesis y revistas, así como también el Internet.

### 3.3.3 INVESTIGACIÓN DOCUMENTAL

La información documental a la cual hemos tenido acceso para el desarrollo de la investigación es:

- Hojas de Datos técnicos de los Micro controladores propuestos
- Manual de uso para el Software propuesto
- Base legal para la implementación de Software con Licencia
- Programa y Plan de la signatura de Microprogramación.

### 3.3.4 INVESTIGACIÓN BIBLIOGRÁFICA

La investigación documental consistió en hacer una revisión y análisis de los trabajos de grado, revistas, páginas Web, documentos, libros, con lo cual se han logrado definir todo lo concerniente a las generalidades del trabajo con micro controladores, programadores, entrenadores y software relacionado.

### 3.3.5 ENTREVISTAS

Las entrevistas han sido dirigidas a las personas involucradas en el trabajo con laboratorios de electrónica, ya sea en las instituciones visitadas, así como a una parte de la comunidad estudiantil de La Universidad de El Salvador en la Unidad Central.

### 3.3.6 ENCUESTAS

Las encuestas como se mencionó anteriormente de forma generalizada han sido dirigidas a la comunidad universitaria (estudiantes) de la carrera de Ingeniería de Sistemas Informáticos de la Facultad Multidisciplinaria de Occidente (Ver anexos). Con esta encuesta, se pretende dar un dato que permita valorar si los laboratorios de prácticas aquí en la facultad agregan ese valor extra al cumplimiento de objetivos de asignatura como motivación en los estudiantes para una conciencia de investigación y experimentación. Esta información será el valor agregado para un fundamento sólido en la implantación de un laboratorio de prácticas con micro controladores. El diseño de las encuestas se ha hecho con preguntas cerradas, con el objetivo de no involucrarse en opiniones dividas afectadas por agentes externos.

### 3.3.7 DETERMINACIÓN DEL UNIVERSO Y TAMAÑO DE MUESTRA

Como se ha mencionado anteriormente en esta investigación utilizaremos dos tipos de fuentes de información: fuentes de información primarias y fuentes de investigación secundarias, entre las primarias tenemos a los estudiantes de la carrera de Ingeniería de Sistemas Informáticos de la Facultad Multidisciplinaria de Occidente, a continuación se detalla el cálculo del tamaño de muestra requerido para el estudio.

### 3.3.7.1 SECTOR ESTUDIANTIL:

Antes de calcular el tamaño de la muestra tanto para el sector estudiantil se han tomado en cuenta las siguientes consideraciones:

- El nivel de confianza que se ha utilizado es el 95% para obtener una mayor representatividad en los datos.
- El porcentaje de error calculado es del 10%, con lo cual se obtiene un riesgo mínimo de sesgo en la información recolectada.
- Los valores de P y Q que representan la probabilidad de éxito y de fracaso en un experimento se han asignado en P = 50% y Q = 50%.
- El sector estudiantil que se estudiara esta compuesto por alumnos de la carrera de Ingeniería de Sistemas Informáticos, se considera la población de 390 alumnos, la fórmula para determinar el tamaño de la muestra es la siguiente:

$$n = \frac{Z^2 PQN}{E^2 (N-1) + Z^2 PQ}$$

Donde:

n= Tamaño de la muestra ¿?
Z= Nivel de Confianza = 95% =1.96
P= Porcentaje de aceptación = 50%
Q= Porcentaje de Rechazo = 50%
E= Error porcentual permisible = 10%
N= Universo de estudiantes = 390

Sustituyendo los valores en la formula:

$$n = \frac{(1.96)^2 (0.5)\ (0.5)\ (390)}{(0.10)^2 (390\text{-}1) + (1.96)^2\ (0.5)\ (0.5)}$$

**n = 77 estudiantes de Ingeniería de Sistemas Informáticos**

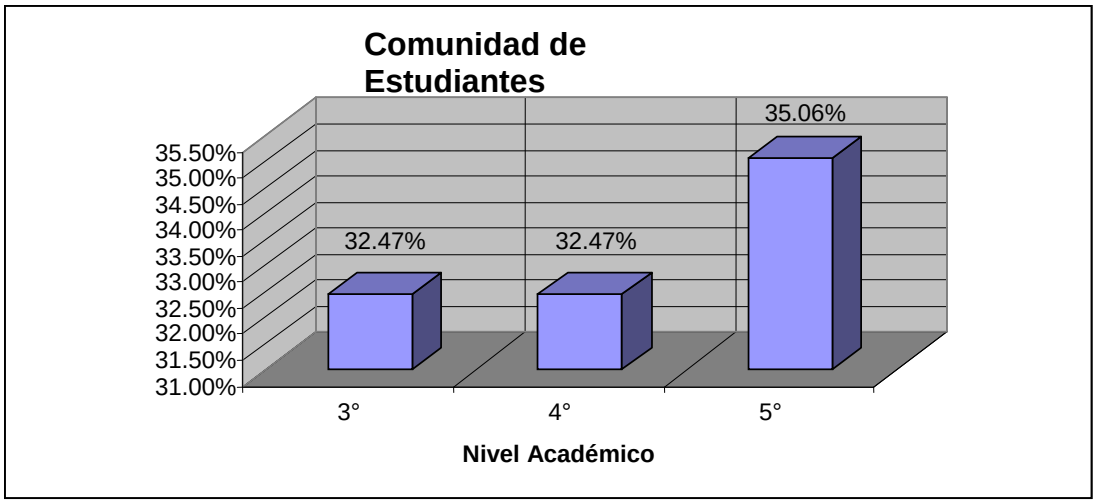
### 3.3.7.2 TABULACIÓN Y ANÁLISIS DE DATOS

Una vez aplicado el instrumento de recolección de datos, se procede a la interpretación de resultados, interpretando cada pregunta mostrándolas en el orden establecido.

Dicha interpretación consiste en la presentación de cada pregunta con su respectivo objetivo y con su respectiva tabla de datos, incluyendo frecuencias, porcentajes y en las que sea necesario una clasificación de acuerdo a códigos. Además se presenta un gráfico como ayuda visual para observar y comprender con mayor facilidad los resultados obtenidos. A continuación se presenta un comentario por cada pregunta, cuyo propósito es describir en forma general los resultados obtenidos, así como también realizar las observaciones adicionales que se consideren pertinentes.

Interpretación de Encuesta dirigida a los estudiantes de Ingeniería de Sistemas Informáticos.

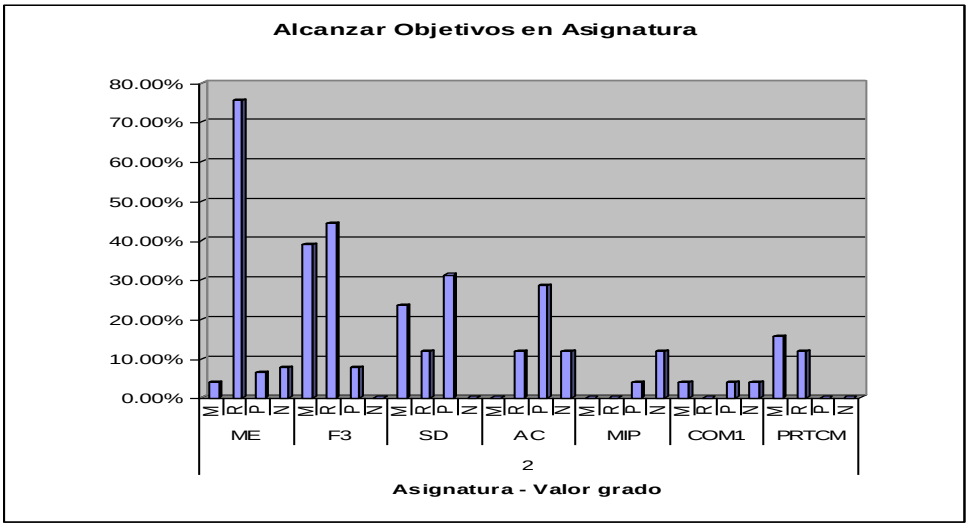
1.- En que nivel de la carrera de INGENIERÍA DE SISTEMAS INFORMÁTICOS se encuentra actualmente.




| 1 | | |
|---|---|---|
| **3°** | **4°** | **5°** |
| **32.47%** | **32.47%** | **35.06%** |
| 25 | 25 | 27 |

Se tomo una muestra equivalente a una población de 25 alumnos pertenecientes al nivel académico de tercer año de la carrera, 25 alumnos pertenecientes al nivel académico de cuarto año de la carrera y 27 alumnos pertenecientes al nivel académico de quinto año de la carrera.

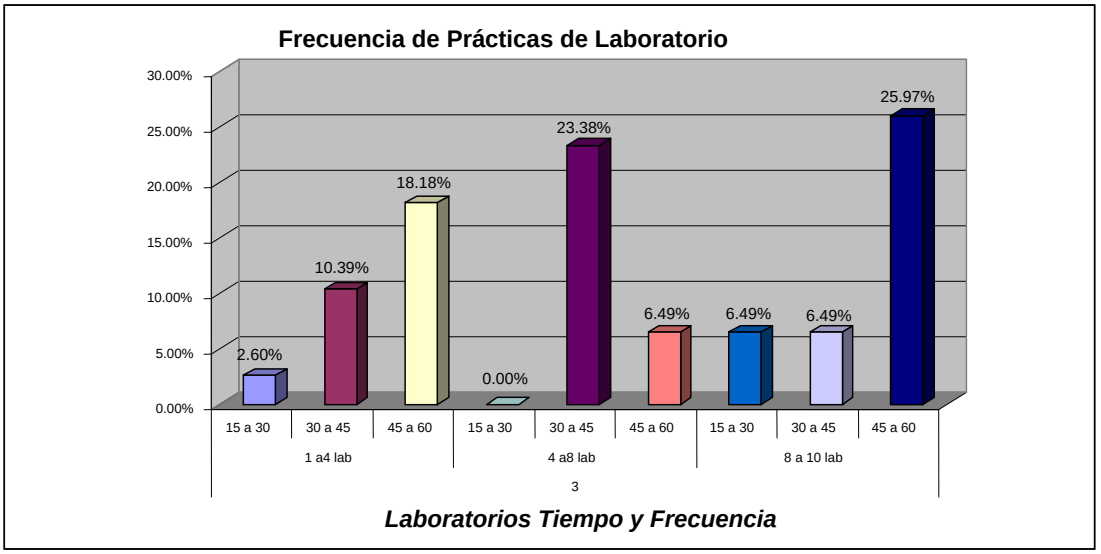
2.- Especifique las asignaturas en las cuales a realizado prácticas de laboratorio y determine el grado en que las prácticas ayudaron a alcanzar los objetivos de la asignatura.




| Métodos Experimentales | | | | Física 3 | | | | Sistemas Digitales | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| M | R | P | N | M | R | P | N | M | R | P | N |
| 3.90% | 75.32% | 6.49% | 7.79% | 38.96% | 44.16% | 7.79% | 0.00% | 23.38% | 11.69% | 31.17% | 0.00% |
| 3 | 58 | 5 | 6 | 30 | 34 | 6 | 0 | 18 | 9 | 24 | 0 |
| **Arquitectura de Computadores** | | | | **Microprogramación** | | | | **Comunicaciones 1** | | | |
| M | R | P | N | M | R | P | N | M | R | P | N |
| 0.00% | 11.69% | 28.57% | 11.69% | 0.00% | 0.00% | 3.90% | 11.69% | 3.90% | 0.00% | 3.90% | 3.90% |
| 0 | 9 | 22 | 9 | 0 | 0 | 3 | 9 | 3 | 0 | 3 | 3 |
| **Protocolos de Comunicación.** | | | | **M: Mucho** | | | | | | | |
| M | R | P | N | **R: Regular** | | | | | | | |
| 15.58% | 11.69% | 0.00% | 0.00% | **P: Poco** | | | | | | | |
| 12 | 9 | 0 | 0 | **N: Nada** | | | | | | | |

El 93% de los encuestados afirmo haber recibido por lo menos en una asignatura prácticas de laboratorio a lo largo de su estadía en la Universidad.

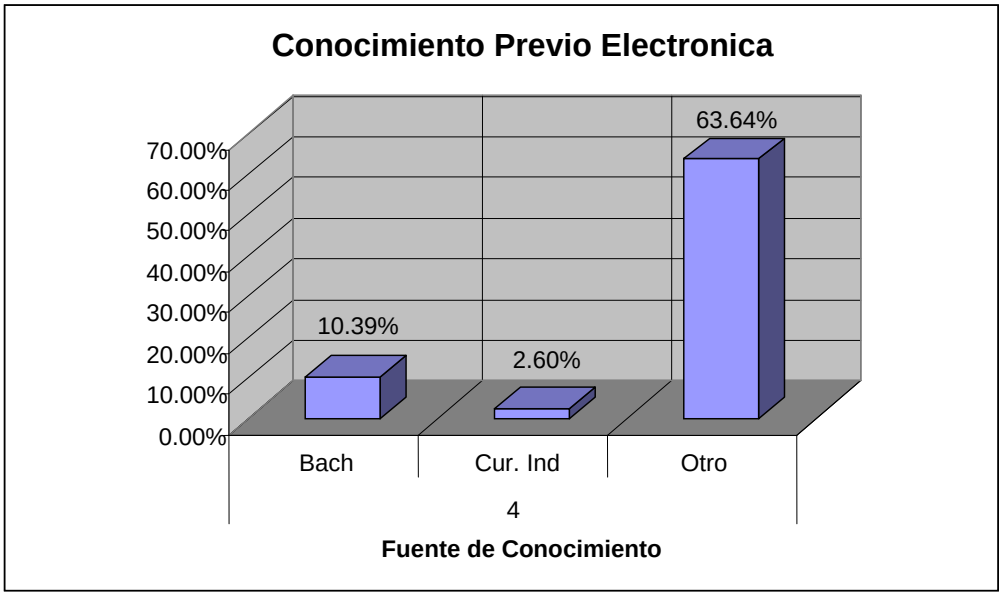
3.- Establezca a su criterio, la cantidad de prácticas de laboratorio por ciclo y tiempo en minutos de cada una.




| 1 a4 LAB | | | 4 a8 LAB | | | 8 a 10 LAB | | |
|---|---|---|---|---|---|---|---|---|
| **15 a 30** | **30 a 45** | **45 a 60** | **15 a 30** | **30 a 45** | **45 a 60** | **15 a 30** | **30 a 45** | **45 a 60** |
| **2.60%** | **10.39%** | **18.18%** | **0.00%** | **23.38%** | **6.49%** | **6.49%** | **6.49%** | **25.97%** |
| 2 | 8 | 14 | 0 | 18 | 5 | 5 | 5 | 20 |

La grafica revela que las preferencias tienden a un aumento directamente proporcional en positivo a la cantidad de laboratorios dentro de un ciclo de estudio y una tendencia a aumentar el tiempo efectivo en minutos de cada laboratorio.

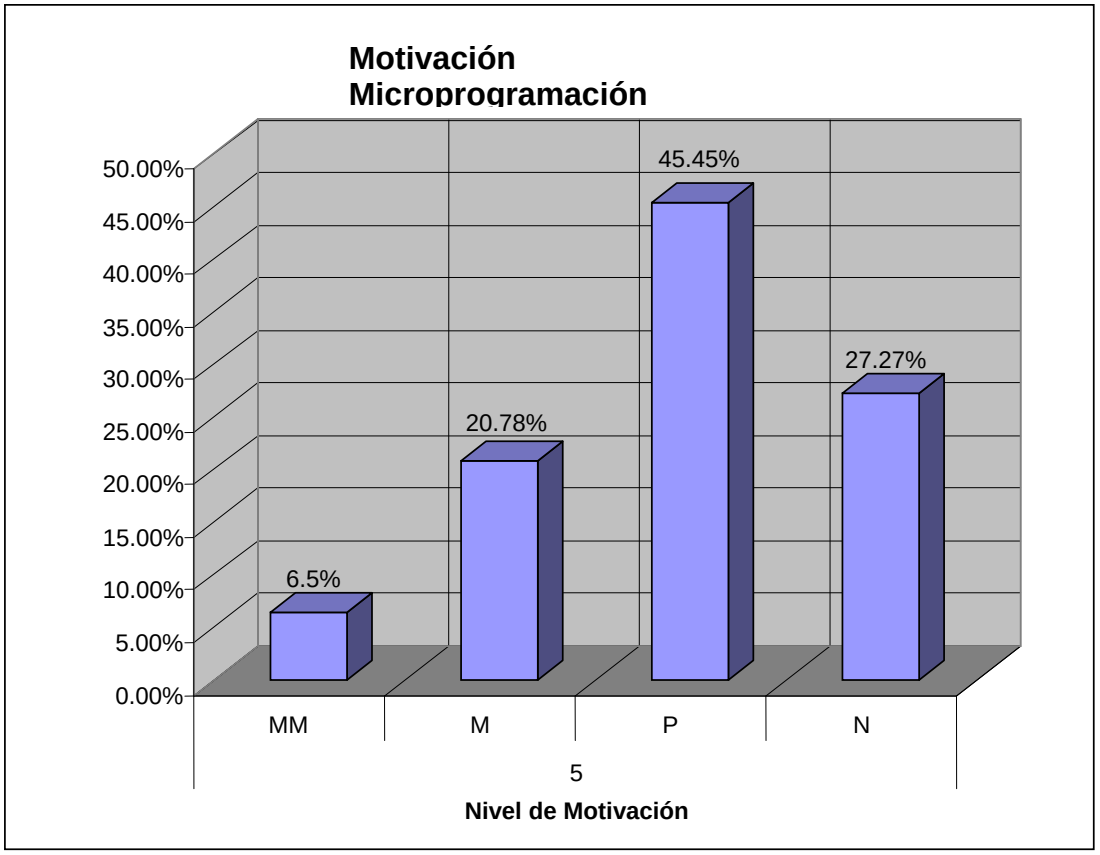
4.- ¿Tenía conocimientos previos de Electrónica antes de cursar la carrera?




| Bachillerato | Cursos Extracurriculares | Otro medios |
|---|---|---|
| **10.39%** | **2.60%** | **63.64%** |
| 8 | 2 | 49 |

El 76.62% de los encuestados afirma poseer conocimientos previos de electrónica, de lo cual se desglosa de este total que el 83.05% a adquirido dicho conocimiento por fuentes externas, el 13.55% lo obtuvo en su bachillerato y el 3.4% en cursos extracurriculares. Nos presenta una tendencia a que el alumno busca medios externos a la educación formal para la adquisición de conocimientos relacionados a la electrónica, los cuales pueden ofrecer en su mayoría una interacción directa con hardware.

5.- ¿Que grado de motivación presenta/o para cursar la asignatura de Microprogramación?

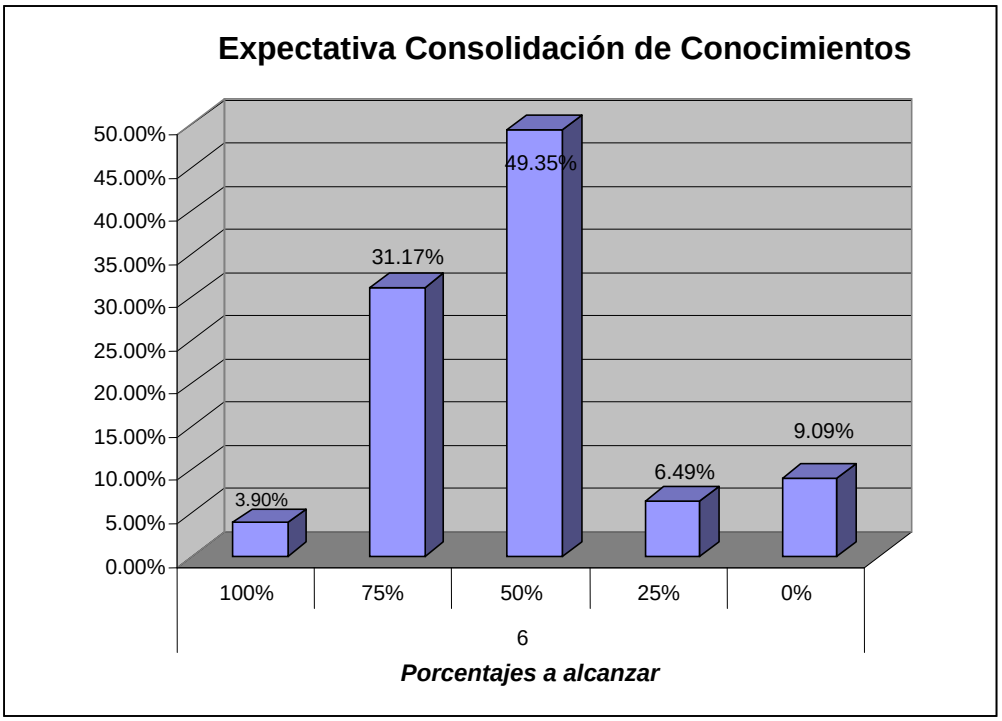



| Muy Motivado | Motivado | Poco Motivado | Nada Motivado |
|---|---|---|---|
| **6.49%** | **20.78%** | **45.45%** | **27.27%** |
| 5 | 16 | 35 | 21 |

El 100% de los encuestados afirmo encontrarse en un grado de motivación para cursar la asignatura de Microprogramación; la tendencia revela que el 6.5% se encuentra muy motivado, aun cuando representan a una población que se encuentra en un nivel académico de tercer año de la carrera, el 20.78% se encuentra motivado para cursar la asignatura, estos representan en su mayoría a alumnos que se encuentran entre el tercer y cuarto año de la carrera; el punto critico se muestra cuando el 45.45% se

siente poco motivado de cursar la asignatura de microprogramación, de este porcentaje las dos terceras partes se ubican dentro de la población de cuarto y quinto año de la carrera, de igual forma para el 27.27% que se ubica en que no se encuentran nada motivados para cursar la asignatura. Nos presenta una tendencia a que los alumnos que aun no han cursado la asignatura pero están cerca de llegar, tienen una expectativa positiva de lo que la asignatura ofrece.

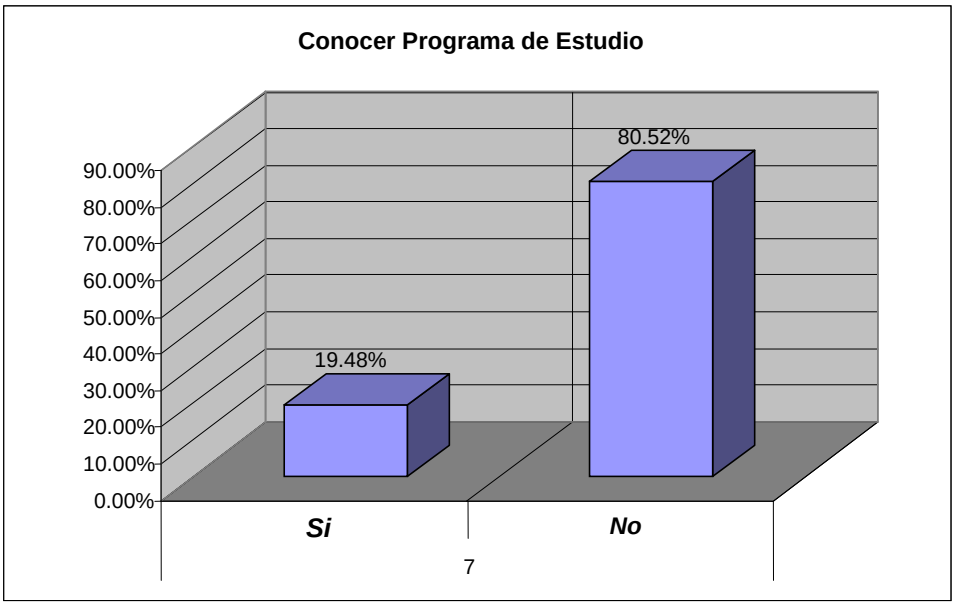
6.- ¿Que expectativa en % considera que las practicas de laboratorio de Microprogramación podría consolidar el conocimiento impartido en el aula, en el tiempo de 1 ciclo de estudio?




| 100% | 75% | 50% | 25% | 0% |
|---|---|---|---|---|
| 3.90% | 31.17% | 49.35% | 6.49% | 9.09% |
| 3 | 24 | 38 | 5 | 7 |

La grafica muestra una tendencia a mantenerse en un centro equitativo para el 49.35% de consolidar un 50% de los conocimientos en una asignatura que se apoye con prácticas de laboratorio, se presenta una tendencia en positivo de un 31.17% de los encuestados que consideran que se puede lograr consolidar el 75% de los conocimientos en las asignaturas con apoyo de prácticas de laboratorio y un decremento de porcentaje de alumnos equivalente al 6.5% que tiene expectativas de consolidar un 25% de conocimientos y un porcentaje de un 9% que considera que las prácticas de laboratorio aporten algo para obtener conocimiento nuevo.

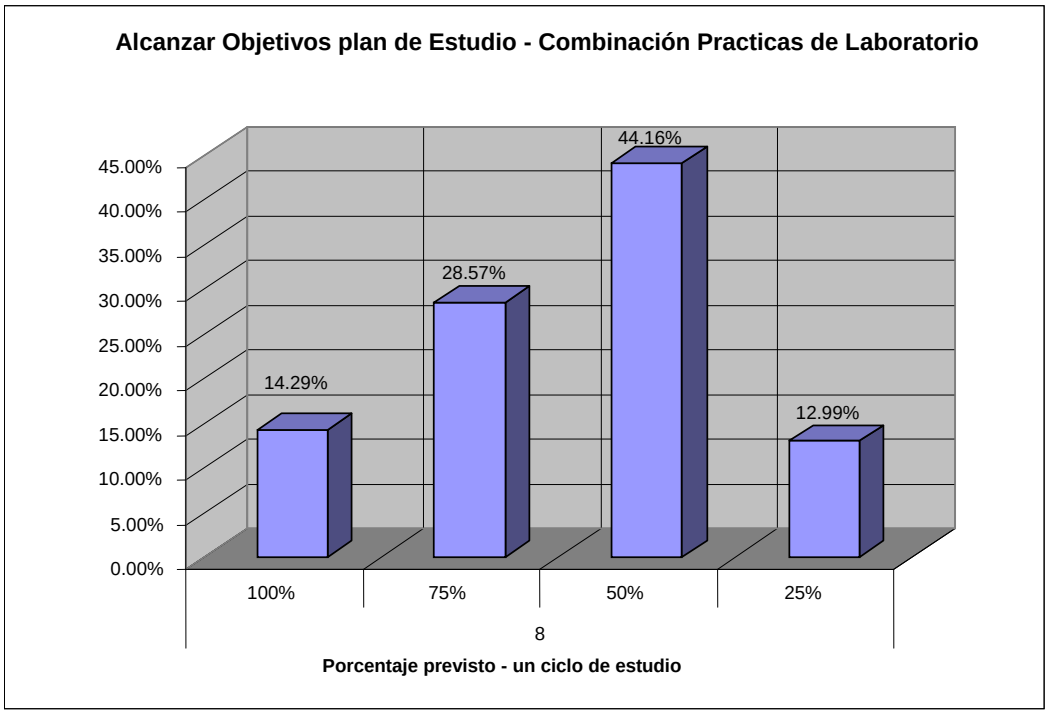
7.- ¿Utiliza/o el programa de estudio de la asignatura de Microprogramación durante la impartición de los conceptos teóricos?




| Nivel | si | no |
|---|---|---|
| 3er Año | 0 | 25 |
| 4to Año | 3 | 22 |
| 5to Año | 12 | 15 |
| Total | 15 | 62 |
| Porcentaje | 19.48% | 80.52% |

El conocer el programa de una asignatura provee al alumno una herramienta valiosa que le permite poder anticipar y prever contratiempos y limitantes para cursar la materia; del total de encuestados el 80.52% no conoce el programa de la asignatura, mientras que el 19.48% si lo conoce, pero cabe destacar que este porcentaje se compone del 3.9% pertenecientes al cuarto año lectivo de la carrera y el 15.58% pertenece al quinto año lectivo de la carrera.

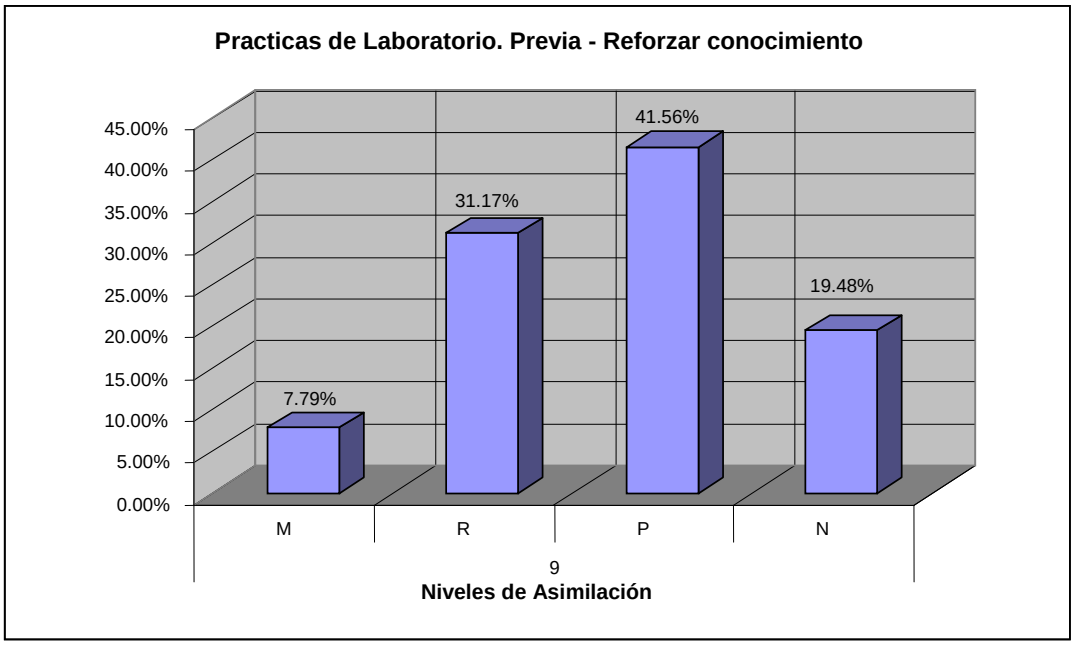
8.- En que medida considera la combinación de prácticas de laboratorio y los temas expuestos en el plan de estudio puedan favorecer en un ciclo del año electivo en alcanzar los objetivos del plan de estudio




| 100% | 75% | 50% | 25% |
|---|---|---|---|
| 14.29% | 28.57% | 44.16% | 12.99% |
| 11 | 22 | 34 | 10 |

Las tendencias de los encuestados ubican a la mayoría dentro de un centro con tendencia positiva a lo que puedan esperar de la combinación de prácticas en los laboratorios y la teoría en el aula; se presentan como un 14.29% que se ubica en poder alcanzar el 100% de los objetivos del plan de estudio, un 28.57% que pretende alcanzar el 75% de los objetivos, un 44.15% que se mantiene en lograr el 50% de los objetivos y solamente un 12.99% que se ubica en lograr únicamente el 25% de los objetivos del plan. Esta tendencia nos presenta un dato objetivo de que la mayoría confía en que las prácticas de laboratorio pueden llegar a ser un factor decisivo en su formación profesional.

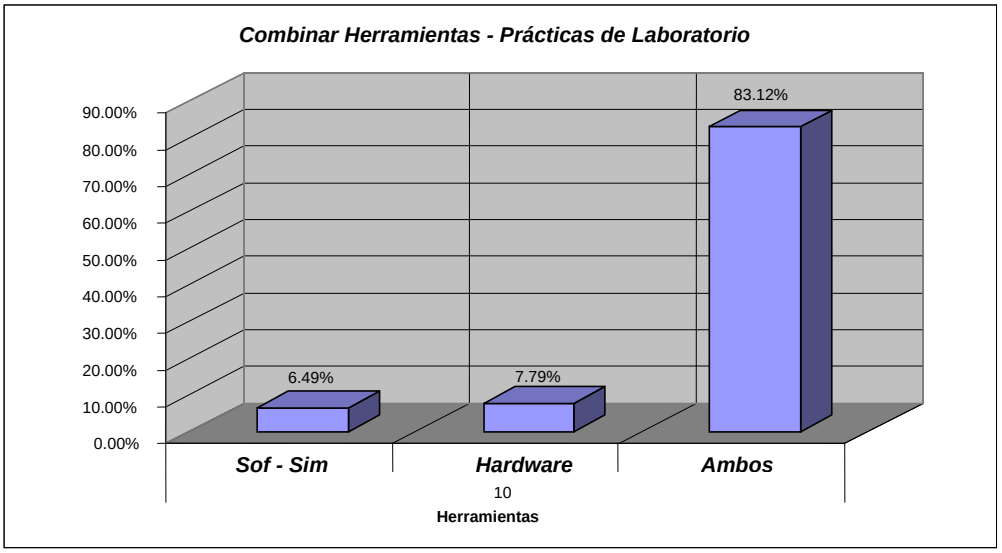
9.- ¿Considera que las practicas de laboratorio de las asignaturas previas de Microprogramación, le ayudaron reforzar los conocimientos básicos de Electrónica Digital?




| Mucho | Regular | Poco | Nada |
|---|---|---|---|
| **7.79%** | **31.17%** | **41.56%** | **19.48%** |
| 6 | 24 | 32 | 15 |

La grafica presenta un punto intermedio de confianza de los encuestados que no desestiman las prácticas previas a microprogramación, pero no poseen una convicción de que puedan dar un apoyo real y sólido al momento de la toma de la asignatura de microprogramación.

10.- ¿Especifique su preferencia en la utilización de las herramientas de apoyo y trabajo en las prácticas de laboratorio de Microprogramación?




| Software–Simuladores | Hardware | ambos |
|---|---|---|
| **6.49%** | **7.79%** | **83.12%** |
| 5 | 6 | 64 |

La tendencia es positiva en la utilización de una combinación de Software y Hardware para las Prácticas de Laboratorio, para una comprensión mas efectiva y dinámica de los problemas que se exponen en el aula y en los laboratorios, creando una visión mas amplia que permite abarcar muchos mas puntos de conocimiento.

## 3.4 INTERPRETACIÓN DE LOS DATOS DE LA ENCUESTA REALIZADA

La recopilación de datos, el análisis y el procesamiento de los mismos para luego realizar la interpretación de las opiniones y necesidades que presentan o que presentaron en su momento los estudiantes de diferentes niveles de la Carrera de Ingeniería en Sistemas para cursar las materias relacionadas a la electrónica digital y hardware de los equipos informáticos y que éstos datos sean significativos ya que demuestran la necesidad de la implementación de laboratorios prácticos. Los cuales tienen como propósito encaminar que sea el alumno el protagonista de sus aprendizajes y a la vez el mejoramiento de la calidad de la enseñanza académica por parte de la Facultad. Debiéndose si es necesario realizarse algunos ajustes en la misma para mejorar el perfil de sus graduados y conseguir una adecuada utilización de los recursos humanos y físicos existentes.

A continuación se presentan algunas de las razones más importantes por las cuales los estudiantes enfrentan problemas al momento de cursar las asignaturas que involucran a la electrónica digital y el hardware de equipo informático son las siguientes:

- La incomprensión de la información técnica.
- La falta de prácticas de laboratorio con equipos reales.
- La falta de ciertos conocimientos previos de electrónica digital y hardware de los equipos informáticos o al menos interés por aprender.

- La falta de habilidades técnicas en el uso de dispositivos electrónicos y equipo de utilizado en los laboratorio prácticos.
- La no existencia de un entorno de trabajo que permita la creación, desarrollo, depuración, pruebas y experimentación de sistemas electrónicos.
- Que los estudiantes no provienen de bachilleratos técnicos.

## 3.5 VISITAS TÉCNICAS REALIZADAS

El contar con las herramientas necesarias permite a las instituciones educativas poder proyectar hacia la comunidad estudiantil oportunidades nuevas de consolidar conocimientos nuevos que sean de un valor tanto general como agregado a su desempeño laboral como profesionales del campo de la Ingeniería de Sistemas Informáticos.

Con el objetivo de retomar ideas innovadoras, tener una visión más objetiva del trabajo en laboratorios de otras instituciones educativas y que puedan aportar un beneficio significativo al desarrollo de nuestra propuesta y trabajo de grado, se han realizado varias visitas de carácter técnico a universidades e instituciones técnicas que cuentan con laboratorios prácticos de microprogramación y con el equipo técnico necesario para poder realizar dichas prácticas de laboratorio. Se han visitado instituciones como la ***Universidad de El Salvador (Cuidad Universitaria)***, que cuentan con laboratorios y equipo básico para el trabajo con micro controladores, ***El Instituto Tecnológico Centro Americano (ITCA/FEPADE)*** que cuenta con laboratorio de microprogramación y equipos de entrenamiento. Además que están en relación directa con la carrera de Ingeniería de Sistemas Informáticos que sirve nuestra Facultad.

Las observaciones realizadas en estas visitas han sido resumidas en fichas de análisis, las cuales se muestran en el anexo. En este anexo se detallan aspectos tales como: datos generales de la institución visitada, condiciones de seguridad entre otros. Se muestran además una serie de fotografías en donde se pueden apreciar las instalaciones de los laboratorios.

## 3.6 CONCLUSIONES DE LA INVESTIGACIÓN TÉCNICA.

Luego de haber llevado acabo la realización de la investigación técnica en otras instituciones educativas destacamos los siguientes puntos:

Uno de los punto más destacados es el de mencionar que respecto a los modelos de micro controladores utilizados por las instituciones educativas para realizar las practicas de laboratorios hacen uso casi exclusivo del **PIC16F84**, debido a que es un sistema sencillo, barato y potente para muchas aplicaciones de electrónicas. Facilitando el proceso de enseñanza – aprendizaje esto es posible gracias su facilidad de programación (esto es porque solo posee 32 instrucciones de programación y además solo cuenta con dos puertos de comunicación, etc.).

Respecto a las herramientas utilizadas para la creación de proyectos mediante PICs hacen uso de la herramienta proporcionada por la compañía fabricante de PICs cuyo nombre es **MPLAB** ya que en esta aplicación posee la ventaja que proporciona un ambiente integral de desarrollo ya que incluye todas las herramientas para el desarrollo de la solución, es decir estas incluyen editor, compilador, simulación y grabación de la solución además de ser gratuita y estar disponible en todo momento en la Internet. Liberando al usuario de tener que adquirir cada herramienta de desarrollo por separado.

Es de mencionar que en la Universidad de El Salvador (Cuidad universitaria) existen dos asignaturas donde se desarrollan temas de estudios fundamentales acerca de Micro controladores estas asignaturas son: SISTEMAS DIGITALES 2 la cual es impartida para los estudiantes de ingeniería Eléctrica utilizando para realizar las practicas de laboratorio dos modelos de micro controladores los cuales son el PIC16F84 y el PIC16F877 además hacen uso de un entrenador (hardware) para el aprendizaje del micro controlador PIC16F877.

Mientras que la otra asignatura es MICROPROGRAMACIÓN la cual es impartida para los estudiantes de la carrera de Ingeniería en Sistemas en la cual hacen uso exclusivo de herramientas de software de simulación para la experimentación para realizar las prácticas de laboratorio de micro controladores.

De la presente investigación se desprenden una serie de conclusiones relevantes para entender los diversos panoramas encontrados en las distintas instituciones educativas, las cuales nos proporcionan una serie de perspectivas y parámetros de cómo se encuentra la enseñanza del tema, investigación y uso de las herramientas utilizadas para el proceso de aprendizaje de los sistemas electrónicos de control industrial, o no, en las que se opta por el micro controlador como elemento básico entorno al cual se implementa el sistema.

Lo cual nos ayuda a determinar la mejor solución a utilizar para la implantación de las prácticas de laboratorio con micro controladores en la FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE.

Debido a que existen muchos fabricantes de micro controladores, así como diversas arquitecturas e implementaciones de ellos proponemos la utilización del **PIC16F84** debido a que es un micro controlador que ha estado en el mercado hace mucho tiempo, y es lo bastante simple como para entender de manera general el funcionamiento de un micro controlador. Al mismo tiempo que facilita el proceso de enseñanza y aprendizaje para el alumno logrando así la proporcionarle conocimientos sólidos acerca del funcionamiento de los microprocesadores.

La incorporación de herramientas de software y hardware al proceso educativo constituye una práctica diaria e imprescindible en nuestra universidad, que se empeñan en aumentar la calidad de la enseñanza y el aprendizaje mediante procesos activos. También por que existen herramientas tanto hardware como software que permiten programarlos, depurarlos, emularlos, con la finalidad de extender, mejorar y hacer más eficiente la forma de enseñanza. ***Proponemos el uso de un sistema de desarrollo (Entorno de desarrollo y kits de entrenamiento).***

## 3.6.1 SISTEMA DE DESARROLLO (HW/SW)

El sistema de desarrollo de aplicaciones con micro controladores está formado por:

Kits de entrenamiento electrónico fundamental basado en un micro controlador PIC16F84 con sus puertos disponibles para la conexión de dispositivos periféricos además de un conjunto de dispositivos electrónicos(diodo leds, servomotores, display de 7 segmentos, LCD, etc.) para poder realizar las guías de practicas de laboratorio propuestas.

Un entorno de desarrollo **MPLAB de MICROCHIP** que es un conjunto de programas (software) que permiten realizar todas las tareas necesarias para desarrollar el programa de una aplicación. Las funciones típicas del programa de desarrollo, que se ejecuta en el computador, son:

- Edición del programa de la aplicación.
- Simulación de su comportamiento.
- Compilación.
- Programación del micro controlador, que en este caso se realiza a través de otro software.

# Capitulo 4

Propuesta de solución.

## *4.1. PROPUESTA DEL PLAN DE GUÍAS DE PRÁCTICAS*

### *4.1.1. INTRODUCCIÓN.*

El desarrollo del presente capítulo presenta la primera etapa de la propuesta de solución al diseño de guías de práctica de laboratorio con micro controladores en el área de hardware de la carrera de Ingeniería de Sistemas Informáticos,

Partiendo de los resultados obtenidos en la fase de investigación del proyecto.
El llevar los conocimientos teóricos expuestos en clase a la práctica constituye una poderosa herramienta para que el estudiante asimile los conocimientos y adquiera habilidades técnicas.

Es en este marco, que las guías de práctica están estructuradas acorde al contenido contemplado en el programa de estudio de la asignatura de Microprogramación de la carrera de Ingeniería de Sistemas Informáticos y las sugerencias y ayuda del docente de la asignatura, ocupan un papel primordial en el logro de los objetivos de éstas.

En este capítulo se presenta la propuesta de las guías de práctica, se Presenta el Plan de Guías de Práctica de micro controladores de la asignatura de Microprogramación que se obtuvieron en el capítulo tres por medio del análisis de la Malla Curricular y de la investigación de campo.

Las guías están diseñadas de una forma clara y explícita, de tal forma que al ser utilizadas por el estudiante, éste pueda desarrollarlas sin mayor complicación y se alcancen los objetivos de aprendizaje planteados tanto en cada práctica como los objetivos globales de la asignatura

### *4.1.2. ANÁLISIS DEL TEMARIO Y ACTIVIDADES.*

La asignatura tiene como uno de sus principales objetivos el proporcionar al alumno conocimientos sólidos acerca del funcionamiento de los micro controladores y sus circuitos periféricos. Aprender técnicas de programación en lenguaje ensamblador. Practicas sobre kits de desarrollo del micro controlador, manejando leds, teclados, displays etc. y como éstos, se integran con otros dispositivos periféricos para dar lugar a la disciplina de los sistemas micro computadores. Un objetivo básico de la asignatura es capacitar al alumno para entender los diversos aspectos que implican esta disciplina y afrontar el desarrollo de aplicaciones prácticas, industriales o no, en las que se opta por el micro controlador como elemento básico entorno al cual se implementa el sistema.

De cara a la actualización tecnológica y adecuación de nuestra asignatura “Microprogramación”, dentro del actual plan de estudios de la Ingeniería en Sistemas Informáticos, hemos expuesto algunas consideraciones referentes al trabajo personal y autónomo del estudiante, de manera que se favorezca un trabajo práctico activo y continuo.

Por un lado, hemos valorado y seleccionado diversas herramientas, tanto software como hardware, para el desarrollo de trabajos y proyectos de asignatura, con los requisitos de que sean gratuitas o de coste muy reducido, de manera que el estudiante pueda descargarlas (software), adquirirlas o montarlas él mismo (hardware), para que cada estudiante o grupo de ellos pueda disponer de sus propias herramientas libremente y trabajar a su ritmo y fuera de las limitaciones de los laboratorios universitarios.

Obsérvese que hemos decidido especialmente en poder a disposición herramientas hardware que permitan el trabajo con circuitería real, pues pensamos que la simulación no es suficiente, y que el alumno debe enfrentarse a la realidad de la conexión del micro controlador con periféricos, con sus características circuitales, temporales, etc.

Además, hemos comprobado a lo largo de esta investigación lo estimulante y motivante que resulta para los estudiantes el sentirse capaces de montar y controlar dispositivos reales A modo de ejemplo, no es lo mismo simular un programa que desplaza unos y ceros en la pantalla de un ordenador, que el mismo programa llevado a un hardware real con el resultado de conseguir realizar juegos de luces sobre diodos LED. Tampoco es lo mismo simular un programa que copia códigos binarios, que el mismo programa cargado en la memoria de un micro controlador y visualizando sus efectos (representación de dígitos o mensajes de texto) en pantallas de siete segmentos o de cristal líquido.

La motivación que las prácticas hardware despiertan en los estudiantes puede ser comprobada a través de los resultados en el laboratorio y habilidades que se destaquen en el uso del hardware y su conexión con la realidad.

### *4.1.2.1. TEMARIO DE LA ASIGNATURA:*

| UNIDAD | CONTENIDOS |
|---|---|
| UNIDAD I. INTRODUCCIÓN AL LENGUAJE ENSAMBLADOR | 1.1 Conceptos generales.<br>1.2 Ventajas y desventajas<br>1.3 Representación de datos. |
| UNIDAD II. ARQUITECTURA DEL PC | 2.1 Componentes básicos del procesador.<br>2.2 Ciclo de Fetch – Execure<br>2.3 Interrupciones<br>2.4 Características de la familia 80X86<br>2.5 Organización de la memoria. |
| UNIDAD III. LENGUAJE ENSAMBLADOR. | 3.1 Instrucciones básicas.<br>3.2 Acceso a registros y memoria.<br>3.3 Aritméticas y lógicas<br>3.4 Manipulación de bits.<br>3.5 Control de secuencia.<br>3.6 Modos de direccionamiento.<br>3.7 Entrada y salidas básicas.<br>3.8 herramientas. |
| UNIDAD IV. TÈCNICAS Y LÓGICA DE PROGRAMACIÓN. | 4.1 Estructura de programación<br>4.2 Subrutinas<br>4.3 Arreglos.<br>4.4 Ciclos.<br>4.5 Macros |
| UNIDAD 5. INTERRUPCIONES | 5.1 Manejo de archivos<br>5.2 Crear Archivos<br>5.3 Abrir archivo<br>5.4 Leer archivo<br>5.5 Escribir en el archivo<br>5.6 Posicionarse en el archivo<br>5.7 Cerrar el archivo<br>5.8 Borrar el archivo<br>5.9 Códigos de error |

### *4.1.3. PLAN DE GUÍAS DE PRÁCTICA*

En la asignatura de Microprogramación el estudiante tiene una introducción al lenguaje ensamblador, estudia la arquitectura del PC, desarrolla instrucciones en lenguaje ensamblador donde aplica técnicas y lógica de programación.

| RELACIÓN CON EL PLAN DE ESTUDIO | Nº DE GUÍA | NOMBRE DE LA GUÍA |
|---|---|---|
| **Instrucciones básicas**<br><br>Están agrupadas, desde el punto de vista del programador, instrucciones que operan sobre registros. En primer lugar se agrupan las instrucciones que operan con bytes y que involucran algún registro de la memoria interna. En segundo lugar se analizaran las instrucciones que operan solo sobre el registro W y que permiten cargarle una constante implícita o incluida literalmente en la instrucción (literales) Ej.<br><br>**MOVF f,d  CLRF f** | 2 | ENCENDER UN DIODO LED EN UNA SALIDA |
| | 3 | ENCENDER UN LED EN UNA SALIDA ATREVES DE UN PULSADOR EN UNA ENTRADA. |
| | 4 | ENCENDER CUATRO LEDS EN FORMA SECUENCIAL |
| | 5 | TRABAJANDO DIRECTAMENTE CON UN DISPLAY DE 7 SEGMENTOS |
| | 6 | TRABAJANDO DIRECTAMENTE CON UN DISPLAY (CONTADOR 0 - F) |
| | 7 | MANEJO DE INTERRUPCIONES |

| | | |
|---|---|---|
| **Acceso a registros y memoria**<br><br>La memoria interna se direcciona en ***forma directa*** por medio de los 5 bits "f" contenidos en las instrucciones que operan sobre registros. De esta manera se puede direccionar cualquier posición desde la 00 a la 1F. Correspondiente a los mapas de memoria, las direcciones 10 a 1F corresponden a los bancos de registros. El programador deberá asegurarse de haber programado los bits de selección de banco en el registro FSR.<br><br>El registro FSR, además de servir para seleccionar el banco activo, sirve como puntero para ***direccionamiento indirecto***. La posición 00 del mapa de RAM es la llamada dirección indirecta. Sí en cualquier instrucción se opera con la dirección 00, en realidad se estará operando con la dirección a donde apunte el contenido del FSR. | 2<br><br>3<br><br>4<br><br>5<br><br>6<br><br>7 | ENCENDER UN DIODO LED EN UNA SALIDA<br><br>ENCENDER UN LED EN UNA SALIDA A TRAVÉS DE UN PULSADOR EN UNA ENTRADA.<br><br>ENCENDER CUATRO LEDS EN FORMA SECUENCIAL<br><br>TRABAJANDO DIRECTAMENTE CON UN DISPLAY DE 7 SEGMENTOS<br><br>TRABAJANDO DIRECTAMENTE CON UN DISPLAY ( CONTADOR 0 - F)<br><br>MANEJO DE INTERRUPCIONES |
| **Aritméticas y lógicas**<br><br>operaciones que realizan la suma, la resta aritméticas y comparaciones lógicas<br><br>**ADDWF f,d SUBWF f,d**<br>**ANDWF f,d IORWF f,d** | 5<br><br>6<br><br>7 | TRABAJANDO DIRECTAMENTE CON UN DISPLAY DE 7 SEGMENTOS<br><br>TRABAJANDO DIRECTAMENTE CON UN DISPLAY (CONTADOR 0 - 9)<br><br>MANEJO DE INTERRUPCIONES |

| | | |
|---|---|---|
| **Interrupciones**<br><br>Utilización del registro **INTCON (Registro Controlador Interrupciones)** para especificar la fuente de la interrupción producida.<br><br>**ISR (Rutina de Servicio de Interrupción)** porción de código utilizada para atender a la interrupción producida.<br><br>la instrucción **RETFIE** para retorno de una interrupción y activar a **GIE** | 5<br><br>6<br><br>7 | TRABAJANDO DIRECTAMENTE CON UN DISPLAY DE 7 SEGMENTOS<br><br>TRABAJANDO DIRECTAMENTE CON UN DISPLAY (CONTADOR 0 - 9)<br><br>MANEJO DE INTERRUPCIONES |
| **Herramientas.**<br><br>Se utilizan para realizar las funciones (edición, simulación, compilación y grabación de los programas en los micro controladores) en este caso se utiliza<br><br>**MPLAB, WINPIC y PROTEUS** | 1 | FAMILIARIZACIÓN CON EL AMBIENTE DE DESARROLLO INTEGRAL DE MPLAB Y WINPIC |

## *4.2. PROPUESTA DE EQUIPAMIENTO PARA LOS LABORATORIOS*

### *4.2.1. INFORMACIÓN DEL MICRO CONTROLADOR PIC16F84.*

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

El PIC 16C84 es un microcontrolador de la familia MICROCHIP, totalmente compatible con el PIC 16F84. Su principal característica es que posee memoria "EEPROM" en lugar de memoria Flash, pero su manejo es igual. Con respecto al PIC16F84, este microcontrolador presenta dos diferencias:

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

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

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

#### *4.2.1.1. MEMORIA DE PROGRAMA*

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

#### *4.2.1.2. VECTOR DE RESET*

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

#### *4.2.1.3. VECTOR DE INTERRUPCIÓN*

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

#### *4.2.1.4. REGISTROS (MEMORIA RAM)*

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

#### *4.2.1.5. PINES Y FUNCIONES*

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

El PIC 16F84 tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8 líneas. Cada pin se puede configurar como entrada o como salida

independiente programado por un par de registros diseñados para tal fin. En ese registro un "0" configura el pin del puerto correspondiente como salida y un "1" lo configura como entrada.

- PUERTO A

RA0 = Pin de Entrada/Salida (TTL).

RA1 = Pin de Entrada/Salida (TTL).

RA2 = Pin de Entrada/Salida (TTL).

RA3 = Pin de Entrada/Salida (TTL).

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

- PUERTO B

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

RB1 = Pin de Entrada/Salida (TTL).

RB2 = Pin de Entrada/Salida (TTL).

RB3 = Pin de Entrada/Salida (TTL).

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

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

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

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

- PINES ADICIONALES

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

Vss = Ground o Tierra

VDD = Fuente Positiva (+5V)

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

OSC1/CLKIN = Entrada del Oscilador del Cristal / Entrada de reloj de una Fuente Externa.

El Puerto B tiene Internamente unas resistencias de pull-up conectadas a sus pines (sirven para fijar el pin a un nivel de cinco voltios), su uso puede ser habilitado o deshabilitado bajo control del programa. Todas las resistencias de pull-up conectan o desconectan a la vez. La resistencia de pull-up es desconectada automáticamente en un pin si este se programa como salida. El pin RB0/INT se puede configurar por software para que funcione como interrupción externa.

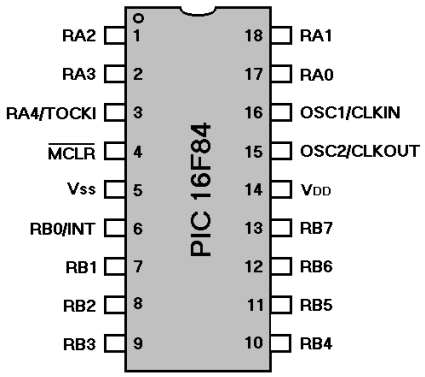
El pin RA4/TOCKI del puerto A puede ser configurado como un pin de entrada/salida como se mencionaba anteriormente o como entrada del temporizador/contador. Cuando este pin se programa como entrada digital, funciona como un disparador de Schmitt (Schmitt trigger, ST), esto quiere decir que puede reconocer señales un poco distorsionadas y llevarlas a niveles lógicos (cero y cinco voltios). Cuando se usa como salida digital se comporta como colector abierto, por lo tanto se debe poner una resistencia de pull-up (resistencia externa conectada a un nivel lógico de cinco voltios). Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega en el pin un "1"

lógico. Además como salida no puede manejar cargas como fuente, sólo en el modo sumidero.

Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectado a alguna parte, nunca dejarlos al aire por que se puede dañar el integrado. Los pines que no se estén usando se deben conectar la fuente de alimentación +5V con una resistencia de < 5 Kilo Ohmio.

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

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




**PINES Y FUNCIONES**

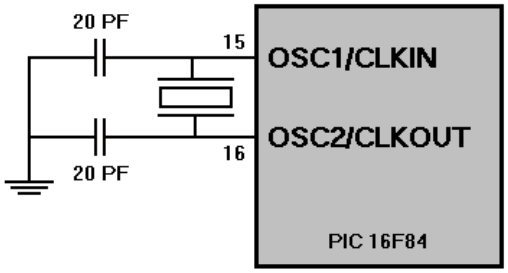
### *4.2.1.6. EL OSCILADOR EXTERNO*

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

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

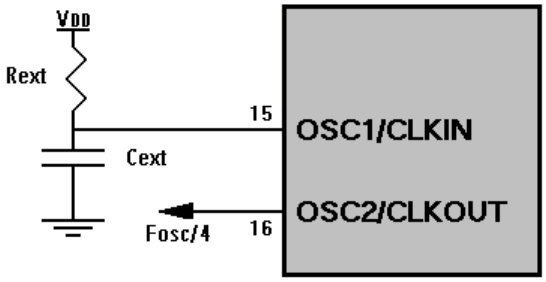
En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados "fusibles de configuración".

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




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

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




Los valores recomendados para este tipo de oscilador son: 5 KW £ Rext £ 100 KW y Cext > 20 pF.

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

#### *4.2.1.7. RESET*

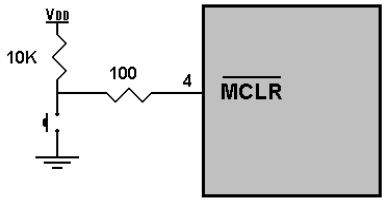
En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando sea necesario, ya sea por una falla que se presente o por que así se halla diseñado el sistema. El pin de reset en los PIC es llamado "Master Clear". El PIC 16F84 admite diferentes tipos de reset:

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

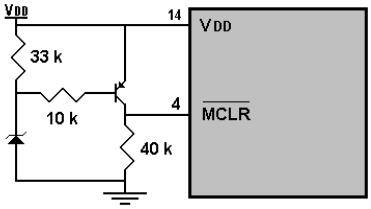
El reset al encendido se consigue gracias a dos temporizadores. El primero de ellos es el OST (Oscillator Star-Up Timer: Temporizador de encendido del oscilador), orientado a mantener el microcontrolador en reset hasta que el oscilador de cristal es estable. El segundo es el PWRT (Power-Up Timer: Temporizador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

El reset por Master Clear se consigue llevando momentáneamente este pin a un estado lógico bajo, mientras que el watchdog WDT produce un reset cuando su temporizador rebasa la cuenta, o sea que pasa de 0FFh a 00H.

Cuando se quiere tener control sobre el reset del sistema se puede conectar un botón como se muestra en la siguiente figura.




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




Circuito de Protección # 1.

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

Circuito de Protección # 2.

Este circuito es más económico, aunque menos eficaz. El transistor Q1 pasará a un estado de corte cuando VDD está por debajo de un cierto nivel tal que:VDD * (R1 / (R1 + R2)) = 0.7 V

### *4.2.1.8. REGISTROS (MEMORIA RAM)*

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

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

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

***02h o PCL:*** CONTADOR DE PROGRAMA. Se utiliza para direccionar las palabras de 14 bits del programa del usuario que se encuentra almacenado en la memoria ROM; este contador tiene un tamaño de 13 bits. Sobre el byte bajo, se puede escribir o leer a voluntad directamente, mientras que en el byte alto, no. El byte alto se maneja mediante el registro PCLATH (0Ah). A diferencia de los PIC de primera generación el 16F84 ante una condición de reset inicia el contador de programa con todos sus bits en "cero". Durante la ejecución normal del programa, y dado que todas las instrucciones ocupan solo una posición de memoria, el contador se incrementa con cada instrucción, a menos que se trate de alguna instrucción de salto.

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

| IRP<br>bit 7 | RP1<br>bit 6 | RP0<br>bit 5 | T0<br>bit 4 | PD<br>bit 3 | Z<br>bit 2 | DC<br>bit 1 | C<br>bit 0 |
|---|---|---|---|---|---|---|---|
| IRP | Selector de página para direccionamiento indirecto. Este bit no se utiliza efectivamente en el PIC 16F84, por lo que se puede utilizar como un bit de propósito general. | | | | | | |

| | |
|---|---|
| RP1,0 | Selectores de página para un seleccionamiento directo. Solamente RP0 se utiliza en el PIC 16F84. RP1 se puede utilizar como un bit de propósito general. |
| T0 | Time Out o bit de finalización del temporizador. Se coloca en 0 cuando el circuito de vigilancia Watchdog finaliza la temporización |
| PD | Power Down o bit de bajo consumo. Se coloca en 0 por la instrucción sleep. |
| Z | Zero o bit de cero. Se coloca en 1 cuando el resultado de una operación aritmética o lógica es cero. |
| DC | Digit Carry o bit de acarreo de dígito. En operaciones aritméticas se activa cuando hay un acarreo entre el bit 3 y 4, es decir cuando hay acarreo entre el nible de menor y de mayor peso. |
| C | Carry o bit de acarreo. En instrucciones aritméticas se activa cuando se presenta acarreo desde el bit más significativo del resultado. |

***04h o FSR: REGISTRO SELECTOR DE REGISTROS.*** En asocio con el registro IND0, se utiliza para seleccionar indirectamente los otros registros disponibles. Mientras que los antecesores del PIC 16F84 solo poseían 5 bits activos, en este microcontrolador se poseen solo 8 bits. Si en el programa no se utilizan llamadas indirectas, este registro se puede utilizar como un registro de propósito general.

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

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

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

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

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

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

| GIE<br>Bit 7 | EEIE<br>bit 6 | TOIE<br>bit 5 | INTE<br>bit 4 | RBIE<br>bit 3 | TOIF<br>bit 2 | INTF<br>bit 1 | RBIF<br>bit 0 |
|---|---|---|---|---|---|---|---|
| GIE | Global Interrup Enable o Habilitador general de interrupciones.<br>0: Deshabilita todas las interrupciones<br>1: Habilita las interrupciones | | | | | | |
| EEIE | EEPROM Write Interrup Enable o Habilitación de interrupción por escritura de la EEPROM.<br>0: La deshabilita<br>1: La habilita | | | | | | |
| T0IE | TMR0 Interrup Enable o Habilitación de interrupción del temporizador TMR0.<br>0: La deshabilita<br>1: La habilita | | | | | | |
| INTE | INT Interrup Enable o Habilitación de la interrupción INT.<br>0: La deshabilita<br>1: La habilita | | | | | | |
| RBIE | RBIF Interrup Enable o Habilitación de la interrupción RBIF.<br>0: La deshabilita<br>1: La habilita | | | | | | |
| TAIF | TMR0 Overflow Interrup Flag o Bandera de la interrupción por desbordamiento del TMR0.<br>Se coloca en 1 cuando el TMR0 pasa de 0FFh a 00h; ésta debe ser puesta a 0 por programa. | | | | | | |
| INTF | INT Interrup Flag o Bandera de interrupción INT.<br>Se coloca en 1 cuando la interrupción INT ocurre; ésta debe ser puesta cero por programa. | | | | | | |
| RBIF | RB Port Change Interrup Flag o Bandera de interrupción por cambio en el puerto B.<br>Se coloca en 1 cuando una de las entradas (RB4 a RB7) cambia; ésta debe ser puesta a 0 por programa | | | | | | |

***81h u OPTION: REGISTRO DE CONFIGURACION MULTIPLE.*** Posee varios bits para configurar el preescalador, la interrupción externa, el timer y las

características del Puerto B. Los bits que contiene y las funciones que realiza este registro se muestran en la figura. El preescalador es compartido entre el TMR0 y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.

<table>
<tr><td>RBPU<br>bit 7</td><td>INTEDG<br>bit 6</td><td>GRTS<br>bit 5</td><td>RTE<br>bit 4</td><td>PSA<br>bit 3</td><td>PS2<br>bit 2</td><td>PS1<br>bit 1</td><td>PS0<br>bit 0</td></tr>
<tr><td>RBPU</td><td colspan="7">Portb Pull-up Enable o Habilitación de pull-up del puerto B.<br>0: Habilita las pull-ups internas<br>1: Las deshabilita</td></tr>
<tr><td>INTEDG</td><td colspan="7">INT Interrup Edge Select o Selector de flanco de la interrupción INT.<br>0: Flanco de bajada<br>1: Flanco de subida</td></tr>
<tr><td>RTS</td><td colspan="7">TMR0 Signal Source o Fuente de señal del TMR0.<br>0: Ciclo de instrucciones interno (Temporizador)<br>1: Transición en el pin RA4/TOCKI (Contador)</td></tr>
<tr><td>RTE</td><td colspan="7">TMR0 Signal Edge o Flanco de la señal del TMR0<br>0: Incremento de transición de bajo a alto<br>1: Incremento en transición</td></tr>
<tr><td>PSA</td><td colspan="7">Preescaler Assignment o Asignación del preescalador<br>0: TMR0 (Contador / Temporizador)<br>1: WDT (Circuito de vigilancia)</td></tr>
<tr><td rowspan="10">PS2, 1, 0</td><td colspan="7">Preescaler Value o Valores del preescalador.</td></tr>
<tr><td>Valor</td><td colspan="3">TMR0</td><td colspan="3">WDT</td></tr>
<tr><td>000</td><td colspan="3">1:2</td><td colspan="3">1:1</td></tr>
<tr><td>001</td><td colspan="3">1:4</td><td colspan="3">1:2</td></tr>
<tr><td>010</td><td colspan="3">1:8</td><td colspan="3">1:4</td></tr>
<tr><td>011</td><td colspan="3">1:16</td><td colspan="3">1:8</td></tr>
<tr><td>100</td><td colspan="3">1:32</td><td colspan="3">1:16</td></tr>
<tr><td>101</td><td colspan="3">1:64</td><td colspan="3">1:32</td></tr>
<tr><td>110</td><td colspan="3">1:128</td><td colspan="3">1:64</td></tr>
<tr><td>111</td><td colspan="3">1:256</td><td colspan="3">1:128</td></tr>
</table>

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

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

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

| U<br>bit 7 | U<br>bit 6 | U<br>bit 5 | EEIF<br>bit 4 | WRERR<br>bit 3 | WREN<br>bit 2 | WR<br>bit 1 | RD<br>bit 0 |
|---|---|---|---|---|---|---|---|
| U | Unimplemented. No implementados. | | | | | | |
| EEIF | EEPROM Write Completion Interrup Flag o Bandera de finalización de la escritura. Se coloca en "1" cuando finaliza con éxito la escritura de la EEPROM de datos; se debe colocar en "0" por programa. El bit de habilitación correspondiente es el EEIE, localizado en el registro INTCON. | | | | | | |
| WRERR | Write Error Flag o Bandera de error de escritura. Si se coloca en "1" cuando la operación de escritura termina prematuramente, debido a cualquier condición de reset. | | | | | | |
| WREN | Write Enable o habilitación de escritura. Si se coloca en "0" no permite las operaciones de escritura; en "1" las habilita. | | | | | | |

| | |
|---|---|
| WR | Write Control o Control de escritura. Al colocarse en "1" inicia un ciclo de escritura. Este bit sólo es puesto a "0" por hardware, una vez la escritura termina. |
| RD | Read Control o Control de lectura. Al colocarse en "1" se inicia una lectura de la EEPROM de datos, la cual toma un ciclo de reloj de instrucciones. Este bit sólo se limpia (colocar en "0") por hardware, al finalizar la lectura de la posición de la EEPROM. |

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

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

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

***PILA (STACK).*** Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario. Se usan para guardar el

valor del contador de programa cuando se hace un llamado a una subrutina (CALL ), o cuando se atiende una interrupción; luego, cuando el micro regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila. El PIC 16F84 tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problema alguno.

#### *4.2.1.9. CARACTERÍSTICAS ESPECIALES*

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

##### *4.2.1.9.1. CIRCUITO DE VIGILANCIA (WATCHDOG TIMER)*

Su función es restablecer el programa cuando éste se ha perdido por fallas en la programación o por alguna razón externa. Es muy útil cuando se trabaja en ambientes con mucha interferencia o ruido electromagnético. Esta conformado por un oscilador RC que se encuentra dentro del microprocesador. Este oscilador corre de manera independiente al oscilador principal. Cuando se habilita su funcionamiento, dicho circuito hace que el micro controlador sufra un reset cada determinado tiempo (que se puede programar entre 18 ms y 2 segundos). Este reset lo puede evitar el usuario mediante una instrucción especial del micro controlador (CLRWT: Borra el contenido del Watchdog), la cual se debe ejecutar antes de que termine el periodo nominal de dicho

temporizador. De esta manera si el programa se ha salido de su flujo normal, por algún ruido o interferencia externa, el sistema se reiniciará (cuando se acabe el tiempo programado y no se haya borrado el contador) y el programa puede restablecerse para continuar con su funcionamiento normal.

#### *4.2.1.9.2. TEMPORIZADOR DE ENCENDIDO (POWER-UP TIMER)*

Este proporciona un reset al micro controlador en el momento de conectar la fuente de alimentación, lo que garantiza un arranque correcto del sistema. En el momento de grabar el micro controlador se debe habilitar el fusible de configuración "Power-up Timer", para ello se debe seleccionar "ON". Su tiempo de retardo es de 72 milisegundos.

#### *4.2.1.9.3. MODO DE BAJO CONSUMO (SLEEP)*

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

#### *4.2.1.9.4. INTERRUPCIONES*

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

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

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

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

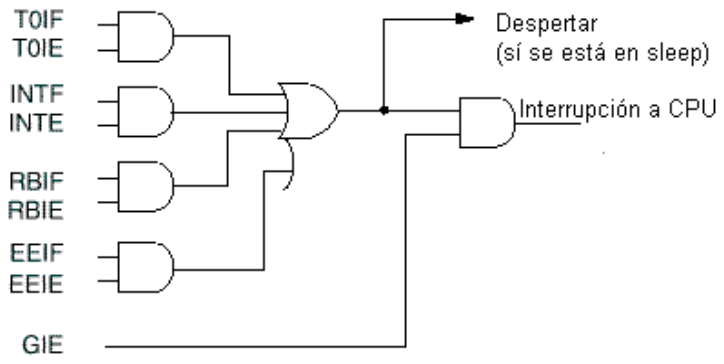
interrupción, a la vez que habilita de nuevo las interrupciones, al colocar el bit GIE en uno. Debe tenerse presente que solamente el contador de programa es puesto en la pila al atenderse la interrupción; por lo tanto, es conveniente que el programador tenga cuidado con el registro de estados y el de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

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

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

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

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




**LÓGICA DE INTERRUPCIONES PARA EL PIC 16FXX**

### *4.2.1.9.5. MEMORIA DE DATOS DE LA EEPROM*

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

El registro EEADR (posición 09), que debe contener la dirección de la posición de la EEPROM a ser accesada.

El registro EEDATA (posición 08), que contiene el dato de 8 bits que se va a escribir o el que se obtuvo de la última lectura.

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

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

#### *4.2.1.9.6. FUSIBLES DE CONFIGURACIÓN*

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

reconstruir. Adicionalmente, todas las posiciones de memoria del programa se protegen contra la reprogramación.

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

#### *4.2.1.9.7. LAS PULL-UP INTERNAS*

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

### *4.2.1.10. LA ARQUITECTURA*

#### *4.2.1.10.1. LA ARQUITECTURA TRADICIONAL*

La arquitectura tradicional de computadoras y microprocesadores se basa en el esquema propuesto por John Von Neumann, en el cual la unidad central de proceso, o CPU, esta conectada a una memoria única que contiene las instrucciones del programa y los datos (ver figura). El tamaño de la unidad de datos o instrucciones esta fijado por el ancho del bus de la memoria. Es

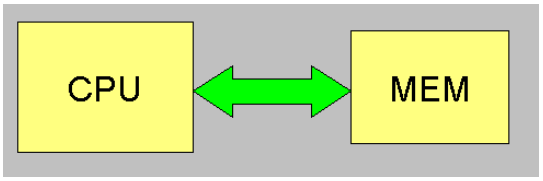
decir que un microprocesador de 8 bits, que tiene además un bus de 8 bits que lo conecta con la memoria, deberá manejar datos e instrucciones de una o más unidades de 8 bits (bytes) de longitud. Cuando deba acceder a una instrucción o dato de más de un byte de longitud, deberá realizar más de un acceso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

1. Que la longitud de las instrucciones esta limitada por la unidad de longitud de los datos, por lo tanto el microprocesador debe hacer varios accesos a memoria para buscar instrucciones complejas.

2. Que la velocidad de operación (o ancho de banda de operación) esta limitada por el efecto de cuello de botella que significa un bus único para datos e instrucciones que impide superponer ambos tiempos de acceso.

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

Arquitectura Von Neumann



### *4.2.1.10.2. LA ARQUITECTURA HARVARD Y SUS VENTAJAS*

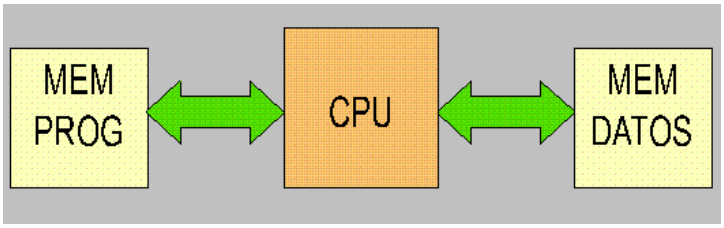
La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que el CPU esta conectado a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos. Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de la memoria de programa pueden diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar. Se puede observar claramente que las principales ventajas de esta arquitectura son:

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

1. Que el tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

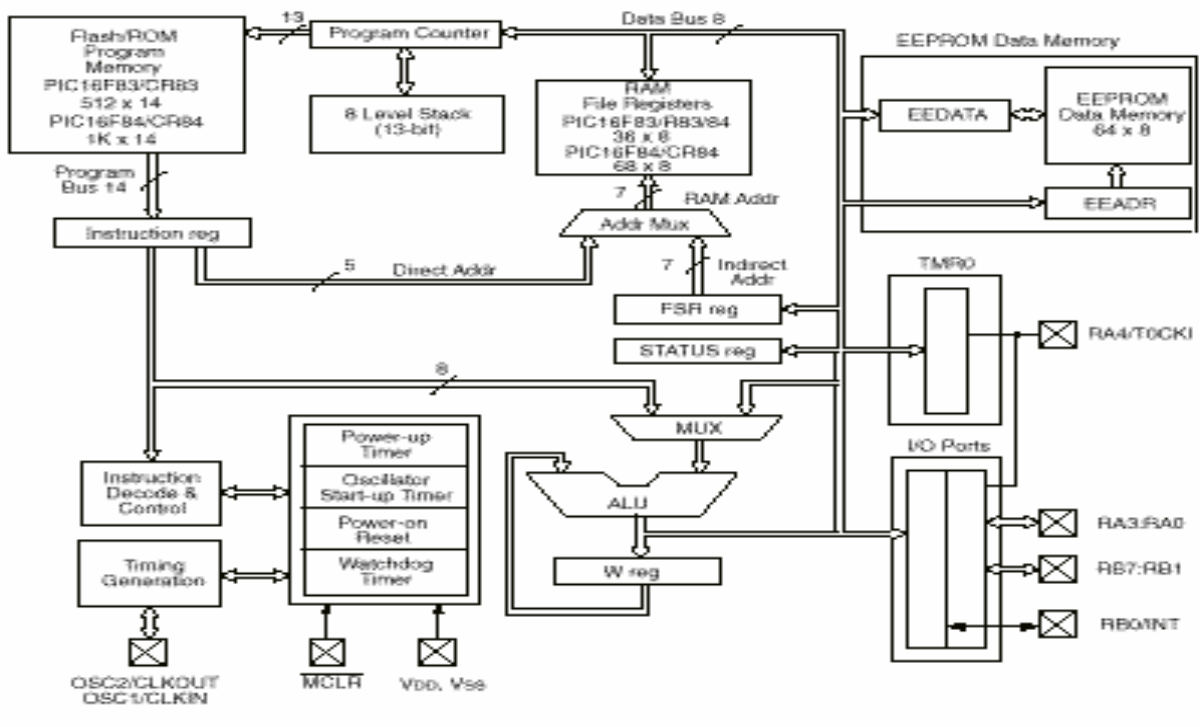
Una pequeña desventaja de los procesadores con arquitectura Harvard, es que deben poseer instrucciones especiales para acceder a tablas de valores constantes que pueda ser necesario incluir en los programas, ya que estas tablas se encontraran físicamente en la memoria de programa (por ejemplo en la EPROM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

Arquitectura Harvard




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

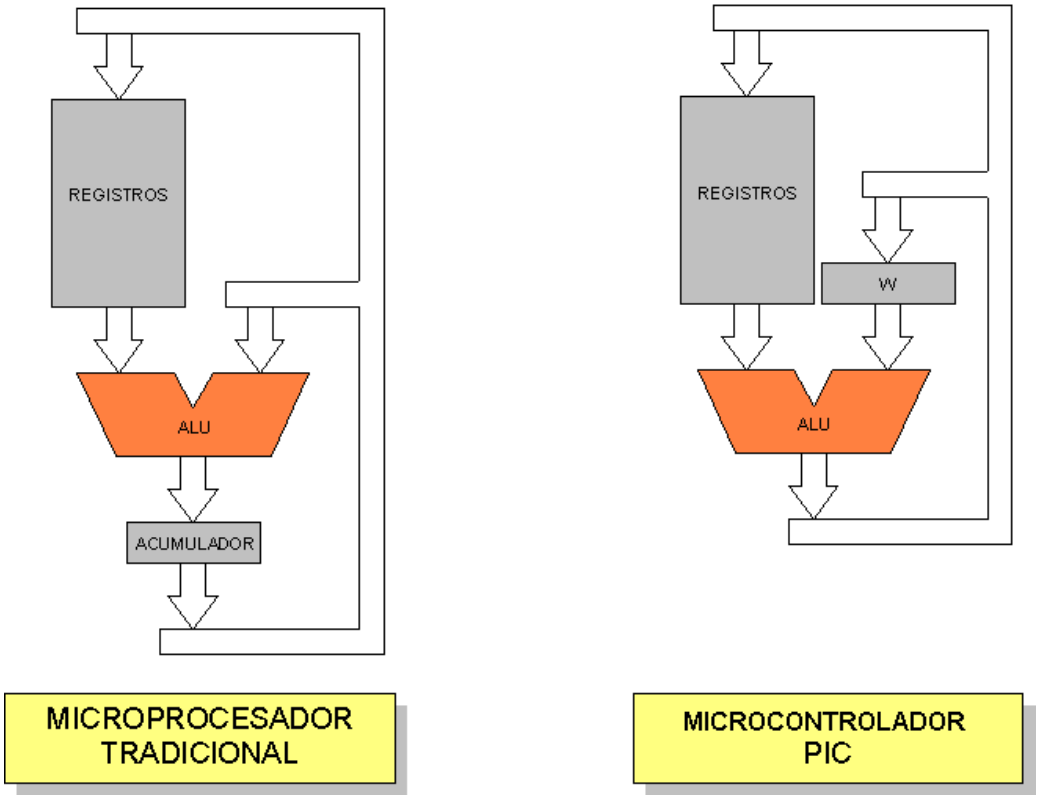
### *4.2.1.10.3. DIAGRAMA DE BLOQUE*




### *4.2.1.11. DIFERENCIA MICROPROCESADORES Y MICRO CONTROLADORES*

La figura siguiente se representa un diagrama simplificado de la arquitectura interna del camino de los datos en el CPU de los micro controladores PIC y los tradicionales microprocesadores. Este diagrama puede no representar con exactitud el circuito interno de estos micro controladores, pero es exacto y claro desde la óptica del programador. La figura representa el mismo diagrama para un microprocesador ficticio de arquitectura tradicional. Se puede observar que la principal diferencia entre ambos radica en la ubicación

del registro de trabajo, que para los PICs se denomina W (Working Register), y para los tradicionales es el Acumulador(A).




En los micro controladores tradicionales todas las operaciones se realizan sobre el acumulador. La salida del acumulador esta conectada a una de las entradas de la Unidad Aritmética y Lógica (ALU), y por lo tanto éste es siempre uno de los dos operandos de cualquier instrucción. Por convención, las instrucciones de simple operando (borrar, incrementar, decrementar, complementar), actúan sobre el acumulador. La salida de la ALU va solamente a la entrada del acumulador, por lo tanto el resultado de cualquier operación siempre quedara en este registro. Para operar sobre un dato de memoria, luego

realizar la operación siempre hay que mover el acumulador a la memoria con una instrucción adicional.

En los micro controladores PIC, la salida de la ALU va al registro W y también a la memoria de datos, por lo tanto el resultado puede guardarse en cualquiera de los dos destinos. En las instrucciones de doble operando, uno de los dos datos siempre debe estar en el registro W, como ocurría en el modelo tradicional con el acumulador. En las instrucciones de simple operando el dato en este caso se toma de la memoria (también por convención). La gran ventaja de esta arquitectura es que permite un gran ahorro de instrucciones ya que el resultado de cualquier instrucción que opere con la memoria, ya sea de simple o doble operando, puede dejarse en la misma posición de memoria o en el registro W, según se seleccione con un bit de la misma instrucción. Las operaciones con constantes provenientes de la memoria de programa (literales) se realizan solo sobre el registro W.

En la memoria de datos de los PICs se encuentran ubicados casi todos los registros de control del microprocesador y sus periféricos auto contenidos, y también las posiciones de memoria de usos generales. En el caso de los 16CXX, algunos registros especiales de solo escritura (TRIS y OPTION) no están accesibles dentro del bloque de memoria de datos, sino que solo se pueden cargar desde el registro W por medio de instrucciones especiales.

### *4.2.1.12. CONTADOR DE PROGRAMA*

Este registro, normalmente denominado PC, es totalmente equivalente al de todos los microprocesadores y contiene la dirección de la próxima instrucción a ejecutar. Se incrementa automáticamente al ejecutar cada instrucción, de manera que la secuencia natural de ejecución del programa es lineal, una instrucción después de la otra. Algunas instrucciones que llamaremos de control, cambian el contenido del PC alterando la secuencia lineal de ejecución. Dentro de estas instrucciones se encuentran el GOTO y el CALL que permiten cargar en forma directa un valor constante en el PC haciendo que el programa salte a cualquier posición de la memoria. Otras instrucciones de control son los SKIP o “salteos” condicionales, que producen un incremento adicional del PC si se cumple una condición específica, haciendo que el programa saltee, sin ejecutar, la instrucción siguiente.

Al resetearse el microprocesador, todos los bits del PC toman valor 1, de manera que la dirección de arranque del programa es siempre la última posición de memoria de programa. En esta posición se deberá poner una instrucción de salto al punto donde verdaderamente se inicia el programa.

A diferencia de la mayoría de los microprocesadores convencionales, el PC es también accesible al programador como registro de memoria interna de datos, en la posición de 02. Es decir que cualquier instrucción común que opere sobre registros puede ser utilizada para alterar el PC y desviar la ejecución del programa. El uso indiscriminado de este tipo de instrucciones complica el programa y puede ser muy peligroso, ya que puede producir comportamientos

difíciles de predecir. Sin embargo, algunas de estas instrucciones utilizadas con cierto método, pueden ser muy útiles para implementar poderosas estructuras de control tales como el goto computado. Como el microprocesador opera con datos de 8 bits, y la memoria de datos es también de 8 bits, estas instrucciones solo pueden leer o modificar los bits 0 a 7 del PC.

#### *4.2.1.13. STACK*

En los micro controladores PIC el stack es una memoria interna dedicada, de tamaño limitado, separada de las memorias de datos y de programa, inaccesible al programador, y organizada en forma de pila, que es utilizada solamente, y en forma automática, para guardar las direcciones de retorno de subrutinas e interrupciones. Cada posición es de 11 bits y permite guardar una copia completa del PC. Como en toda memoria tipo pila, los datos son accedidos de manera tal que el primero que entra es el ultimo que sale.

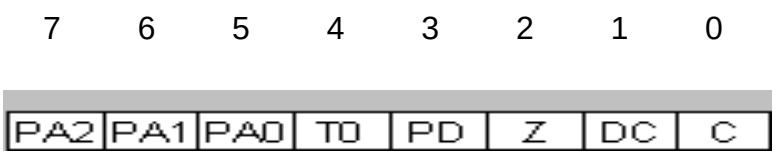
En los 16C5X el stack es de solo dos posiciones, mientras que en los 16XXX es de 8 posiciones y en los 17CXX es de 16 posiciones. Esto representa, en cierta medida, una limitación de estos micro controladores, ya que no permite hacer uso intensivo del anidamiento de subrutinas. En los 16C5X, solo se pueden anidar dos niveles de subrutinas, es decir que una subrutina que es llamada desde el programa principal, puede a su vez llamar a otra subrutina, pero esta ultima no puede llamar a una tercera, porque se desborda la capacidad del stack, que solo puede almacenar dos direcciones de retorno. Esto de hecho representa una traba para el programador y además parece impedir o dificultar la programación estructurada, sin embargo es una

buena solución de compromiso ya que estos micro controladores están diseñados para aplicaciones de alta velocidad en tiempo real, en las que el overhead (demoras adicionales) que ocasiona un excesivo anidamiento de subrutinas es inaceptable. Por otra parte existen técnicas de organización del programa que permiten mantener la claridad de la programación estructurada, sin necesidad de utilizar tantas subrutinas anidadas.

Como ya se menciono anteriormente, el stack y el puntero interno que lo direcciona, son invisibles para el programador, solo se los accede automáticamente para guardar o rescatar las direcciones de programa cuando se ejecutan las instrucciones de llamada o retorno de subrutinas, o cuando se produce una interrupción o se ejecuta una instrucción de retorno de ella.

### *4.2.1.14. PALABRA DE ESTADO DEL PROCESADOR*

La palabra de estado del procesador contiene los tres bits de estado de la ALU (C, DC y Z), y otros bits que por comodidad se incluyeron en este registro.



### *4.2.1.15. REGISTRÓ STATUS*

El bit Z indica que el resultado de la última operación fue CERO. El bit C indica acarreo del bit más significativo (bit 7) del resultado de la última operación de suma. En el caso de la resta se comporta a la inversa, C resulta 1 si no hubo pedido de préstamo. El bit DC (Digit Carry) indica acarreo del cuarto bit (bit 3) del resultado de la última operación de suma o resta, con un comportamiento análogo al del bit C, y es útil para operar en BCD (para sumar o restar números en código BCD empaquetado). El bit C es usado además en las operaciones de rotación derecha o izquierda como un paso intermedio entre el bit 0 y el bit 7.

El bit PD (POWER DOWN) sirve para detectar si la alimentación fue apagada y encendida nuevamente, tiene que ver con la secuencia de inicialización, el Watch Dog Timer y la instrucción sleep, y su uso se detallara en la sección referida al modo POWER DOWN. El bit TO (TIME-OUT) sirve para detectar si una condición de reset fue producida por el Watch Dog Timer, esta relacionado con los mismos elementos que el bit anterior y su uso se detallara en la sección referida al WATCH DOG TIMER. Los bits de selección de pagina PA0/PA1/PA2 se utilizan en las instrucciones de salto GOTO y CALL, y se explicaran con detalle en la sección referida a las instrucciones de control, y a la organización de la memoria de programa. En realidad en el 16C54 estos bits no se usan y sirven para propósitos generales. En el 16C57 el PA0 si se usa pero los otros dos no. En el 16C55 se utilizan PA0 y PA1. PA2 esta reservado para uso futuro y en cualquiera de los PIC 16C5X sirve para propósitos generales.

### *4.2.1.16. OTROS REGISTROS ESPECIALES*

Las ocho primeras posiciones del área de datos están reservadas para alojar registros de propósito especial, quedando las restantes libres para contener los datos u operandos que se desee (registros de propósito general).

El registro ***INDF*** que ocupa la posición 0 no está implementando físicamente y, como se ha explicado, se le referencia en el direccionamiento indirecto de datos aunque se utiliza el contenido de FSR.

En la dirección esta el registro ***TAR0*** (Temporizador) que puede ser leído y escrito como cualquier otro registró. Puede incrementar su valor con una señal externa aplicada al pin T0CKI o mediante el oscilador interno.

El PC ocupa la posición 2 del área de datos en donde se halla el registro PCL al que se añaden 3 bits auxiliares y se conectan con los dos niveles de la Pila en las instrucciones CALL y RETLW.

El registro de Estado ocupa la posición 3 y entre sus bits se encuentran los señalizadores C, DC y Z y los bits PA1 y PA0 que seleccionan la página en la memoria de programa. El bit 7 (PA2) no está implementando en los PIC de la gama baja.

FRS se ubica en la dirección 4 y puede usarse para contener la dirección del dato en las instrucciones con direccionamiento indirecto y también para guardar operandos en sus 5 bits de menos peso.

Los registros que ocupan las posiciones 5 ,6 y 7 soportan los Puertos A, B y C de E/S. Pueden ser leídos y escritos como cualquier otro registro y manejan los valores de los bits que entran y salen por los pines de E/S del micro controlador.

### *4.2.2. PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR*

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. El comando puede ser un mnemónico del conjunto de instrucciones. 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 aunque es buena práctica considerarlo obligatorio para el programador.

La etiqueta, 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. Identifica la línea de programa haciendo que el compilador le asigne un valor automáticamente. Si se trata de una línea cuyo comando es una instrucción de programa del micro controlador, se le asigna el valor de la dirección de memoria correspondiente a dicha instrucción (location counter). En otros casos se le asigna un valor de una constante, o la dirección de una variable, o será el nombre de una macroinstrucción, etc.

El comando puede ser un código mnemónico de instrucción del micro controlador, o una directiva o pseudoinstrucción para el compilador. En el primer caso será directamente traducido a código de maquina, en el segundo caso será interpretado por el compilador y realizara alguna acción en tiempo de compilación como ser asignar un valor a una etiqueta, etc.

El campo de parámetros puede contener uno o más parámetros separados por comas. Los parámetros dependen de la instrucción o directiva. Pueden ser números o literales que representen constantes o direcciones.

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. De esta manera pueden incluirse líneas que contengan solo comentarios, y es muy buena práctica hacer uso y abuso de esta posibilidad para que los programas resulten autodocumentados.

#### *4.2.2.1. CONJUNTO DE INSTRUCCIONES*

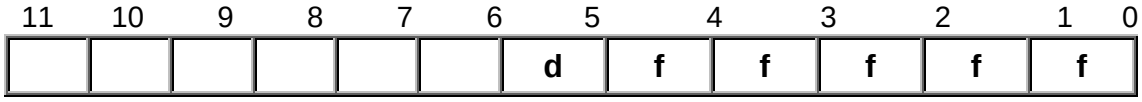
El conjunto de instrucciones de los microprocesadores PIC 16CXX consiste en un pequeño repertorio de solo 33 instrucciones de 12 bits, que pueden ser agrupadas para su estudio en tres a cinco grupos. En este curso se ha optado por clasificarlas, desde el punto de vista del programador, en cinco categorías bien definidas de acuerdo con la función y el tipo de operandos involucrados. En primer lugar se agrupan las instrucciones que operan con

bytes y que involucran algún registro de la memoria interna. En segundo lugar se analizaran las instrucciones que operan solo sobre el registro W y que permiten cargarle una constante implícita o incluida literalmente en la instrucción (literales). En tercer lugar se agrupan las instrucciones que operan sobre bits individuales de los registros de la memoria interna. En cuarto lugar se clasifican las instrucciones de control de flujo del programa, es decir las que permiten alterar la secuencia lineal de ejecución de las instrucciones. Por último se agrupan unas pocas instrucciones que llamaremos especiales, cuyas funciones o tipos de operandos son muy específicos y no encajan en ninguna de las clasificaciones anteriores.

#### *4.2.2.1.1. INSTRUCCIONES DE BYTE QUE OPERAN CON REGISTROS*

Estas instrucciones pueden ser de simple o doble operando de origen. El primer operando de origen será siempre el registro seleccionado en la instrucción, el segundo, en caso de existir, será el registro W. El destino, es decir donde se guardara el resultado, será el registro seleccionado o el W, según se seleccione con un bit de la instrucción.

El formato genérico de estas instrucciones es el siguiente:



Los bits 0 a 4 (5 bits), denominados “f” permiten seleccionar uno de 32 registros de la memoria interna. El bit 5, denominado “d”, permite especificar el destino del resultado. Si d = 1 el resultado se guardara en el registro seleccionado. Si d = 0 el resultado se guardara en W. Los bits 6 a 11 identifican la instrucción específica a realizar.

Las instrucciones siguientes son las tres operaciones lógicas de doble operando:

***ANDWF f, d; operación AND lógica, destino = W Ù f***
***IORWF f, d; operación OR lógica, destino = W Ú f***
***XORWF f, d; operación XOR lógica, destino = W Å f***

Los nombres mnemónicos de estas instrucciones provienen de: AND W con F, Inclusive OR W con F y XOR W con F.

Las que siguen son las cuatro operaciones aritméticas y lógicas sencillas de simple operando:

***MOVF f, d; movimiento de datos, destino = f***
***COMF f, d; complemento lógico, destino = NOT f***
***INCF f, d; incremento aritmético, destino = f + 1***
***DECF f, d; decremento aritmético, destino = f - 1***

Los mnemónicos de estas instrucciones provienen de: MOVe File, COMplement File, NCrement File y DECrement File.

En las siete instrucciones anteriores el único bit afectado de la palabra de estado del procesador es el Z, que se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor.

A continuación siguen las dos instrucciones de rotación de bits a través del CARRY:

***RLF f, d; rotación a la izquierda, destino = f ROT ¬***
***RRF f, d; rotación a la derecha, destino = f ROT ®***

En estas operaciones (Rotate Left File y Rotate Right File) los bits son desplazados de cada posición a la siguiente, en sentido derecho o izquierdo. El desplazamiento es cerrado, formando un anillo, con el bit C (CARRY) de la palabra de estado.

En estas dos instrucciones, el único bit afectado de la palabra de estado del procesador es el bit C, que tomará el valor que tenia el bit 7 o el bit 0, según sea el sentido del desplazamiento.

Estas instrucciones son muy útiles para la manipulación de bits, y además para realizar operaciones aritméticas, ya que en numeración binaria, desplazar un número a la izquierda es equivalente a multiplicarlo por 2, y hacia la derecha, a dividirlo por 2.

La instrucción siguiente realiza el intercambio de posiciones entre los cuatro bits menos significativos y los cuatro más significativos (nibble bajo y nibble alto).

***SWAPF f, d; intercambia nibbles, destino = SWAP f***

Esta instrucción (SWAP File) no afecta ninguno de los bits de la palabra de estado del procesador.

Esta instrucción es muy útil para el manipuleo de números BCD empaquetados, en los que en un solo byte se guardan dos dígitos BCD (uno en cada nibble).

Las dos operaciones que siguen son la suma y la resta aritméticas:

***ADDWF f, d; suma aritmética, destino = f + W***

***SUBWF f, d; resta aritmética, destino = f - W***

Estas operaciones (ADD W a F y SUBstract W de F) afectan a los tres bits de estado C, DC y Z.

El bit Z se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor.

La suma se realiza en aritmética binaria pura sin signo. Si hay un acarreo del bit 7, es decir que el resultado es mayor que 255, el bit C (Carry) resulta 1, en caso contrario resulta 0. Si hay un acarreo del bit 3, es decir que la suma de las dos mitades (nibbles) menos significativas (bits 0 a 3) resulta mayor que 15, se pone en 1 el bit DC (Digit Carry), en caso contrario se pone en 0.

Ejemplos:

```
  1010 0010                1101 0000
+ 0100 1111  C DC Z  +    0110 1111 C DC Z
  1111 0001   0  1  0      0011 1111  1  0  0
```

La resta se realiza sumando, en binario puro sin signo, el registro f más el complemento a dos de W (el complemento a 1, o complemento lógico, más 1)

Ejemplos:

```
f →          0100 0100              0010 1000
W →        - 0010 1000  C DC Z    - 0100 0100  C DC Z
             0001 1100   1 0  0     1110 0100   0  1  0
```

Equivalente a:

```
f →          0100 0100              0010 1000
cmp.2 W →  + 1101 1000  C DC Z    + 1011 1100  C DC Z
             0001 1100  1  0  0     1110 0100   0  1  0
```

Los bits de estado C y DC toman el valor normal correspondiente a la suma de f con el complemento a 2 de W. De esta manera el significado para la operación de resta resulta invertido, es decir que C (Carry) es 1 si no hubo

desborde en la resta, o dicho de otra manera, si el contenido de W es menor que el de f. El bit DC se comporta de manera similar, es decir que DC es 1 si no hubo desborde en la mitad menos significativa, lo que equivale a decir que el nibble bajo del contenido de W es menor que el del registro f.

Las instrucciones que siguen son de simple operando, pero son casos especiales ya que el destino es siempre el registro seleccionado:

***CLRF f; borrado de contenido, f = 0***

***MOVWF f; copia contenido W ® f, f = W***

La instrucción CLRF (CLeaR File) afecta solo al bit Z que resulta siempre 0.

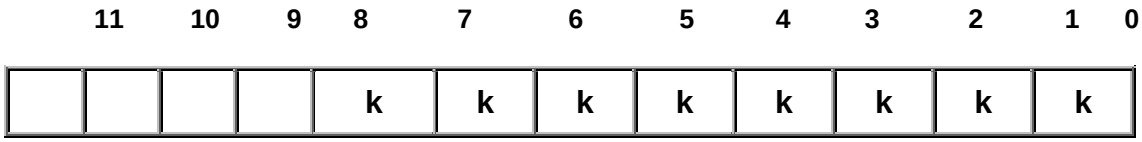
La instrucción MOVWF (MOVe W a F) no afecta ningún bit de la palabra de estado.

#### *4.2.2.1.2. INSTRUCCIONES DE BYTE QUE OPERAN SOBRE W Y LITERALES*

Estas instrucciones se refieren todas al registro W, es decir que uno de los operandos de origen y el operando de destino son siempre el registro W. En las instrucciones de este grupo que tienen un segundo operando de origen, este

es siempre una constante de programa literalmente incluida en la instrucción, llamada constante literal o simplemente literal.

El formato genérico de estas instrucciones es el siguiente:




Los bits 0 a 7 especifican la constante literal de 8 bits que se utilizara en la operación.

Las tres instrucciones que siguen son las operaciones lógicas tradicionales, similares a las que ya vimos anteriormente, pero realizadas entre una constante de programa y el registro W:

***IORLW k; operación OR lógica, W = W Ú k***

***ANDLW k; operación AND lógica, W = W Ù k***

***XORLW k; operación XOR lógica, W = W Å k***

En estas tres instrucciones (Inclusive OR Literal W, AND Literal W y XOR Literal W) el único bit afectado de la palabra de estado del procesador es el Z, que se pone en 1 si el resultado de la operación es 00000000, y se pone en 0 si el resultado tiene cualquier otro valor.

La instrucción que sigue sirve para cargar una constante de programa en el registro W:

***MOVLW k; carga constante en W, W = K***

Esta (MOVe Literal W) instrucción no afecta ninguno de los bits de estado del procesador.

La instrucción que sigue (CLeaR W) no correspondería incluirla en este grupo, y pertenece en realidad al primero, el de las instrucciones que operan sobre registros, ya que se trata de un caso especial de la instrucción CLRF, con destino W, y f = 0. La incluimos aquí porque como se le ha asignado un mnemónico particular referido específicamente al registro W, creemos que, desde el punto de vista del programador, es más útil verla dentro del grupo de instrucciones referidas a W.

***CLRW; borra el contenido de W, W = 0***

Al igual que en la instrucción CLRF, el único bit de estado afectado es el Z que resulta 1.

#### *4.2.2.1.3. INSTRUCCIONES DE BIT*

El formato genérico de estas instrucciones es el siguiente:

| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | **b** | **b** | **b** | **F** | **f** | **f** | **f** | **f** |

Los bits 0 a 4 (5 bits), denominados “f”, permiten seleccionar uno de 32 registros de la memoria interna. Los bits 5 a 7, denominados “b”, permiten especificar el numero de bit (0 a 7) sobre el que se operara. Estas instrucciones operan solamente sobre el bit especificado, el resto de los bits del registro no son alterados. Estas instrucciones no tienen especificación de destino, ya que el mismo es siempre el registro seleccionado.

***BCF f, b; borra el bit b de f; bit f (b) = 0***

***BSF f, b; coloca en uno el bit b de f; bit f (b) = 1***

Estas instrucciones (Bit Clear File y Bit Set File) no afectan ningún bit de la palabra de estado del procesador.

#### *4.2.2.1.4. INSTRUCCIONES DE CONTROL*

***GOTO k; salto a la posición k (9 bits) del programa***

Esta es la típica instrucción de salto incondicional a cualquier posición de la memoria de programa (que en la mayoría de los microprocesadores convencionales se llama JUMP). La constante literal k es la dirección de destino del salto, es decir la nueva dirección de memoria de programa a partir de la cual comenzarán a leerse las instrucciones después de ejecutar la instrucción GOTO. Esta instrucción simplemente carga la constante k en el registro PC (contador de programa). La única complicación de esta instrucción es que la constante k es de solo 9 bits, mientras que el registro PC es de 11 bits, ya que en el 16C57 debe permitir direccionar una memoria de programa de 2 K. Los dos bits faltantes, bit 9 y 10 del PC, son tomados respectivamente de los bits de selección de página PA0 y PA1 de la palabra de estado.

Este comportamiento particular hace que la memoria de programa aparezca como dividida en paginas de 512 posiciones como se vera más adelante. El programador debe tener en cuenta que antes de ejecutar una instrucción GOTO es posible que haya que programar los bits PA0 y PA1.

La que sigue es la instrucción de llamado a subrutina:

***CALL k; salto a la subrutina en la posición k (8 bits)***

Su comportamiento es muy similar al de la instrucción GOTO, salvo que además de saltar guarda en el stack la dirección de retorno de la subrutina (para la instrucción RETLW). Esto lo hace simplemente guardando en el stack

una copia del PC incrementado, antes de que el mismo sea cargado con la nueva dirección k. La única diferencia con la instrucción GOTO respecto de la forma en la que se realiza el salto, es que en la instrucción CALL la constante k tiene solo 8 bits en vez de 9. En este caso también se utilizan PA0 y PA1 para cargar los bits 9 y 10 del PC, pero además el bit 8 del PC es cargado siempre con 0. Esto hace que los saltos a subrutina solo puedan realizarse a posiciones que estén en las primeras mitades de las páginas mencionadas. El programador debe tener en cuenta este comportamiento y asegurarse de ubicar las posiciones de inicio de las subrutinas en las primeras mitades de las páginas.

La instrucción que aparece a continuación es la de retorno de subrutina:

***RETLW k; retorno de subrutina con constante k, W = k***

Esta (RETurn con Literal in W) instrucción produce el retorno de subrutina con una constante literal k en el registro W. La operación que realiza consiste simplemente en sacar del stack un valor y cargarlo en el PC. Ese valor es el PC incrementado antes de realizar el salto, de la última instrucción CALL ejecutada, por lo tanto es la dirección de la instrucción siguiente a dicho CALL. Dado que el stack es de 11 bits, el valor cargado en el PC es una dirección completa, y por lo tanto se puede retornar a cualquier posición de la memoria de programa, sin importar como estén los bits de selección de página. Esta instrucción además carga siempre una constante literal en el registro W. Ya que esta es la única instrucción de retorno de subrutina de los PIC16CXX, no hay en estos microprocesadores forma de retornar de una subrutina sin alterar el registro W.

Por otro lado, y con una metodología especial de programación, un conjunto de sucesivas instrucciones RETLW puede ser usado como una tabla de valores constantes incluida en el programa (Ej.: tablas BCD/7 Seg., hexa/ASCII, etc.).

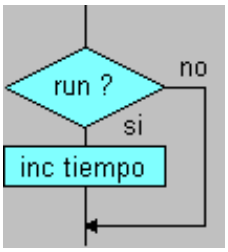
A continuación se presentan las dos únicas instrucciones de “salteo” (skip) condicional. Estas instrucciones son los únicos medios para implementar bifurcaciones condicionales en un programa. Son muy generales y muy poderosas ya que permiten al programa tomar decisiones en función de cualquier bit de cualquier posición de la memoria interna de datos, y eso incluye a los registros de periféricos, los puertos de entrada/salida e incluso la palabra de estado del procesador. Estas dos instrucciones reemplazan y superan a todo el conjunto de instrucciones de salto condicional que poseen los microprocesadores sencillos convencionales (salto por cero, por no cero, por Carry, etc.).

***BTFSC f, b; salteo si bit = 0, bit = f (0) Þ saltea***

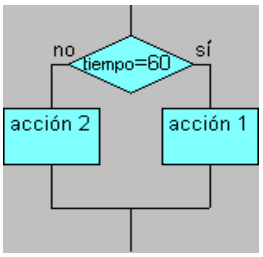
***BTFSS f, b; salteo si bit = 1, bit = f (1) Þ saltea***

BTFSC (Bit Test File and Skip if Clear) saltea la próxima instrucción si el bit b del registro f es cero. La instrucción BTFSS (Bit Test File and Skip if Set) saltea si el bit es 1. Estas instrucciones pueden usarse para realizar o no una acción según sea el estado de un bit, o, en combinación con GOTO, para realizar una bifurcación condicional.

**Ejemplo 1:**




**Ejemplo 2:**




Las instrucciones que siguen son casos especiales de las de incremento y decremento vistas anteriormente. Estas instrucciones podrían categorizarse dentro del grupo de instrucciones orientadas a byte sobre registros (primer grupo), ya que efectivamente operan sobre los mismos, y el formato del código de la instrucción responde al de ese grupo, pero, a diferencia de las otras, pueden además alterar el flujo lineal del programa y por eso se les incluyó en este grupo.

***DECFSZ f, d; decrementa y saltea sí 0, destino= f - 1, = 0 Þ saltea***

***INCFSZ f, d; incrementa y saltea sí 0, destino= f + 1, = 0 Þ saltea***

Estas dos instrucciones (DECrement File and Skip if Zero, e INCrement File and Skip if Zero) se comportan de manera similar a DECF e INCF, salvo que no afectan a ningún bit de la palabra de estado. Una vez realizado el

incremento o decremento, si el resultado es 00000000, el microprocesador salteara la próxima instrucción del programa. Estas instrucciones se utilizan generalmente en combinación con una instrucción de salto (GOTO), para el diseño de ciclos o lazos (loops) de instrucciones que deben repetirse una cantidad determinada de veces.

Ejemplo:

Clrf 10; pongo cero en la posición 10 de la memoria interna

Loop; lo que sigue se ejecutará 256 veces

Incfsz 10,1;  incremento la posición 10 hasta que llegue a 0

Goto loop;  si no llego a cero voy a repetir la secuencia  cuando llegue a cero salteo el goto y sigue la continuación del programa.

#### *4.2.2.1.5. INSTRUCCIONES ESPECIALES*

En este grupo se reunieron las instrucciones que controlan funciones específicas del microprocesador o que actúan sobre registros especiales no direccionados como memoria interna normal.

La instrucción que sigue es la típica NO OPERATION, existente en casi todos los microprocesadores.

***NOP***; no hace nada, consume tiempo

Esta instrucción solo sirve para introducir una demora en el programa, equivalente al tiempo de ejecución de una instrucción. No afecta ningún bit de la palabra de estado.

La siguiente es una instrucción específica de control de los puertos de entrada/salida.

***TRIS f; carga el tristate control, TRISf = W***

Esta instrucción (TRIState) carga el registro de control de los buffers tristate de un puerto de entrada salida (data dirección Register), con el valor contenido en W. El parámetro f debe ser la dirección de memoria interna del puerto, aunque el valor W no será cargado en el puerto sino en el registro de tristate del mismo. Los valores validos para f son 4 y 5 en los 16C54/56 y 4, 5 y 6 en los 16C55/57. Esta instrucción no afecta ningún bit de la palabra de estado.

La siguiente instrucción sirve para programar el registro OPTION que controla el RTCC y prescaler

***OPTION; carga el registro OPTION, OPTION = W***

El registro OPTION no es accesible como memoria interna y solo se lo puede programar con esta instrucción. Esta instrucción no afecta ningún bit de la palabra de estado.

La instrucción que sigue borra el contador del Watch Dog Timer. Este registro tampoco esta accesible como memoria, y esta es la única instrucción que lo modifica.

***CLRWDT; borra el watch dog timer, WDT = 0***

Esta instrucción, además, coloca en uno los bits PD (Power Down) y TO (time-Out) de la palabra de estado.

La siguiente es una instrucción especial de control del micro controlador que lo pone en el modo Power Down. En este modo el microprocesador se detiene, el oscilador se apaga, los registros y puertos conservan su estado, y el consumo se reduce al mínimo. La única forma de salir de este estado es por medio de un reset o por time-Out del Watch dog Timer.

***SLEEP; coloca el µC en modo sleep, WDT = 0***

Esta instrucción, además, borra el bit PD (Power Down) y setea el bit TO (time-Out) de la palabra de estado.

#### *4.2.2.1.6. DIRECCIONAMIENTO DE LA MEMORIA DE DATOS (RAM)*

La memoria interna se direcciona en forma directa por medio de los 5 bits “f” contenidos en las instrucciones que operan sobre registros. De esta manera se puede direccionar cualquier posición desde la 00 a la 1F. Como se vió en el capítulo correspondiente a los mapas de memoria, las direcciones 10 a 1F corresponden a los bancos de registros, por lo tanto, en los micro controladores que tengan más de un banco, antes de acceder a alguna variable que se encuentre en esta zona, el programador deberá asegurarse de haber programado los bits de selección de banco en el registro FSR.

Los registros especiales y de uso general de la posición 00 a la 0f están presentes en todos los PIC16CXX, al igual que el banco 0 de registros. Los bancos 1, 2 y 3 de registros están presentes solo en el 16C57.

El registro FSR, además de servir para seleccionar el banco activo, sirve como puntero para direccionamiento indirecto. La posición 00 del mapa de RAM es la llamada dirección indirecta. Sí en cualquier instrucción se opera con la dirección 00, en realidad se estará operando con la dirección a donde apunte el contenido del FSR. Por ejemplo si el FSR contiene el valor 14, una instrucción que opere sobre la dirección 0, operara en realidad sobre la dirección 14. Se puede decir en este ejemplo que la posición 14 de memoria fue direccionada en forma indirecta a través del puntero FSR.

Ejemplo:

Esta porción de programa borra 5 posiciones de memoria a partir de la dirección 12

***FSR equ 04 ; (definición al comienzo del programa)***

***Movlw 5; prepara para repetir 5 veces***

***Movwf 08 ;(el registro 08 es el contador del loop)***

***Movlw 12h; apunta a la dirección 12h***

***Movwf FSR;***

***Loop:***

***Clrf 0; borra una posición de memoria***

***Incf FSR; apunta a la siguiente***

***Decfsz 08; si todavía no borra todas***

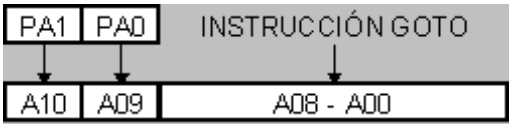
***Goto loop; sigue borrando***

El direccionamiento indirecto es muy útil para el procesamiento de posiciones consecutivas de memoria, como en el ejemplo, o para el direccionamiento de datos en subrutinas.

#### *4.2.2.1.7. DIRECCIONAMIENTO DE LA MEMORIA DE PROGRAMA (EPROM, OTP)*

La instrucción GOTO dispone solo de 9 bits en el código de operación para especificar la dirección de destino del salto. Al ejecutar una instrucción GOTO el microprocesador toma los dos bits que restan para completar la dirección de 11 bits, de los bits 5 y 6 de la palabra de estado. Estos últimos son llamados bits de selección de página (PA0 y PA1). El programador deberá asegurarse de que estos dos bits tengan el valor correcto antes de toda instrucción GOTO.

Direccionamiento directo con instrucción GOTO



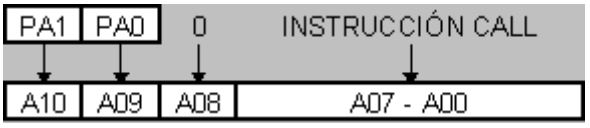

Deberá tenerse en cuenta además que es posible avanzar de una página a otra en forma automática cuando el PC se incrementa. Esto ocurre si el programa empieza en una página y sigue en la siguiente.

Sin embargo, al incrementarse el PC desde la última posición de una página a la primera de la siguiente, los bits PA0 y PA1 no se modifican, y por lo tanto sí se ejecuta una instrucción GOTO, CALL o alguna que actúe sobre el PC, esta producirá un salto a la página anterior, a menos que el programador

tenga la precaución de actualizar el valor de dichos bits. Por este motivo es conveniente dividir el programa en módulos o rutinas que estén confinados a una página.

En el caso de la instrucción CALL, el direccionamiento se complica un poco más, ya que la misma solo dispone de 8 bits para especificar la dirección de destino salto. En este caso también se utilizan los mismos bits de selección de página para completar los bits décimo y decimoprimero de la dirección, pero falta el noveno bit. En estas instrucciones este bit se carga siempre con 0, lo que implica que solo se pueden realizar saltos a subrutina a las mitades inferiores de cada página. En este caso también el programador deberá asegurarse que el estado de los bits PA0 y PA1 sea el correcto al momento de ejecutarse la instrucción.

Direccionamiento directo con instrucción CALL




Las instrucciones que operan sobre el PC como registro y alteran su contenido provocando un salto, responden a un mecanismo muy similar al de las instrucciones CALL para la formación de la dirección de destino. En este caso los bits 0 a 7 son el resultado de la instrucción, el bit 8 es 0 y los bits restantes se toman de PA0 y PA1.

Este mecanismo se llama paginado, y a pesar de que representa una complicación bastante molesta para el programador, resulta muy útil ya que permite ampliar la capacidad de direccionamiento de memoria de programa para las instrucciones de salto.

### *4.2.3. EL PROGRAMADOR JDM.*

El programador JDM, también conocido como PIC-Programmer 2, ha sido y es ampliamente utilizado y funciona bien. Fue desarrollado por Jens Dyekjær Madsen (JDM) entre 1996 y 1998. Los programadores TE-20, TE-20SE y el propio JDMD, así como otros, están basados en él. Se utiliza con el WinPic800 seleccionando JDM Programer en la configuración de hardware.

Este programador es alimentado por el puerto RS232 del PC y funciona con niveles RS232 <±8.6V.

#### *4.2.3.1. DISPOSITIVOS QUE SOPORTA*

12C508, 12C508A, 12C509, 12C509A, 12CE518, 12CE519

12C671, 12C672, 12CE673, 12CE674

12F629, 12F675

16C433

16C54, 16C56, 16C58

16C61, 16C62A, 16C62B, 16C63, 16C63A, 16C64A, 16C65A, 16C65B, 16C66, 16C67

16C71, 16C72, 16C72A, 16C73A, 16C73B, 16C74A, 16C74B, 16C76, 16C77

16F73, 16F74, 16F76, 16F77

16C84, 16F83, 16F84, 16F84A

16F88

16C505

16C620, 16C620A, 16C621, 16C621A, 16C622, 16C622A

16CE623, 16CE624, 16CE625

16F627, 16F628

16F628A, 16F648A

16F630, 16F676

16C710, 16C711, 16C712, 16C715, 16C716, 16C717, 16C745, 16C765

16C770, 16C771, 16C773, 16C774, 16C781, 16C782

16F818, 16F819

16F870, 16F871, 16F872, 16F873, 16F874, 16F876, 16F877

16F873A, 16F874A, 16F876A, 16F877A

16C923, 16C924

18F242, 18F248, 18F252, 18F258, 18F442, 18F448, 18F452, 18F458

18F1320, 18F2330, 18F4320, 18F6620, 18F6720, 18F8620, 18F8720

### *4.2.3.2. ESQUEMA*

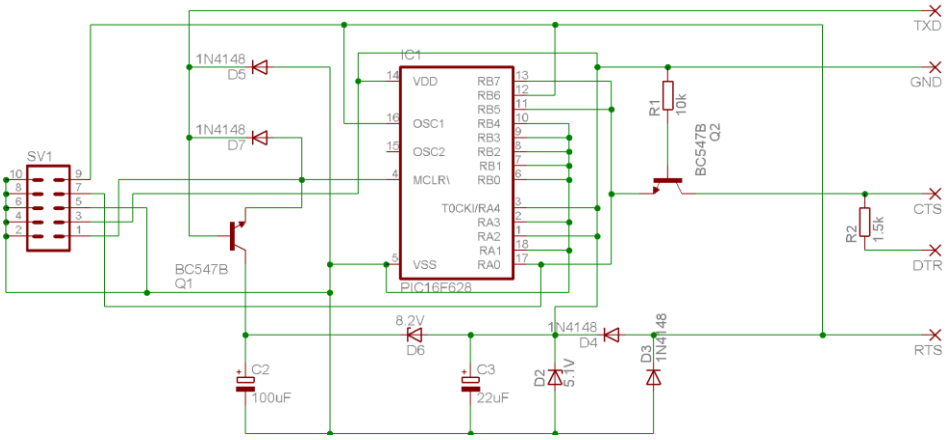



Figura 4.1 Esquema de un programador JDM

Las tensiones necesarias, Vpp (tensión de programación) y Vdd se obtienen a través del puerto. La tensión se estabiliza con diodos zener y condensadores.

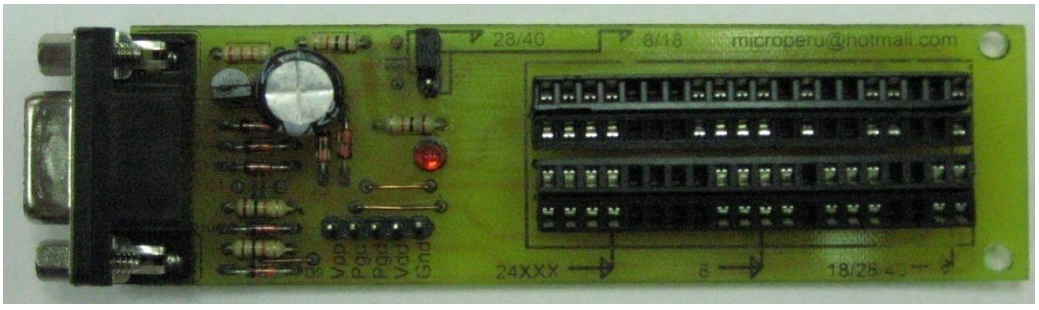



Figura 4.2 Fotografía de Programador JDM para prácticas de Laboratorio

#### *4.2.3.3. ICSP.*

El programador JDM soporta ICSP, In-Circuit Serial Programming, Esta salida de 5 líneas (Vpp, Pgc, Pgd, Vdd, Gnd) sirve para programar PICs que no sean soportados por el zócalo del programador o que tengan otro tipo de empaque, se conecta las líneas a los pines correspondientes del PIC a programar, y se procede de manera similar para programar usando el zócalo.

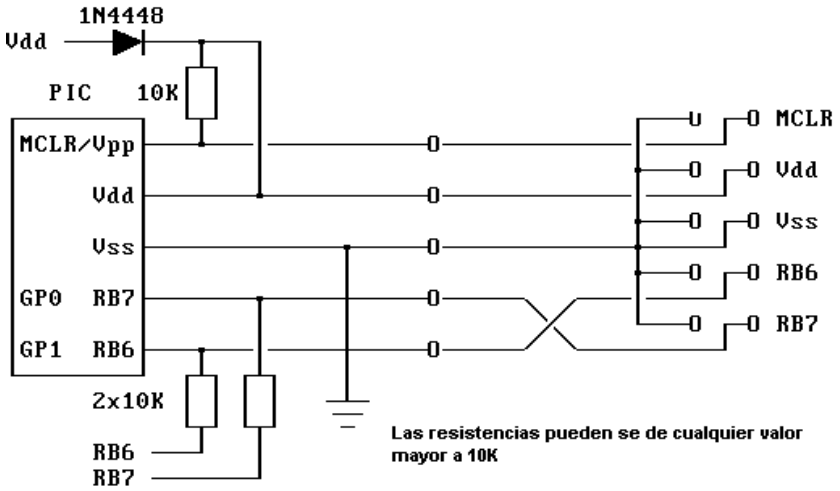



Figura 4.3 Esquema de conexión a puerto serial

IMPORTANTE: Mientras se utiliza el programador no debe utilizarse la alimentación del circuito de utilización, la alimentación necesaria es aplicada por el propio programador.

**1. Jumper 8/18 - 28/40.**

Este jumper selecciona el tipo de PIC a programar, por lo tanto si se programan PICs de 8 y 18 pines (12F675, 12F683, 16f84A, 16F628A, etc.) se coloca el

jumper hacia ese lado, de ese modo cuando se programen PICs de 28 y 40 pines (16F876, 16F877A, etc.) se coloca el jumper hacia ese lado.

2. **Zócalo.**

El zócalo tiene una indicación a su costado, las cuales indican el tipo de PIC a programar señalando la cabeza o muesca del PIC con una flecha:

- **18/28/40**. Indica que en esa posición se coloca los PICs de 18, 28 y 40 pines con la cabeza o muesca del PIC en la posición indicada por la flecha.

- **8.** Indica que en esa posición se coloca los PICs de 8 pines con la cabeza o muesca del pic en la posición indicada por la flecha.

- **24XXX**. Indica que en esa posición se coloca las memorias i2c con la cabeza o muesca de la memoria en la posición indicada por la flecha.

### *4.2.3.4. CARACTERÍSTICAS DEL PUERTO SERIE RS232*

- Nivel alto: -3v a -15v
- Nivel bajo: +3v a +15v
- Tamaño de las palabras enviadas: 5,6, 7 u 8 bits
- Posee paridad par, impar o ninguna
- Conectores de 9 y 25 pines
- Un PC de escritorio suele soportar 1 o 2 (COM1 y COM2)

El puerto serie de 25 pines no suele encontrarse ya en los Pcs de escritorio, donde sólo se presenta el de 9 pines. Al utilizar el JDM deberá tenerse en cuenta.

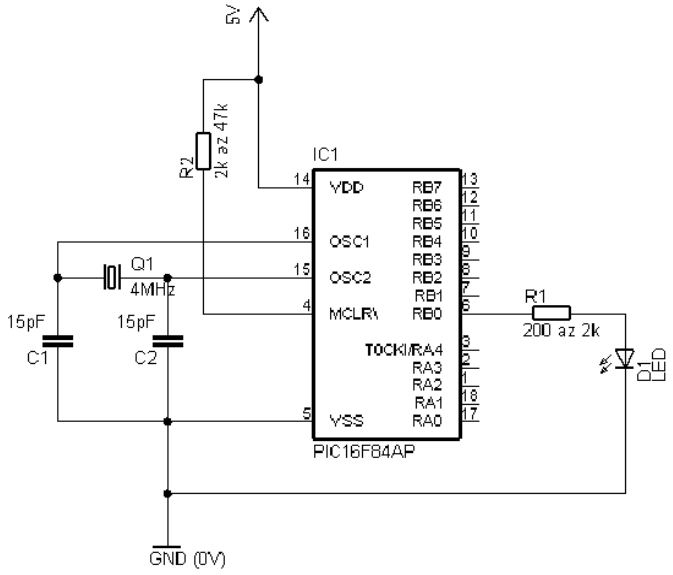
### *4.2.4. KIT DE PRÁCTICAS.*

Básicamente el Kit de prácticas es el conjunto de todos los componentes y herramientas que el alumno puede necesitar para el desarrollo de las prácticas en el laboratorio. Sin el kit de prácticas el desarrollo de las guías de laboratorio no seria el apropiado.

#### *4.2.4.1. ENTRENADOR.*

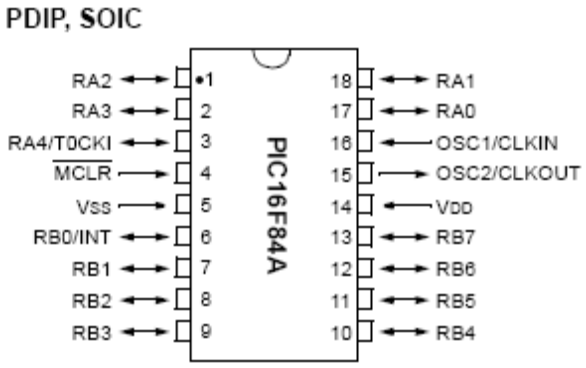
Existe una gran variedad de diseños en cuanto a entrenadores se refiere, se puede ir desde modelos sencillos para comprobar a través de un led el accionar de un puerto hasta el manejo de dispositivos de la talla de LCD´s o motores paso a paso. En este apartado, es de interés el presentar un modelo factible que sea fácil de entender al alumno en cuanto a su diseño y modos de funcionamiento. El entrenador es un conjunto de periféricos organizado de forma estática o escalables que puedan acomodarse a la necesidad de probar un programa determinado o generalización de pruebas de entrada y salida; estos se disponen sobre una placa de cobre o perforada (especiales para el diseño de prototipos) conectados de tal forma que les permitan poder interactuar directamente con el micro controlador según el programa almacenado.

Un diseño básico para hacer funcionar el 16F84 es necesario conectarle un cristal oscilador (de 4MHz o 20MHz) con condensadores de 22pF a masa, una resistencia de Vdd a MCLR y alimentar a 5V. Se inicia conectando un led en el pin que se va a utilizar con una resistencia de 330ohm a masa. Eso será suficiente para empezar, se muestra en la siguiente figura un esquema.




La alimentación del PIC nunca debe exceder de 5V. Si se va a utilizar una pila o cualquier otro medio que supere lo recomendado se tendrá que regular el voltaje mediante un LM7805 para conseguir los 5V de rigor. Si se utiliza una fuente de PC (computadora de escritorio), no se tendrá ningún problema (el cable rojo provee 5V y el amarillo 12V).

La escalabilidad del dispositivo va a permitir el acomodar nuevas piezas de prueba como el de utilizar las básicas de su diseño o agregar nuevas, según la necesidad de las guías de trabajo o la prueba al Hardware; avanzamos un poco incluyendo al diseño básico dispositivos de entrada que proporcionen datos al Micro controlador, estos serán los pulsadores (Switch) que se conectara a uno de los puertos disponibles como entrada. Presentemos entonces una pequeña organización de lo que será el diseño del entrenador. Empezaremos tomando en cuenta la distribución de las patas (pines) con las que cuenta nuestro PIC, utilizaremos el modelo 16F84 en encapsulado PDIP para el diseño.




La distribución de los puertos se presenta así:

**Pines 1, 2, 3, 17 y 18 (RA0-RA4/TOCKI):** Es el PORT A. Corresponden a 5 líneas bidireccionales de E/S (definidas por programación). Es capaz de entregar niveles TTL cuando la alimentación aplicada en VDD es de 5V ± 5%. 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, no te preocupes, mas abajo lo entenderás mejor). Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega a la salida un "1" lógico. Este pin como salida no puede manejar cargas como fuente, sólo en el modo sumidero.

**Pin 4 (MCLR / Vpp):** Es una pata de múltiples aplicaciones, es la entrada de Reset (máster clear) si está a nivel bajo y también es la habilitación de la tensión de programación cuando se está programando el dispositivo. Cuando su tensión es la de VDD el PIC funciona normalmente.

**Pines 5 y 14 (VSS y VDD):** Son respectivamente las patas de masa y alimentación. La tensión de alimentación de un PIC está comprendida entre 2V y 6V aunque se recomienda no sobrepasar los 5.5V.

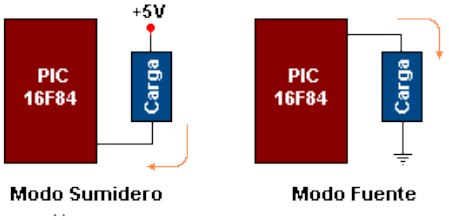
**Pines 6, 7, 8, 9, 10, 11, 12, 13 (RB0-RB7):** Es el PORT B. Corresponden a ocho líneas bidireccionales de E/S (definidas por programación). Pueden manejar niveles TTL cuando la tensión de alimentación aplicada en VDD es de 5V ± 5%. RB0 puede programarse además como entrada de interrupciones externas INT. Los pines RB4 a RB7 pueden programarse para responder a interrupciones por cambio de estado. Las patas RB6 y RB7 se corresponden con las líneas de entrada de reloj y entrada de datos respectivamente, cuando está en modo programación del integrado.

**Pines 15 y 16 (OSC1/CLKIN y OSC2/CLKOUT):** Corresponden a los pines de la entrada externa de reloj y salida de oscilador a cristal respectivamente.

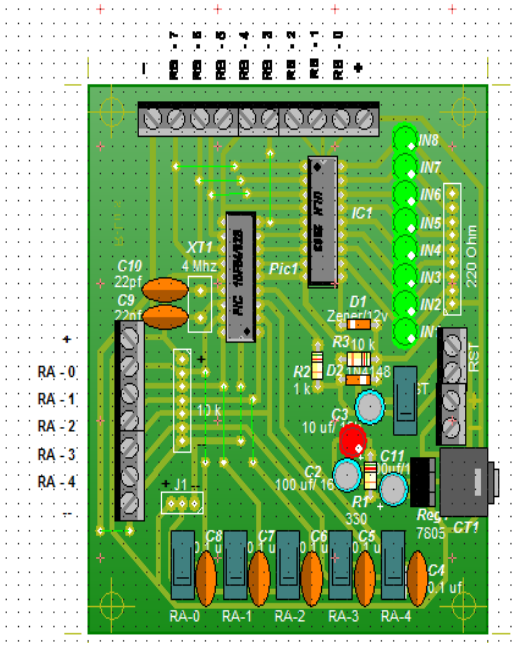
## Capacidad de corriente en los puertos

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

| | PUERTO A | PUERTO B |
|---|---|---|
| **Modo Sumidero** | 80 mA | 150 mA |
| **Modo Fuente** | 50 mA | 100 mA |

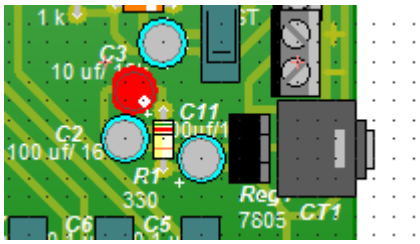


Teniendo en cuenta esta descripción de los pines y una pequeña explicación de su funcionamiento al momento de ser emisores como receptores de voltaje, procedemos a mostrar el diseño básico del prototipo de entrenador, el cual consta de las siguientes partes:




### *4.2.4.1.1. FUENTE DE ALIMENTACIÓN:*

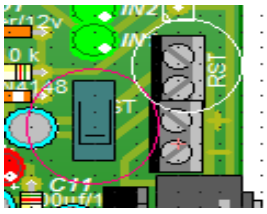
Se ha diseñado dos tipos de alimentación, una con baterías y otra con una Terminal VCC, por lo cual se incluye un rectificador de tensión 7805 para mantener un voltaje de 5V de entrada al entrenador; por lo general se utilizara un fuente rectificada en las prácticas de laboratorio para lo que se ha colocado un conector tipo Plug, pero para su uso con fuentes alternativas como las baterías, se ha colocado la bornera de 2 pines que recibe los polos positivo (+) y negativo (-). Se ha colocado un Led con propósito informativo para saber si le esta llegando tensión a la placa




**Conector Plug y Bornera para Baterías**

#### *4.2.4.1.2. RESET:*

Del mismo lado que la fuente de alimentación, a un lado; se encuentra el Switch de reset que regularmente se utiliza en los entrenamientos. En más de alguna ocasión será necesario resetear el micro controlador según la aplicación que se este ejecutando en la placa. Se presenta en dos modos distintos; el primero con un pulsador común y el otro en una bornera, cuando en alguno de los proyectos sea necesario resetear el PIC desde algún sensor.

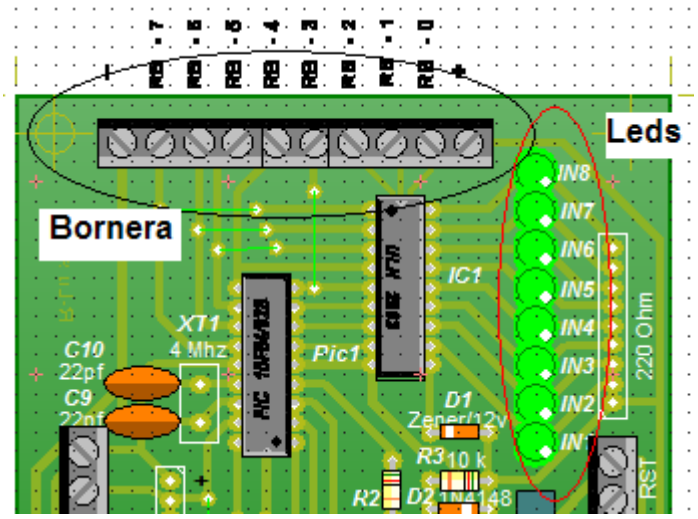



**Switch y Bornera para reset del PIC**

#### *4.2.4.1.3. PUERTO B:*

Consta de 8 pines dispuestos en el Micro controlador de forma bilateral, es decir, que podrían ser programados como receptores (puerto de entrada) o emisores (puerto de salida). En el diseño particular de el entrenador, su función es la de puerto de salida, destinado a dos caminos, un Array de Leds dispuesto de manera correcta y un arreglo ce Borneras con el fin de ampliar la posibilidad de nuestro entrenador de poder adaptarle dispositivos extras como lo serian Display de 7 segmentos, módulos electrónicos que manejen motores paso a

paso, relés como Switches electrónicos, Triacs, etc. En fin, destinado a una gran variedad de posibilidades.



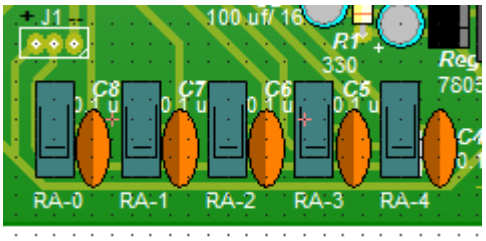

**Ubicación de Borneras y Leds para el Puerto B**

Se ha provisto al puerto B del Micro controlador con un integrado ULN2803 y 8 Leds acompañados de una serie de resistencias de 220 Ohm para el juego de Leds. El integrado ULN2803 tiene como función la de activar los Leds correspondientes a los pines que envíen las señales de activación por parte del Micro controlador (Ver hoja de datos del ULN2803 en Anexos No___), la función de este integrado se limita únicamente a los Leds, la señal de las borneras se deriva directamente de los pines del puerto B, esto con el fin de destinarse a activar otros dispositivos de módulos que puedan ser agregados al entrenador. Al extremo del juego de Borneras se han colocado dos extras, una emitiendo voltaje de 5V y lastra como tierra (Ground) para efectos de proporcionar al dispositivo extra la tensión básica de trabajo, a menos que utilice un voltaje diferente será suministrado por una fuente externa, por ejemplo para activar Relés de 12V.

#### *4.2.4.1.4. PUERTO A:*

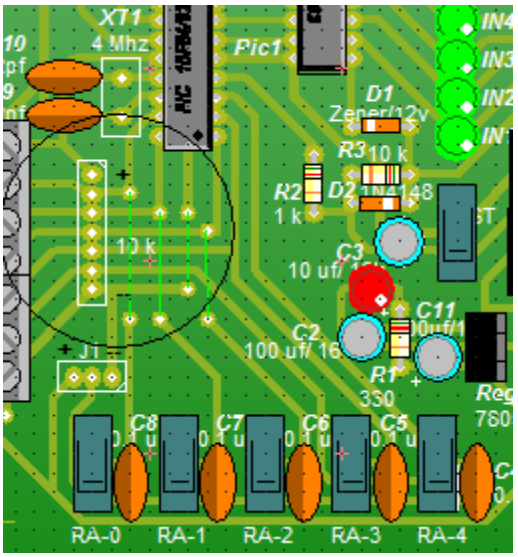
Al puerto A se le han colocado una serie de Switches o Pulsadores, se han colocado de tal manera que pueda ser posible seleccionar la polaridad de ellos, todo esto mediante el uso de un puente de tres puntos o también conocido como Jumper (J1), en donde el punto medio será el común de los Switches y los extremos serán el positivo (+) y el Negativo (-); esto presenta una flexibilidad al usuario de poder establecer una programación en donde se requiera que el Micro controlador reciba pulsos positivos en los puertos de entrada o pulsos negativos, según sea requerido. Es de saber que es el Puerto A el destinado como Entrada, en cada uno de sus pines.

Los capacitares al lado de cada Switch se implementan para evitar la inestabilidad del pulsador, lo que se conoce como rebotes eléctricos.




**Switch (Pulsadores) ubicados en el Puerto A.**

Se ha dispuesto también un Array de resistencias con un valor de 10khm cerca del Jumper de cambio de polaridad, a este juego de resistencias en sus extremos les sobra un pin con polaridad negativa y otro positiva, se explica su función así; si al activar el pulsador envían 0 lógico al Micro controlador, cuando estén en reposo deberán estar en 1 lógico. Esto significa que si el Jumper (J1) de los pulsadores esta unido al polo negativo (-), el Array de resistencias deberá estar unido al positivo, y al reverso cuando el Jumper (J1) se encuentre conectado a la otra polaridad y el Array de resistencias deberá estar conectado al negativo (-), y así al activarse los pulsadores se enviara un 1 lógico.




**Disposición de los Pulsadores destinados al Puerto A**

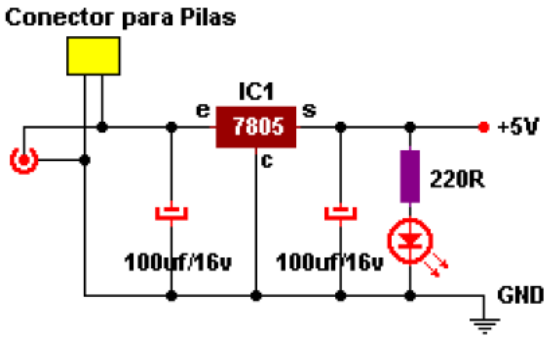
También se ha dispuesto un juego de Borneras para el Puerto A, destinada para cuando sea necesario recibir señales de dispositivos externos, dirigidas al Micro controlador, a cada lado de las Borneras se encuentran los polos positivo (+) y Negativo (-) con el fin de proporcionar a estos dispositivos externos la tensión de trabajo necesaria, de que estos necesiten de un voltaje mayor, será necesaria entonces una fuente de alimentación extra para proporcionar la tensión de trabajo adecuada al dispositivo externo.

El diagrama eléctrico nos proporciona una vista mas adecuada para entender la conexión de los dispositivos en el diseño de nuestro entrenador, este diagrama solo muestra la conexión necesaria, no incluye una descripción detallada de los dispositivos y su ubicación física en la placa, es mas bien su ubicación lógica lo que nos permite ver y hacernos una imagen mental de cómo funcionaria.

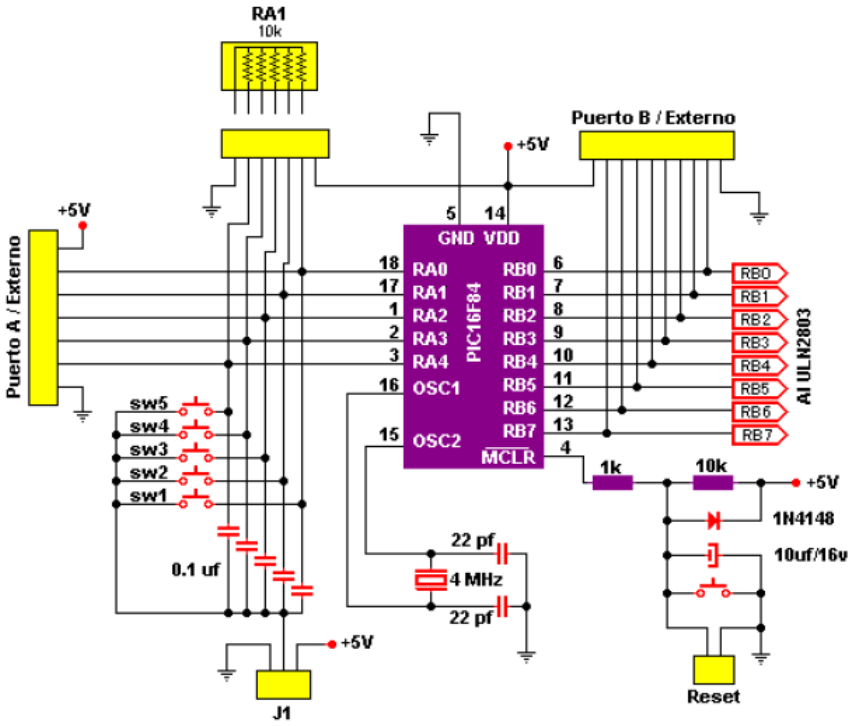
Las imágenes están dispuestas de tal manera de dar una idea de generalización de los dispositivos y no se han dibujado en detalle como lo son

en la imagen del prototipo y el diseño de pistas para la placa cobreada, que se muestran a continuación también.

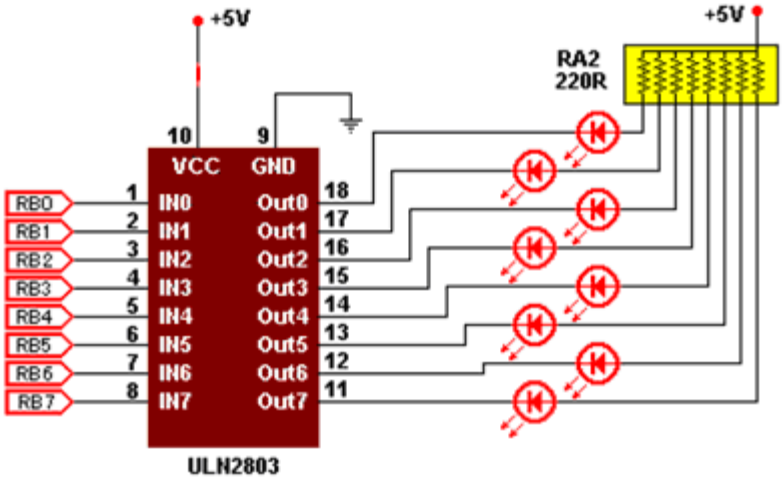
#### *4.2.4.1.5. FUENTE DE ALIMENTACIÓN*




#### *4.2.4.1.6. DIAGRAMA DE CONEXIÓN LÓGICO*



#### *4.2.4.1.7. DIAGRAMA LÓGICO DE CONEXIÓN PARA EL ULN2803*




#### *4.2.4.1.8. DISEÑO DE PROTOTIPO*

El diseño del prototipo del entrenador se ha desarrollado bajo el software del “PCB Wizard”, software de diseño para impresos en cobre. Estos se manejan bajo la técnica de “Transferencia Térmica”.

#### *4.2.4.1.9. TRANSFERENCIA TÉRMICA.*

Existen, como es sabido, diversos métodos para la elaboración de circuitos impresos siendo sin lugar a duda el método fotográfico aquél con el cual se alcanza un nivel muy cercano al profesional, si nos ceñimos única y exclusivamente al diseño y realización de circuitos impresos “**Bicapas**” como máximo. Este método, no obstante, presenta algunos inconvenientes para el aficionado, pues obliga a poseer un cierto instrumental si se quieren alcanzar

resultados profesionales, a la vez que resulta un método no demasiado simple ni económico.

En los últimos años ha aparecido un nuevo método que presenta las ventajas de ser relativamente simple, económico y de muy buenos resultados si se realiza correctamente.

Este método consiste en transferir, mediante aplicación de calor, el "Tóner" de nuestro diseño sacado a partir de una fotocopiadora o, mejor, de una impresora láser, directamente en la superficie de cobre de una placa de circuito impreso virgen. La impresión ya sea mediante fotocopiadora o impresora láser se realiza sobre una lámina especial llamada PRESS-N-PEEL que presenta las características de poder resistir altas temperaturas sin llegar a deformarse y además presenta una capa rugosa donde lleva un determinado material para film que reacciona químicamente con el tóner depositado. El papel utilizado es papel fotográfico blanco brillante typo glossy de 140g./m2 referencia SO41126

Descripción del Proceso:

En primer lugar será necesario tener el diseño del cual se requiera realizar el circuito impreso. Es diseño puede ser realizado en casi cualquier software de simulaciones de Circuitos Integrados que permitan la exportación a PCB; en el caso se ha utilizado PCB Wizard para el diseño de las pistas de cobre. Es te Programa trabaja en conjunto con Limewire, que es el simulador; el diseño en Limewire puede ser exportado a PCB y este invoca PCB Wizard para realizar el diseño de las pistas, en el proceso se puede dar la elección de que el sistema por si mismo ofrezca un diseño en base a la colocación de los dispositivos sobre la placa teniendo en cuenta tamaño de la placa, tipo de materiales a colocar, etc. La otra elección permite que el usuario pueda

determinar tamaño de placas, tipos de dispositivos, grosor de las pistas > a 0.3mm, lo recomendable para el caso de usar el método de transferencia térmica es que las pistas sean de un mínimo de 1.0mm, con lo que garantizamos cualquier problema de impresión o que la pista sea muy delgada para ser impresa con la precisión necesarias. En caso de que se tratara de un diseño que no hemos realizado nosotros mismos o no tenemos el fichero en ningún formato, sino que sólo tenemos la imagen impresa deberemos en primer lugar escanear el diseño. Existe otra solución consistente en realizar una fotocopia del diseño directamente sobre el papel glossy, siempre y cuando la imagen que poseamos sea la correspondiente a la capa de cobre vista por "transparencia" desde la cara de componentes. En cualquier caso siempre es mejor utilizar una impresora láser para lo cual deberemos de poseer el fichero correspondiente al diseño en cualquier tipo de formato para poder realizar cualquier modificación o edición de éste.

Téngase en cuenta que al ser todo el proceso manual (atacado mediante percloruro de hierro, taladrado y estañado), es aconsejable proceder a un necesario retoque del diseño del PCB con anterioridad a poder imprimirlo en el papel glossy. Deberán de ensancharse las pistas menores de 0.3mm hasta 1.0mm., así como prestar una atención especial a los pads (para evitar dañarlos en el proceso de taladrado). Se aconseja que los centros correspondientes al posterior taladrado sean visibles y posean ya el diámetro ligeramente mayor al que se taladrarán con el fin de no erosionar inútilmente las brocas taladrando el cobre.

Una vez se tenga ya el diseño modificado, se puede utilizar Paint de Windows, (en este estadio podemos rotar y/o voltear nuestro diseño libremente), la mejor solución consiste en no imprimirlo directamente desde aquí, sino realizar un copiar/pegar hacia el Word. La razón es que aquí

podemos modificar las dimensiones del diseño y conseguir con éxito una impresión a escala 1:1, además de poder ajustar el contraste al 100%. También podemos aprovechar para "pegar" otros diseños, con lo que optimizaremos el papel glossy.

En el momento de imprimir mediante la impresora láser, configuraremos ésta para una impresión con la máxima resolución y contraste posible. Dependiendo del tipo de impresora los resultados pueden diferir mínimamente, pero en ningún modo afectará al resultado final. Se imprimirá en el papel glossy por la cara brillante de éste.

Una vez tengamos el diseño impreso en el papel glossy pasaremos seguidamente a preparar la placa de circuito impreso, para lo cual el primer paso (y el más importante) es limpiar cuidadosamente la cara de cobre sobre la cual se va a transferir el tóner del diseño. Para obtener mejores resultados se recomienda utilizar los llamados “Pastes para lavar platos”, estos son de color verde y superficie muy rugosa, pueden ser obtenidos con facilidad en las tiendas locales o los supermercados, su coste es muy bajo. Hay que proceder a limpiar enérgicamente la superficie cobreada y enjuagar generosamente con agua directamente del Chorro o toma de agua, hasta que ésta "resbale" por la superficie, hecho éste que nos indicará que la superficie está totalmente libre de grasa. A partir de este momento queda absolutamente prohibido tocar la superficie de cobre con las manos. Para proceder al secado total de la placa se utilizará un secador para el cabello, teniendo la precaución de no tocar en absoluto la cara de cobre.

Para realizar el proceso de transferencia del tóner a la placa se utilizará una plancha en la posición de temperatura correspondiente para "planchado de

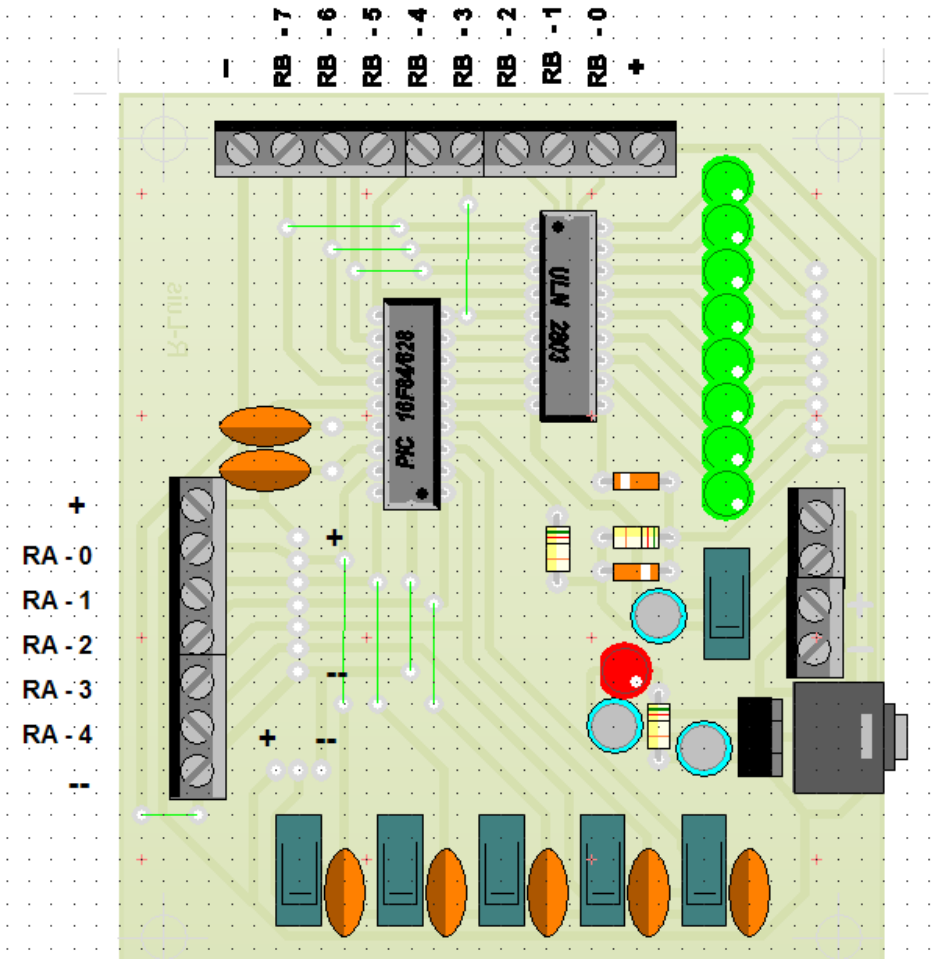
algodón" y sin vapor; así mismo puede también utilizarse una vieja plancha a la cual ya no le funcione la prestación de vapor.

Se recortara ahora el diseño impreso en el papel con unos márgenes aproximadamente de 1.0cm (centímetros) también de más. Posicionaremos el diseño con el tóner directamente en contacto con la superficie de cobre, tras lo cual presionaremos con los dedos en un determinado ángulo el papel mientras que por la zona opuesta acercaremos la plancha (que habrá alcanzado ya la temperatura correcta), presionando ligeramente para empezar a fundir ya el tóner con lo cual se irá adhiriendo al cobre. Continuaremos "planchando" toda la superficie presionando para que la adherencia del tóner sea perfecta. Continuaremos así entre 3 y 5 minutos, teniendo la precaución de presionar todas las zonas de los bordes y hacerlo con la parte central de la plancha (pues esta zona es la que está a la temperatura adecuada).

Terminado este proceso introduciremos rápidamente la placa de circuito impreso en una cubeta u otro recipiente que contenga agua fría. La dejaremos reposar un mínimo de unos veinte minutos, tras lo cual observaremos ya como el papel puede desprenderse de la placa de circuito impreso, si tiramos de él despegándolo levemente. Dejar la placa en el agua más tiempo ayuda a que el proceso de separación del papel sea más fácil.

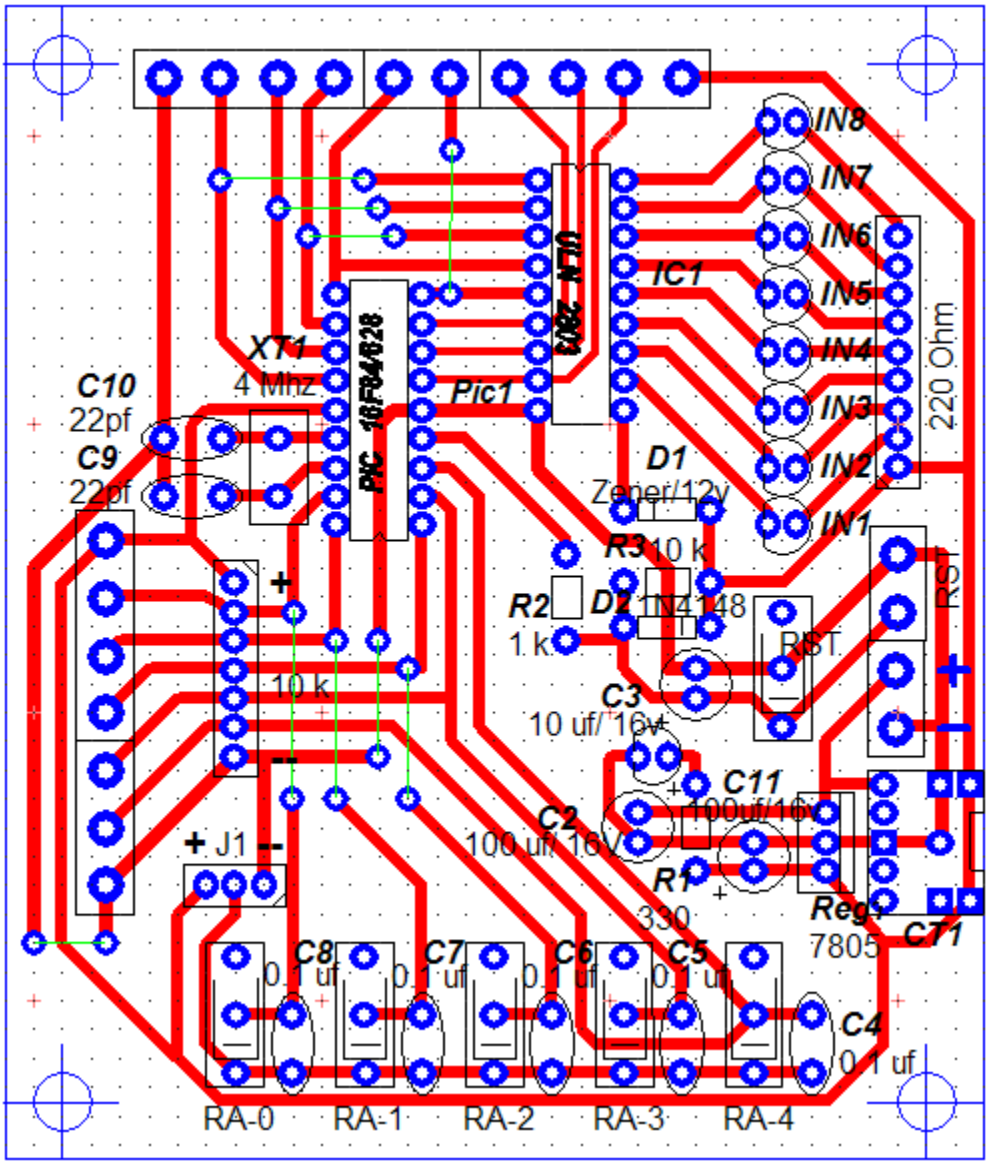
Una vez desprendido el papel se tendrá ya el diseño de la PCB. Llegados a esta fase el tóner adherido a la superficie de cobre resulta, mecánicamente hablando, muy resistente. Si se observa detenidamente el diseño, podrán apreciarse restos de fibras de papel y gelatina adheridos a la superficie del cobre. Si se cree oportuno, estos restos podrán eliminarse mediante un cepillo de dientes gastado, frotando muy suavemente teniendo la placa inmersa en el agua o bajo el grifo.

### *4.2.4.1.10. DISEÑO DE PLACA DE PROTOTIPO.*




**Diseño de Placa para Prototipo.**

### *4.2.4.1.11. DISEÑO DE IMPRESOS PARA PLACA COBREADA.*




**Diseño de Pistas con líneas guía de dispositivos.**

### *4.2.4.1.12. DISEÑO DE IMPRESOS EN PLACA DE COBRE*

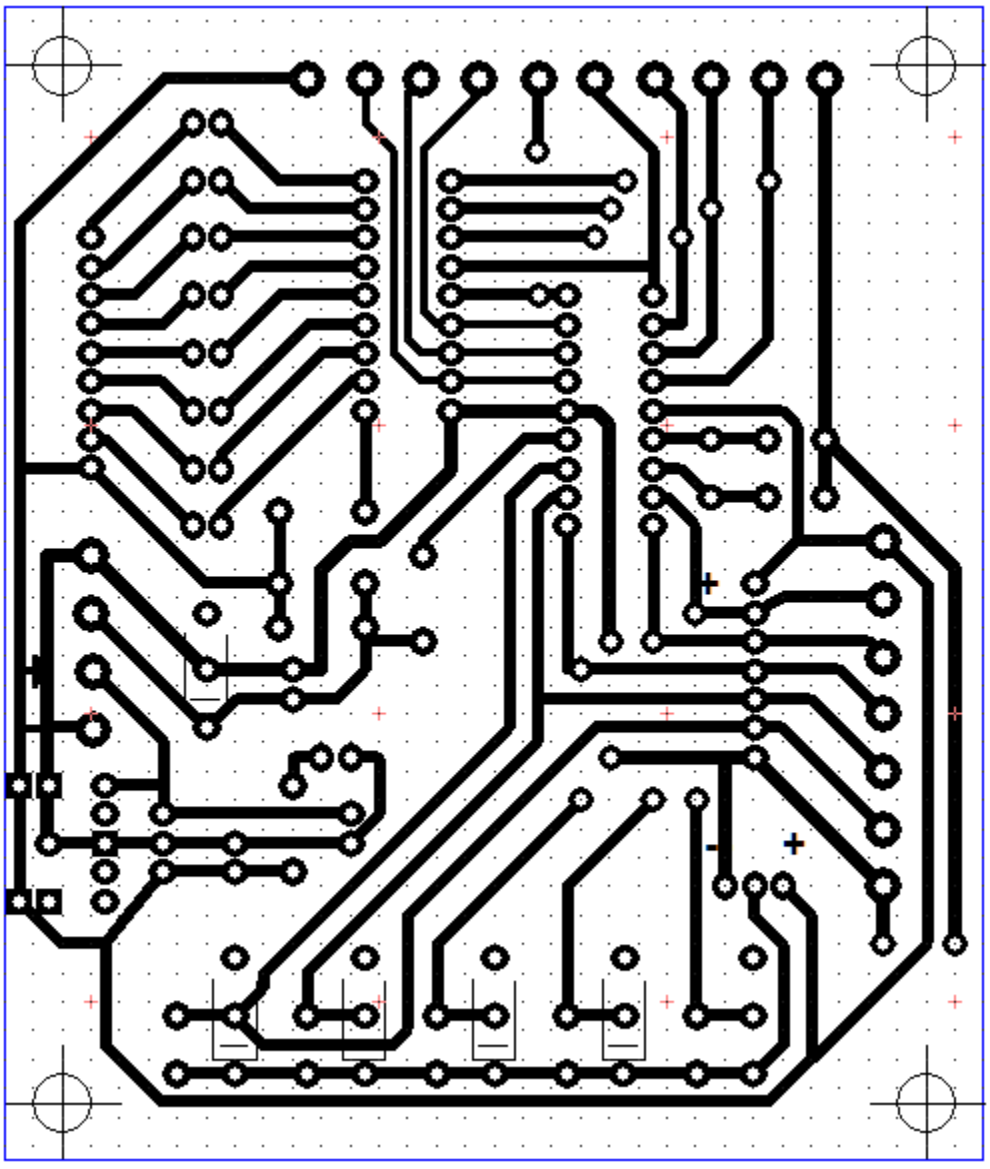


**Diagrama de Líneas a imprimir en placa cobreada**

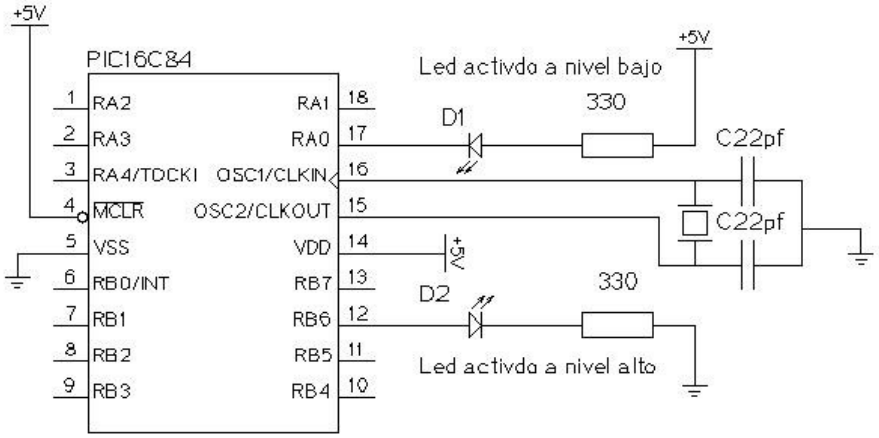
### *4.2.4.1.13. DISPOSITIVOS DE E/S PARA EL PIC 16F84*

#### *4.2.4.1.13.1. LEDS.*

Son dispositivos indicadores, son diodos emisores de luz muy baratos y se pueden conectar de dos formas:

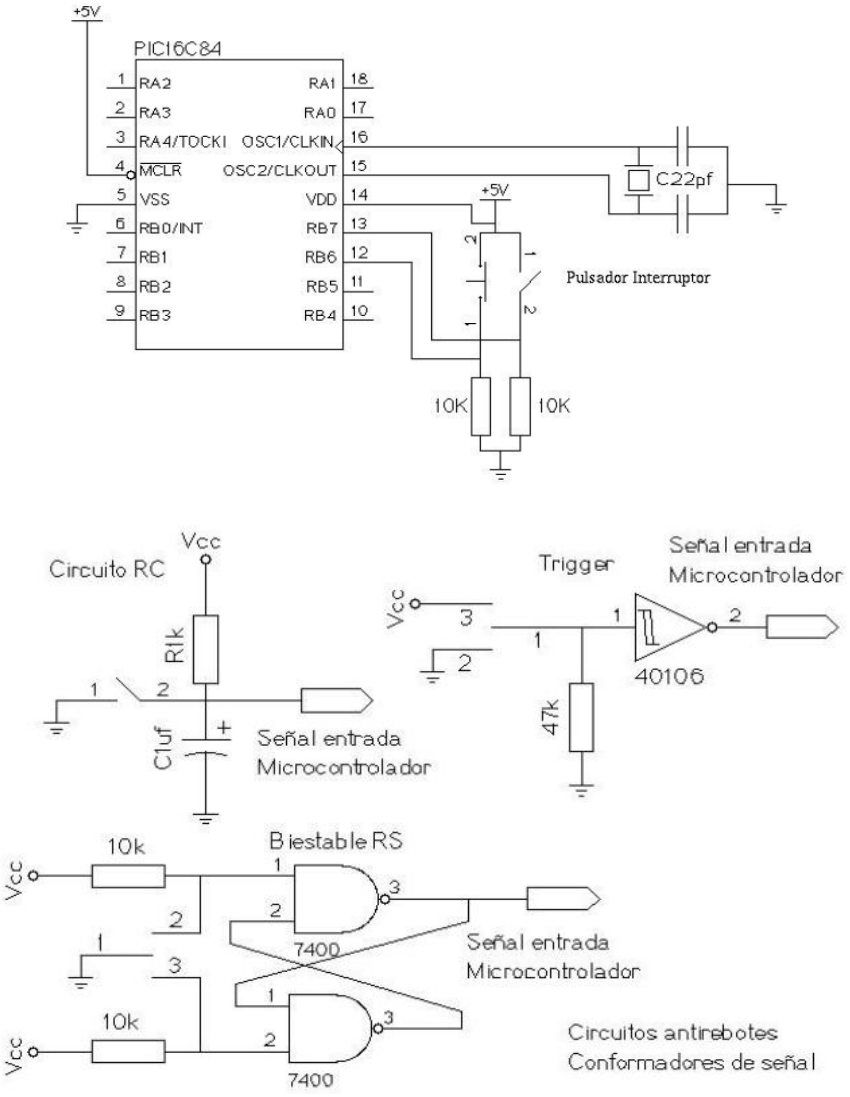
- Activados por nivel bajo

-Activados por nivel alto



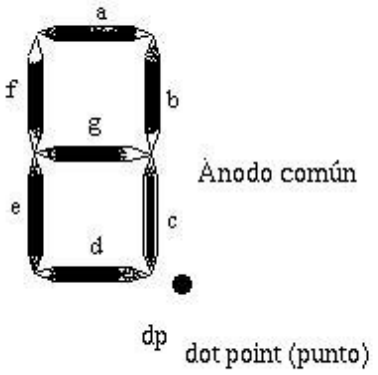

#### *4.2.4.1.13.2. PULSADORES E INTERRUPTORES*

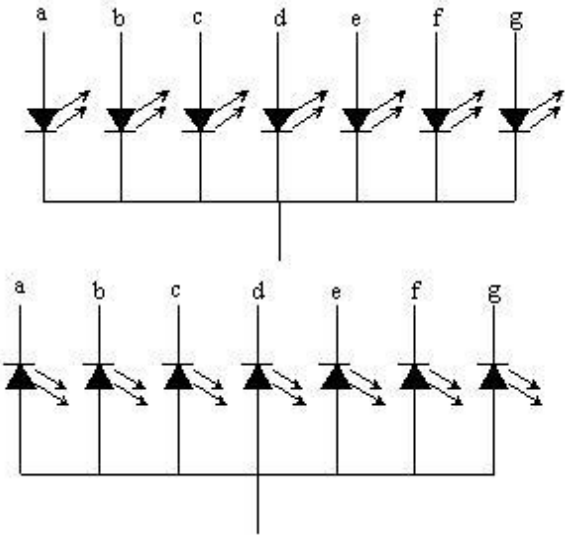
Son dispositivos que permiten la entrada de datos 1 / 0 según estén activados o no. El problema que pueden representar estos dispositivos pueden ser los rebotes, para evitarlos podemos realizar retardos por software o utilizar circuitos anti reboté hardware utilizando redes RC, circuitos biestables RS o puertas inversoras trigger.



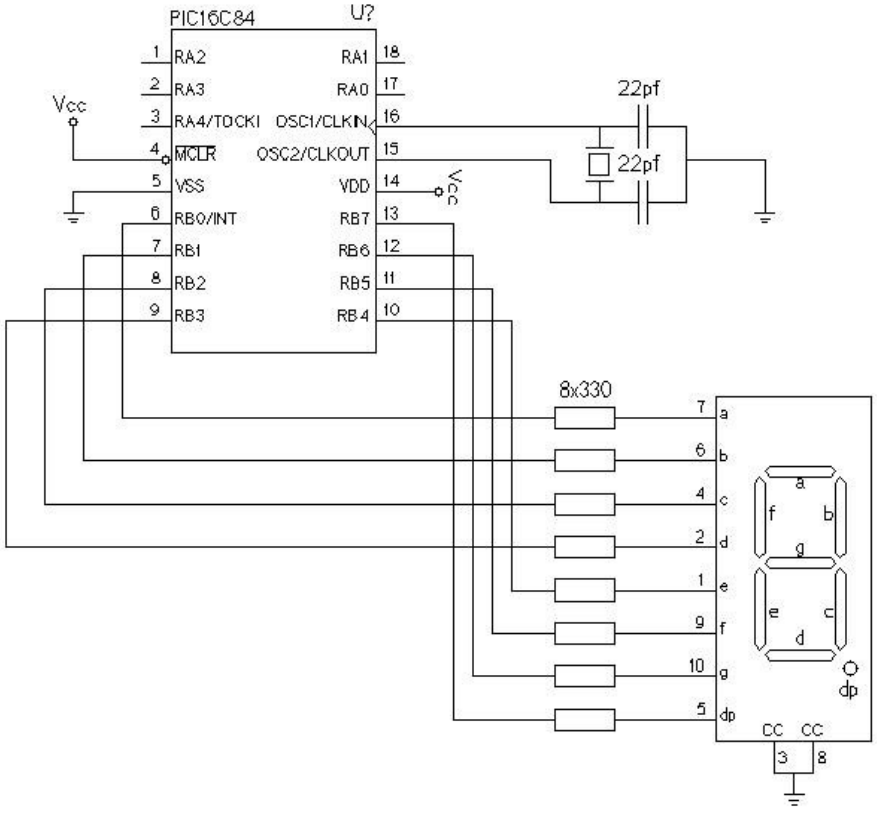

### *4.2.4.1.13.3. DISPLAYS 7 SEGMENTOS*

Permiten representar valores alfanuméricos mediante 8 leds. Generalmente tenemos displays de cátodo común pero existen también en ánodo común. El siguiente esquema representa la interconexión de un display en cátodo común. La siguiente tabla muestra los valores en hexadecimal para mostrar caracteres en un display 7 segmentos.



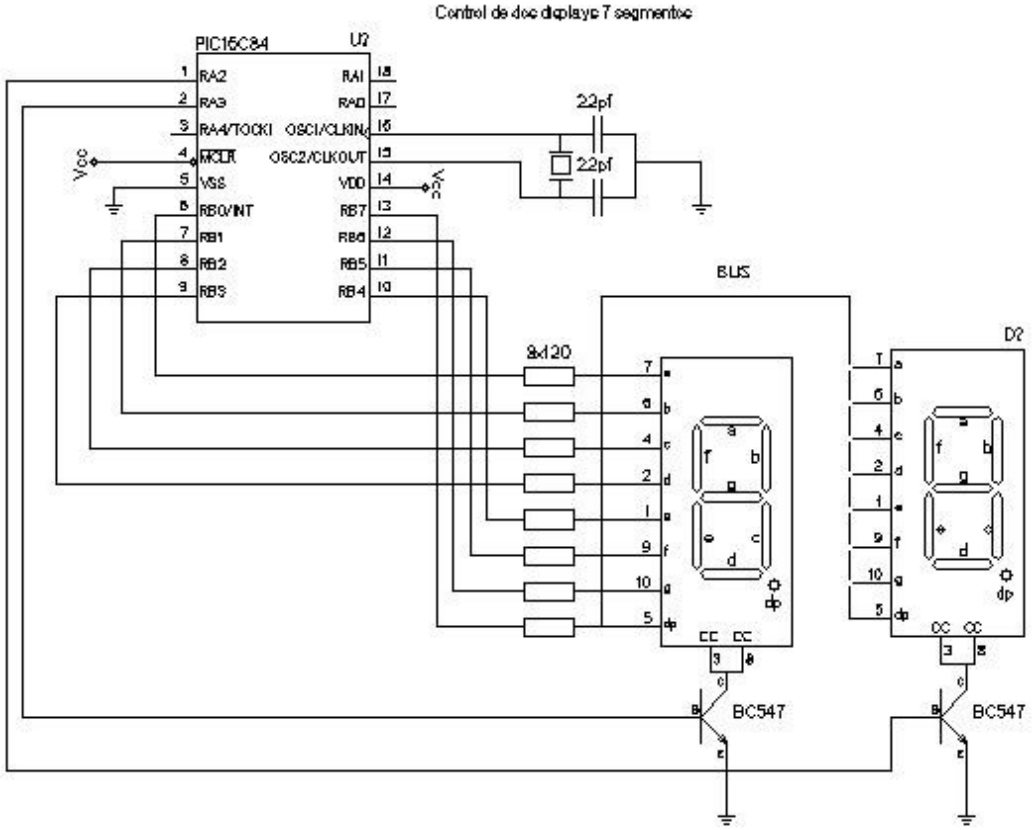

Control de un display 7 segmentos



**VALOR HEX CARÁCTER**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | **-** | 3F | **0** | 06 | **1** | 5B | **2** | 4F | **3** | 66 | **4** |
| 6D | **5** | 7D | **6** | 07 | **7** | 7F | **8** | 67 | **9** | 48 | **=** |
| 77 | **A** | 7C | **b** | 39 | **C** | 5E | **d** | 79 | **E** | 71 | **F** |
| 6F | **G** | 76 | **H** | 19 | **i** | 1E | **J** | 7A | **K** | 38 | **L** |
| 39 | **M** | 54 | **n** | 3F | **O** | 73 | **P** | 67 | **Q** | 50 | **r** |
| 6D | **S** | 78 | **t** | 1C | **u** | 3E | **U** | 6E | **Y** | 49 | **Z** |
| 55 | **Ñ** | 63 | **º** | 80 | **.** | | | | | | |

Podemos controlar varios displays utilizando transistores de corte saturación para controlar las patillas 3 y 8 de cada display; también se puede utilizar un driver SAA1064 de Philips para controlar más displays.



Se recomienda leer las notas técnicas y de aplicación de este circuito integrado.

| SYMBOL | PIN | DESCRIPTION |
|---|---|---|
| ADR | 1 | I²C-Bus slave address input |
| $C_{EXT}$ | 2 | external control |
| P8 to P1 | 3-10 | segment output |
| MX1 | 11 | multiplex output |
| $V_{EE}$ | 12 | ground |
| $V_{CC}$ | 13 | positive supply |
| MX2 | 14 | multiplex output |
| P9 to P16 | 15-22 | segment output |
| SDA | 23 | I²C-Bus serial data line |
| SCL | 24 | I²C-Bus serial clock line |

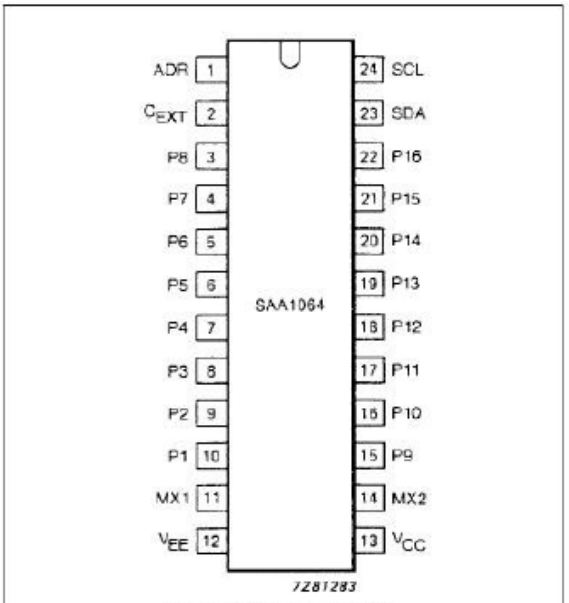
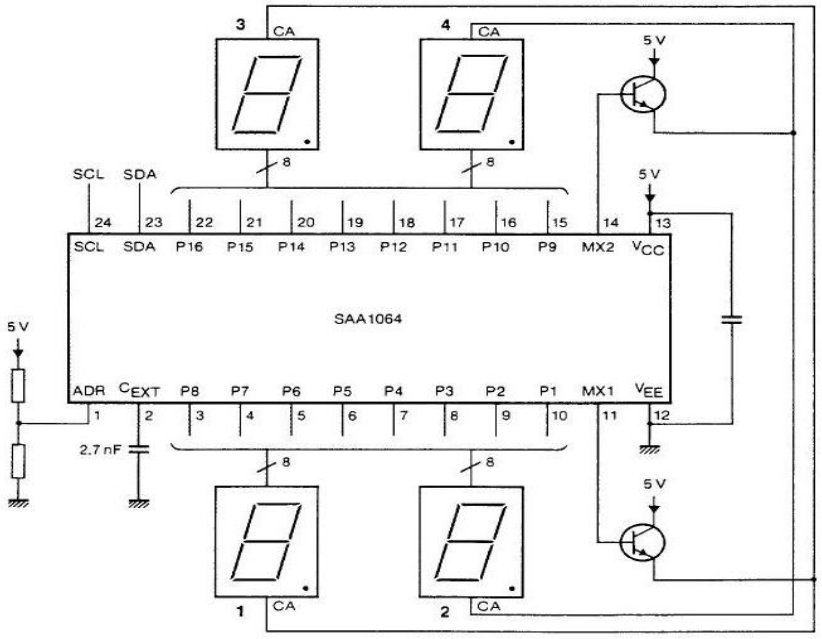


Fig.2 Pinning diagram.

## Diagrama de Conexiones



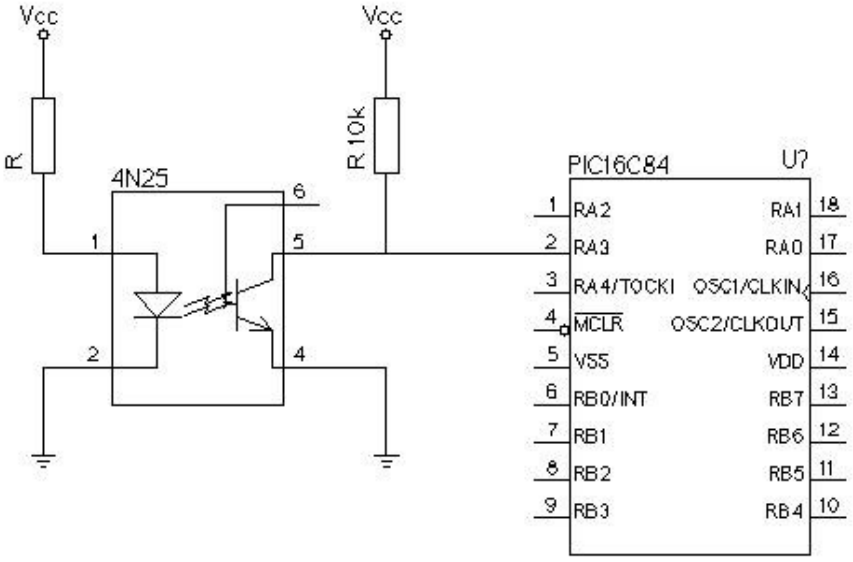
### *4.2.4.1.13.4. OPTO ACOPLADORES*

Siempre que se quiera aislar las entradas del PIC, podemos utilizar un opto acoplador como el 4N25 o similares.

Es un circuito integrado en cuyo interior hay un diodo emisor de luz y un fototransistor, cuando se aplica una señal al diodo, éste emite luz y hace que el transistor pase al modo de saturación y por tanto el transistor deja pasar corriente y tendremos un '0' en la entrada del PIC.

El cálculo de la resistencia es como sigue:

R= (Vin -1.2V)/ 5mA




### *4.2.4.1.13.5. RELÉS*

Los relés sirven para controlar cargas de mayor potencia a diferentes tensiones , incluyendo tensiones elevadas de CA.

Podemos encontrar micro relés para controlar pequeñas cargas (hasta 0.5 A) o relés de potencia.

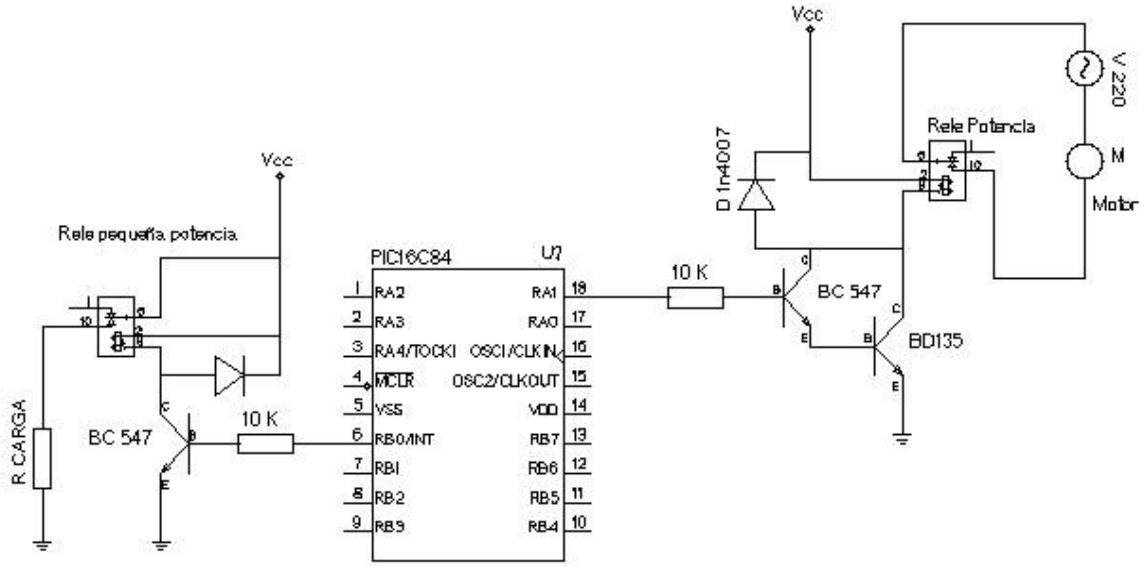
Los circuitos a implementar son los siguientes.

En RB0 hay conectado un micro relé de cc que controla una resistencia de carga.

En RA1 hay conectado un relé de CA de potencia que controla un motor de CA monofásico.

Es importante colocar los diodos en paralelo con la bobina de los relés para proteger el circuito de la fuerza electromotriz de la bobina en el momento de la conmutación.

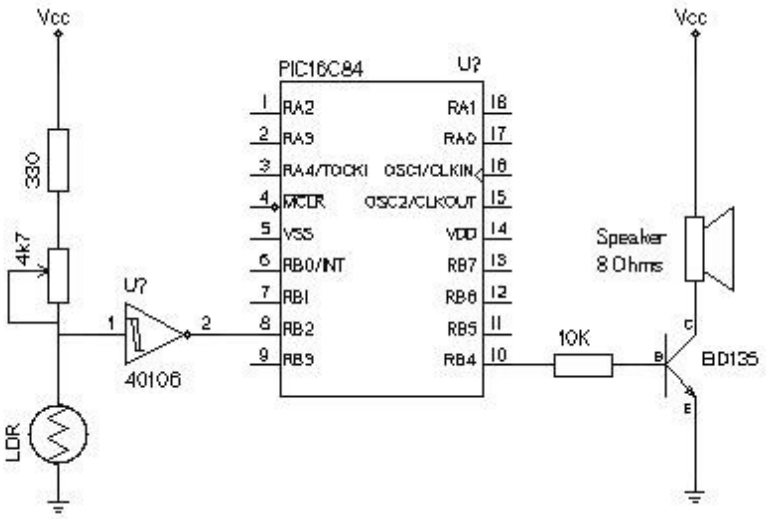
Diagrama con Relés.




### *4.2.4.1.13.6. OTROS DISPOSITIVOS*

Al micro controlador se pueden conectar infinidad de sensores y dispositivos de salida o de entrada como LDR's (Resistencias dependientes de la luz), zumbadores, sensores ópticos como el CNY70.

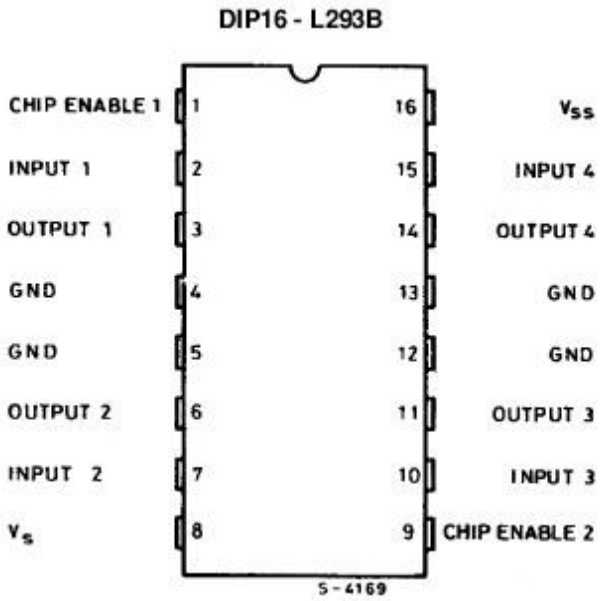
Otro tipo de sensores son los sensores de ultrasonidos, sensores de infrarrojos y dispositivos de entrada como bumpers etc.



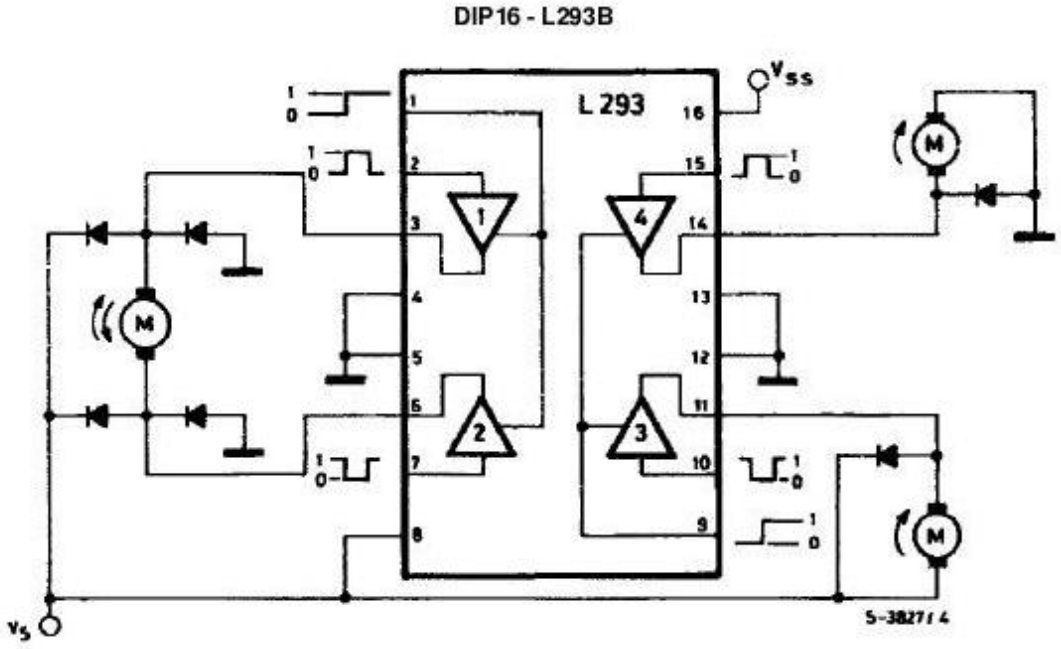

El circuito anterior tiene como entrada una LDR en RB2 y como salida en RB4 hay un pequeño amplificador para un altavoz.
La LDR está conectada mediante una puerta trigger inversora. Se puede utilizar como barrera para contar objetos y que cada vez suene un tono.

### *4.2.4.1.13.7. CONTROL DE MOTORES CC*

Para utilizar motores de cc o motores paso a paso necesitamos una corriente importante, podemos utilizar un puente en H de transistores, o podemos optar por una solución más compacta utilizando el driver L239B de STMicroelectronics.



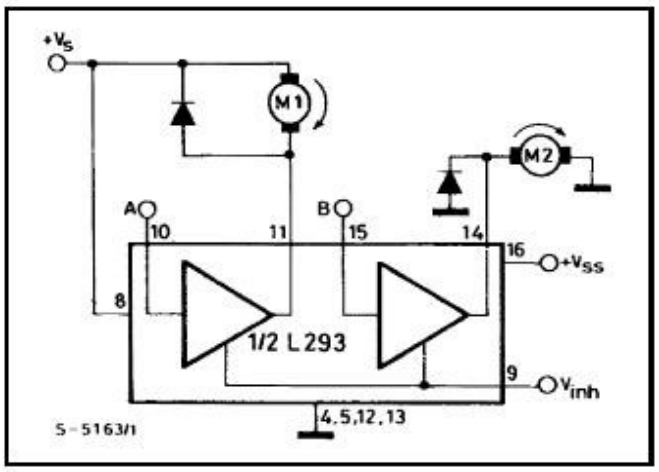

Patillaje del driver LB293B




Aplicación típica de control de motores (2 en un sentido y otro con posibilidad de inversión de giro)

Configuración típica del driver para el control de dos motores con un único sentido de giro

Las patillas de entrada son A (10) B (15) y Vinh




| $V_{inh}$ | A | M1 | B | M2 |
|---|---|---|---|---|
| H | H | Fast Motor Stop | H | Run |
| H | L | Run | L | Fast Motor Stop |
| L | X | Free Running | X | Free Runni |
| | | Motor Stop | | Motor Stop |

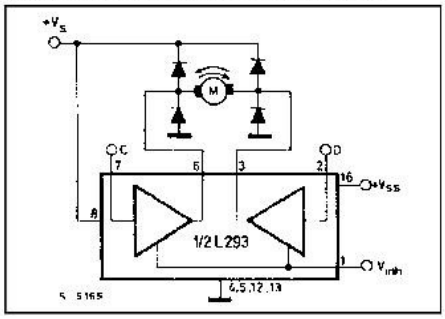
L = Low H = High X = Don't Care

El funcionamiento es como sigue:

| Vinh | A | B | Motor 1 | Motor 2 |
|---|---|---|---|---|
| 1 | 1 | 1 | Paro rápido | Funcionamiento |
| 1 | 0 | 0 | Funcionamiento | Paro |
| 0 | x | x | Motor giro | libre Motor giro libre |

Es importante proteger el circuito contra los picos de fuerza electromotriz generados en los devanados del motor en el momento de los cambios.

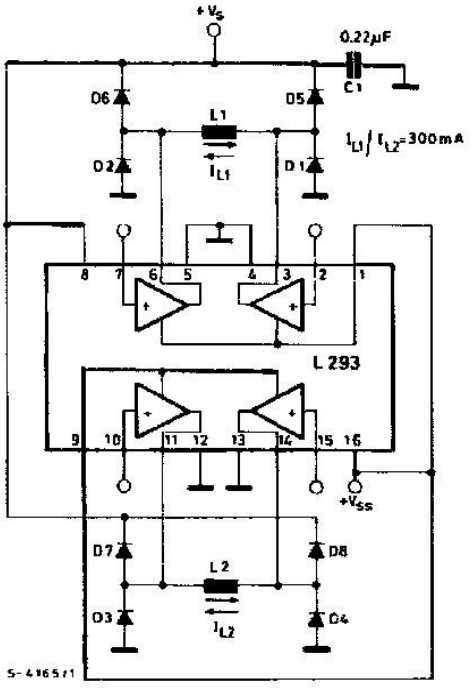
Control de un motor de forma bidireccional; éste circuito permite el control de giro en dos sentidos de un motor de CC




| Inputs | Function | |
|---|---|---|
| $V_{inh}$ = H | C = H ; D = L | Turn Right |
| | C = L ; D = H | Turn Left |
| | C = D | Fast Motor Stop |
| $V_{inh}$ = L | C = X ; D = X | Free Running Motor Stop |

L = Low H = High X = Don't Care

Hay que tener en cuenta la colocación de diodos de protección del circuito. Configuración del circuito para el control de un motor paso a paso bipolar.




### *4.2.5. SOFTWARE PARA APOYO DE PRÁCTICAS.*

Así como el hardware es una parte importante para el desarrollo de las prácticas lo es el software, ya que para poder seguir las guías de práctica adecuadamente necesitamos de estos dos elementos. A continuación se muestra una descripción del software a utilizarse para el trabajo en el laboratorio.

#### *4.2.5.1. LENGUAJE ENSAMBLADOR.*

El lenguaje ensamblador como sabemos es el utilizado para la programación de los micro controladores. Existen 5 partes principales en lenguaje ensamblador:

- Etiquetas
- Instrucciones
- Operandos
- Directivas
- Comentarios

Para la programación se utiliza una cierta tabulación que se debe respetar, además utilizar una tabulación adecuada hace a los programas más claros y legibles. Las etiquetas se escriben en la primera columna de cualquier línea, las instrucciones y directivas en la segunda, y por último, en la tercera columna los operandos. Los comentarios de pueden escribir en cualquier parte del programa.

#### *4.2.5.1.1. ETIQUETAS.*

Una etiqueta es una palabra utilizada para designar alguna línea o sección del programa, se pueden utilizar para saltar de una parte hacia esa etiqueta. Es importante que las etiquetas empiecen con una letra o con un guión bajo "_". La longitud de una etiqueta puede ser de hasta 32 caracteres y como ya se dijo se deben escribir en la primer columna.

#### *4.2.5.1.2. INSTRUCCIONES.*

Las instrucciones son las operaciones que realiza el Micro controlador, así que estas ya están definidas para cada familia de PIC. El 16F877A así como todos los PICs de gama media utiliza un conjunto de 35 instrucciones que están definidas en la hoja de datos del PIC.

#### *4.2.5.1.3. OPERANDOS.*

Son los elementos que emplea la instrucción que se está ejecutando. Usualmente los operandos son los registros, las variables o las constantes.

#### *4.2.5.1.4. DIRECTIVAS.*

Las directivas son similares a las instrucciones, pero a diferencia de estas, las directivas son propias del lenguaje ensamblador e independientes de Micro controlador que se utilice. Las directivas representan algunas características del lenguaje ensamblador, se utilizan para especificar el procesador empleado así como la configuración de este, también para asignar ubicaciones de memoria, entre otras cosas.

#### *4.2.5.1.5. COMENTARIOS.*

Los comentarios son palabras, frases y oraciones que se pueden escribir en el código para hacer el programa más claro y legible. Los comentarios de pueden escribir en cualquier parte del código pero siempre deben empezar con punto y coma “;”.

El siguiente ejemplo es un programa simple escrito en ensamblador donde se muestran los elementos básicos del lenguaje:

```
; Primer programa de prueba. Inicializa el Puerto B y pone todos los
; pines del puerto en un estado lógico "1"
; Fecha: 17.01.07   Autor: Jorge A. Bojórquez   micropic.wordpress.com

        list        p=16f628a       ; Declaración del procesador
        include     p16f628a.inc    ;
        __config    0x3F38          ; Declaración de la configuración

                                ; Inicio del programa
        org     0x00            ; Vector de Inicio
        goto    Inicio          ; Ir a la etiqueta 'Inicio'

Inicio  bsf     STATUS,RP0      ; Seleccionar el banco de memoria 1
        clrf    PORTB           ; Configurar puerto B como salida
        bcf     STATUS,RP0      ; Seleccionar el banco de memoria 0

        movlw   0xFF            ; Cargar al acumulador W el valor 0xFF
        movwf   PORTB           ; Pone todos los pines del Puerto B en "1"

Ciclo   goto    Ciclo

        end
```

Figura 4.4 Ejemplo de programa en lenguaje Ensamblador

El software ensamblador que presenta Microchip es un compilador llamado MPASM. Soporta a TODOS los micro controladores de la familia PIC de Microchip. El conjunto de instrucciones de los micro controladores PIC es en esencia la base del lenguaje ensamblador soportado por este software.

Estas son algunas instrucciones para el compilador.

#DEFINE

Ej.  #define <nombre>  [<valor a remplazar>]

Explicación: declara una cadena de texto como substituto de otra

END

Ej.  End

Explicación: indica fin de programa

EQU

Ej.  Status equ 05

Explicación: define una constante de ensamble

INCLUDE

Ej. include <PIC16F84.h>

Explicación: incluye en el programa un archivo con código fuente

ORG

Ej.  Org 0x100

Explicación: ensambla a partir de la dirección especificada

La información anterior es un breve resumen acerca de este lenguaje, para una referencia completa en la sección de anexos (ó CD) encontrara un completo manual.

#### *4.2.5.2. EL MPLAB.*

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

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

MPLAB se pueden obtener de forma gratuita en la página web www.microchip.com, en la cual se encuentra una amplia información sobre todos los dispositivos que fabrica Microchip.

El MPLAB es la herramienta correcta para el desarrollo de aplicaciones en lenguaje ensamblador, como ya se ha mencionado, éste posee un simulador; el cual es capaz de mostrar el cambio en los registros y el avance del programa, pero no entra en la categoría de simulador gráfico, ya que no muestra un esquema del circuito, por lo tanto no le es posible mostrar una

simulación gráfica de este. Para las simulaciones gráficas existen otras aplicaciones las cuales se desvían de los objetivos del proyecto, pero de igual manera sugerimos una muy potente e útil para la simulación de circuitos electrónicos.

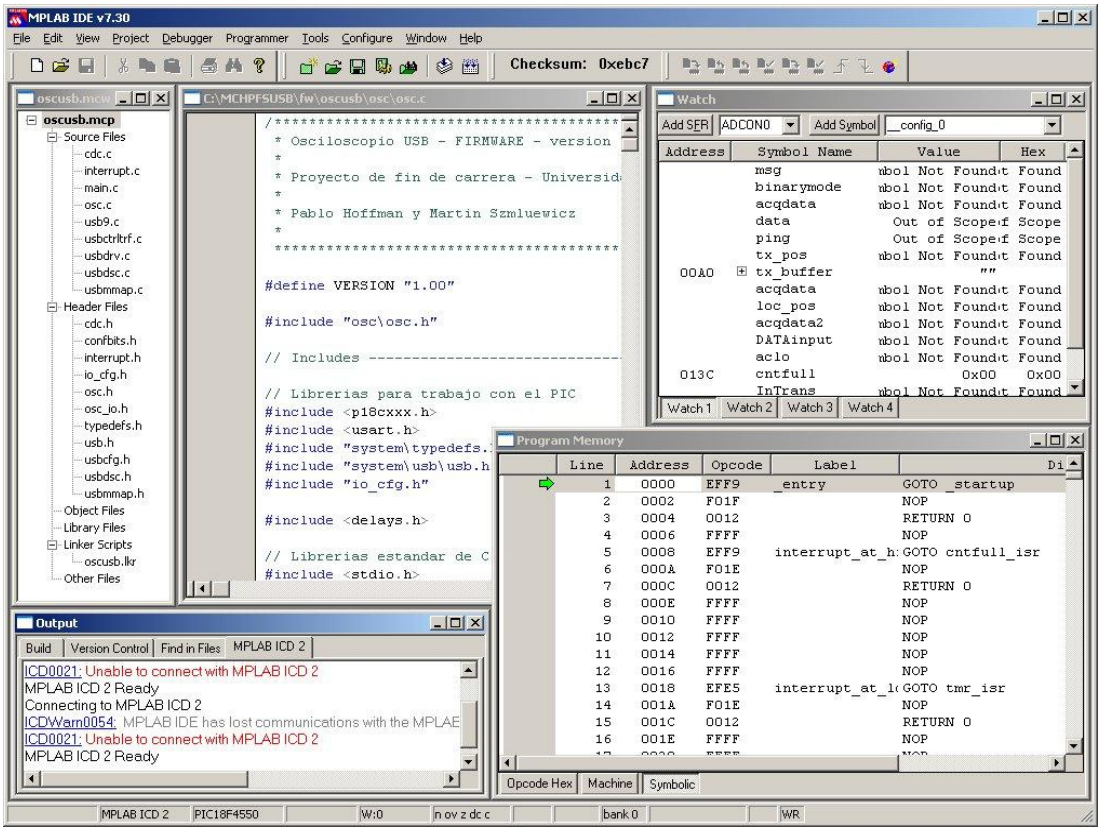



Figura 4.5 Entorno de desarrollo MPLAB

Los requerimientos mínimos para la instalación de MPLAB son:

Procesador 386, 486 o Pentium*

Windows 95/ 98/2K/XP, MACOS 7.0 y superiores, o Unix.

16 MB de memoria RAM para sistema.

45 MB de espacio libre en disco duro.

Se recomienda por Microchip Technology:

Procesador Pentium

32 MB de memoria RAM

En la sección de anexos podrá encontrar un manual extenso y detallado de MPLAB, ofreciendo de esta manera una herramienta más para facilitar al alumno el aprendizaje y permitiéndole avanzar en el estudio de los Micro controladores PIC de Microchip.

#### *4.2.5.3. WINPIC800*

WinPic800 es un software diseñado para programar Micro controladores, usando un dispositivo grabador de chips el cual puede ser conectado por medio de un puerto serial, paralelo o USB. WinPic800 soporta una gran variedad de Micro controladores PIC, así como también muchos equipos grabadores. WinPic800 posee una sencilla interfaz gráfica la cual nos permite visualizar el programa (en Hex) que deseamos introducir en el Micro controlador, de la misma forma nos permite ver la información contenida dentro de un Micro controlador grabado previamente. WinPic800 es freeware, por lo tanto es gratuito y es posible distribuirlo sin ningún inconveniente.

WinPIC800 es una herramienta intuitiva la cual nos permitirá enviar la aplicación que hemos diseñado en MPLAB hacia el micro controlador por medio del hardware programador. WinPIC800 no presenta problema al integrarse a las demás herramientas (Programador JDM, MPLAB) utilizadas para el desarrollo de las practicas con Micro controladores y además trabaja perfectamente con el PIC16F84. Es por esta razón que lo hemos elegido como el software programador para las prácticas con Micro controladores.

En la sección de anexos es expone la forma adecuada de configurar y utilizar el WinPIC800.

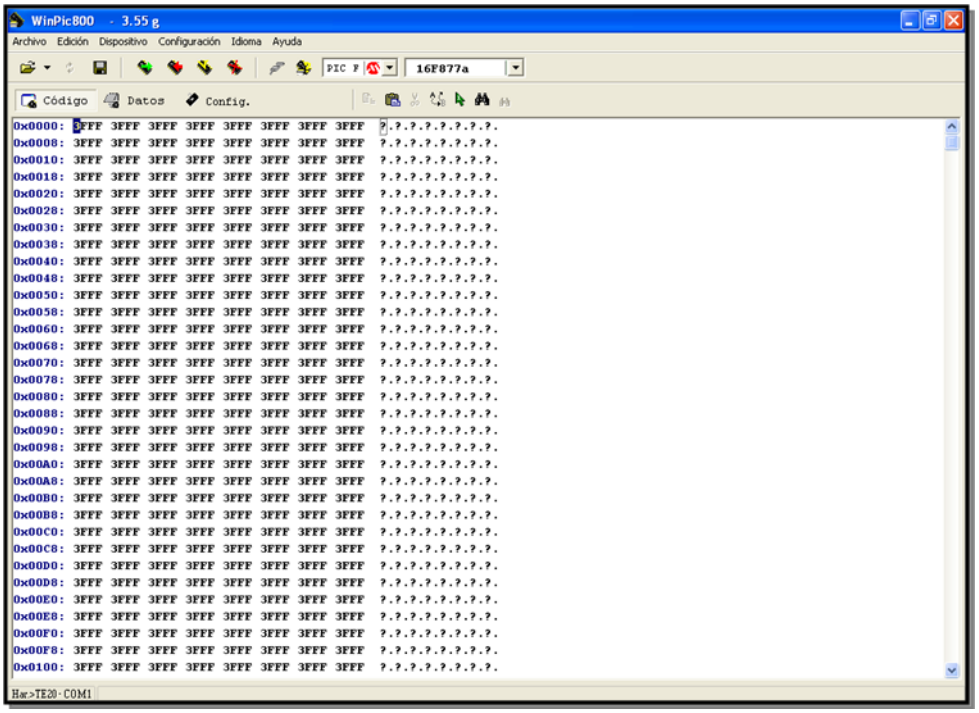



Figura 4.6 Entorno de Programación para el WINPIC

#### *4.2.5.4. PROTEUS.*

Proteos es una sugerencia que resulta muy útil en el trabajo con simulaciones.

Es un entorno integrado diseñado para la realización completa de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño, simulación, depuración y construcción. La suite se compone de cuatro elementos, perfectamente integrados entre sí:

##### *4.2.5.4.1. ISIS*

La herramienta para la elaboración avanzada de esquemas electrónicos, que incorpora una librería de más de 6.000 modelos de dispositivos digitales y analógicos.

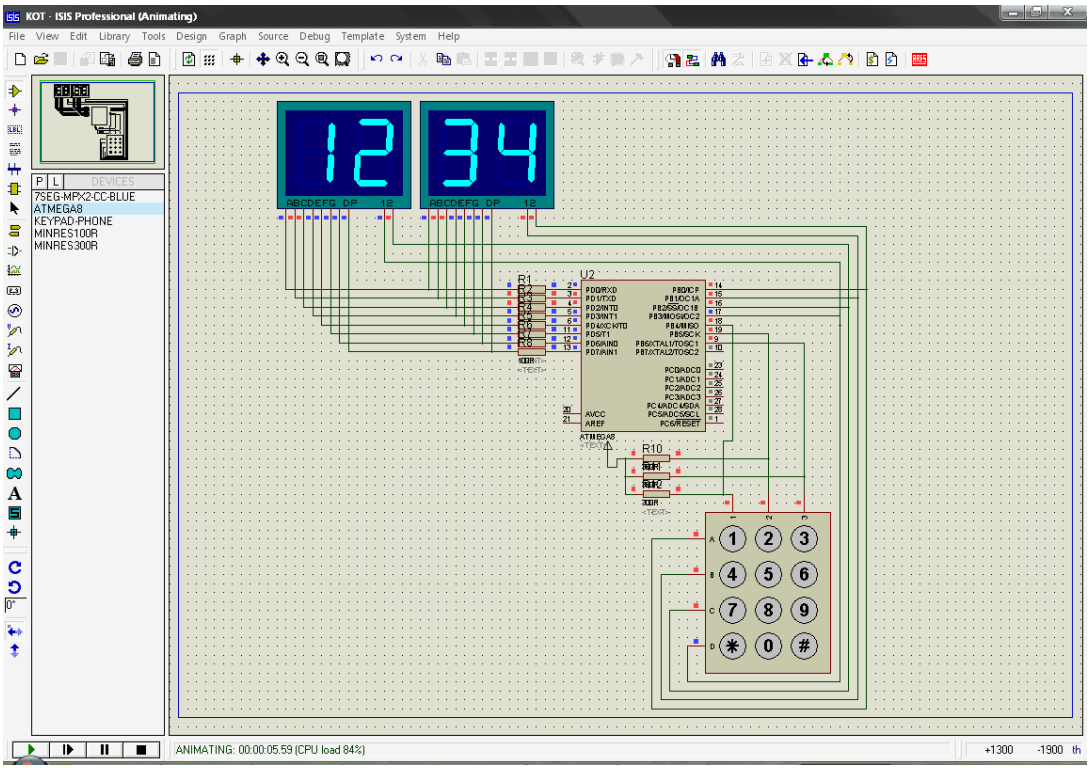



Figura 4.7 Entorno de simulación y Diseño para Proteus

#### *4.2.5.4.2. ARES*

La herramienta para la elaboración de placas de circuito impreso con posicionador automático de elementos y generación automática de pistas, que permite el uso de hasta 16 capas. Con ARES el trabajo duro de la realización de placas electrónicas recae sobre el PC en lugar de sobre el diseñador.

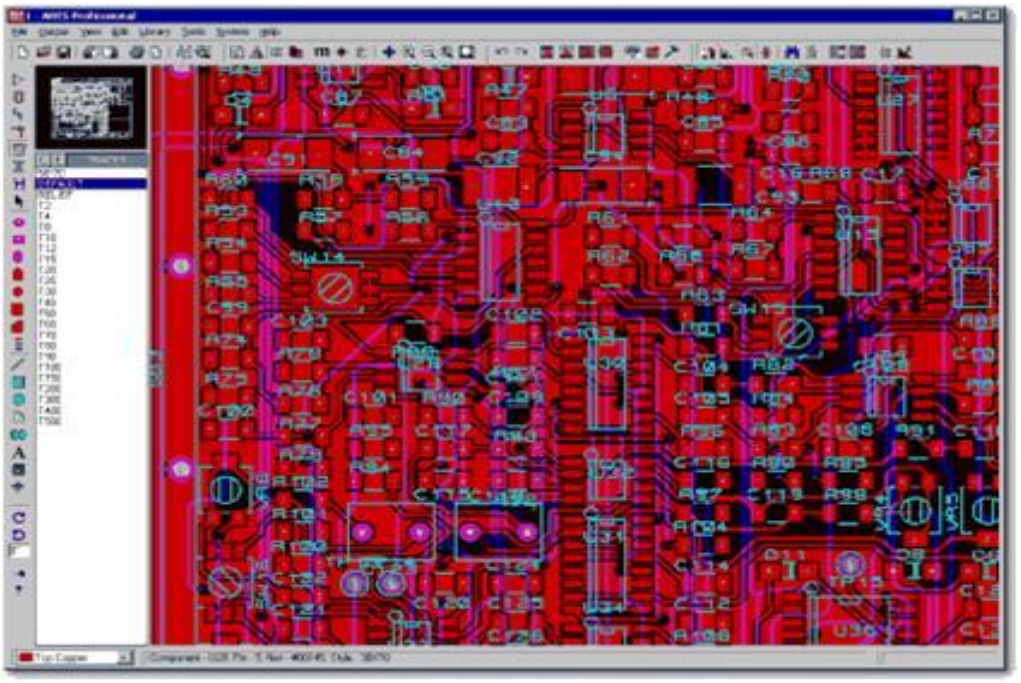
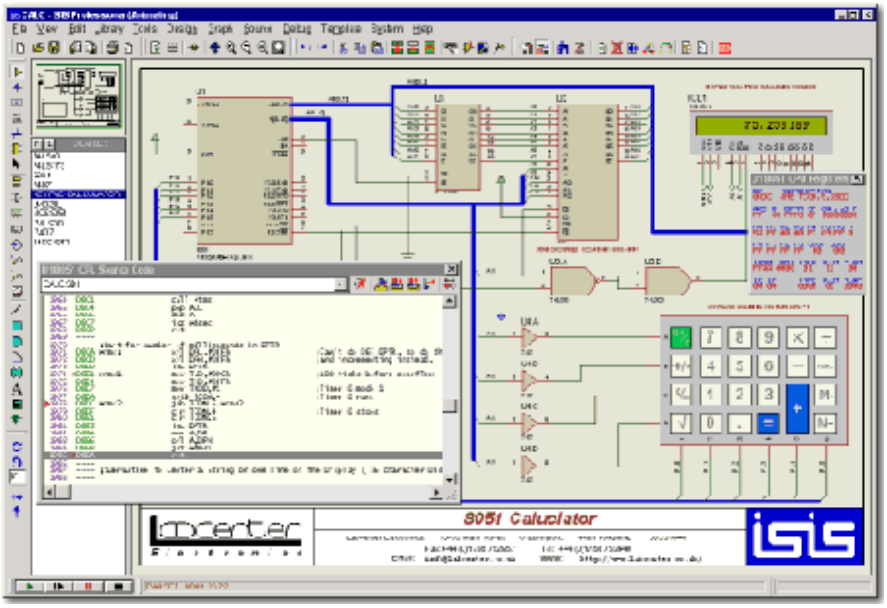


Figura 4.8 Entorno para diseño de Placa Impresa en ARES

#### *4.2.5.4.3. VSM*

La revolucionaria herramienta que permite incluir en la simulación de circuitos el comportamiento completo de los Micro controladores más conocidos del mercado. PROTEUS es capaz de leer los ficheros con el código ensamblado para los microprocesadores de las familias PIC, AVR, 8051, HC11, ARM/LPC200 y BASIC STAMP y simular perfectamente su comportamiento. Incluso puede ver su propio código interactuar en tiempo real con su propio hardware pudiendo usar modelos de periféricos animados tales como displays Leds o LCD, teclados, terminales RS232, simuladores de protocolos I2C, etc. Proteus es capaz de trabajar con los principales compiladores y ensambladores del mercado.




Las principales características de Proteus son:

- Entorno de diseño gráfico de esquemas electrónicos (ISIS) extremadamente fácil de utilizar y dotado de poderosas herramientas para facilitar el trabajo del diseñador.

- Entorno de diseño de placas de circuito impreso (ARES) de ultra-altas prestaciones con bases de datos de 32 bits, posicionador automático de elementos y generación automática de pistas con tecnologías de auto corte y regeneración.

- La mayor parte de los módulos que componen PROTEUS han sido escritos por el mismo equipo, garantizando al máximo nivel posible la compatibilidad e inter-operatividad de todas las herramientas que componen el entorno PROTEUS.

- Ejecutable en los diferentes entornos Windows: 98, Me, 2000, XP.

- Herramienta de máximas prestaciones, basadas en los más de 15 años de continuo desarrollo y presencia en el mercado.
- Miles de instalaciones vendidas en más de 35 países a todo lo largo del mundo.
- Se presenta un manual de usuario en la sección de anexos, el cual permitirá a los interesados poder trabajar fácilmente con esta aplicación.

## *4.3. PRECAUCIONES Y RECOMENDACIONES EN LAS PRÁCTICAS*

### *4.3.1. INTRODUCCIÓN*

A continuación presentamos recomendaciones cuyo objetivo es proveer las condiciones adecuadas para el desarrollo de las prácticas de laboratorio con micro controladores para la asignatura de Microprogramación y así evitar accidentes o posibles daños en el equipo por el uso erróneo del equipo y componentes electrónicos, así como por tomar en cuenta las condiciones ambientales de las instalaciones donde se llevaran dichas practicas. Es de hacer notar que son recomendaciones y que en el momento de la implementación pueden ser modificadas por el Departamento de Ingeniería y Arquitectura.

### *4.3.2. PRECAUCIONES GENERALES*

Como se trata de dispositivos electrónicos, se debe tener ciertas consideraciones básicas para manipularlos y cumplirlas estrictamente.

- Antes de manipular cualquier objeto que se conecte a la electricidad, debe desenchufarlo, sin excepción.
- Actuar en el puesto de trabajo respetando las normas de seguridad personal y de los medios y materiales utilizados en el desempeño de las actividades.
- Los manuales de cada dispositivo suelen traer las recomendaciones y métodos para utilizarlo de la manera más adecuadas, esas instrucciones tienen prioridad ante cualquier recomendación ajena.
- Algunos dispositivos son muy sensibles, nunca sea brusco.
- Aparatos de aire comprimido son muy útiles para la limpieza de polvo para algunos dispositivos, siempre limpian mejor que los tradicionales soplidos.
- Los trapos anti-estáticos son geniales para atraer polvo.
- Para proporcionar cierto grado de protección antiestática utilice una muñequera antiestática conectada a la tierra del chasis cualquier superficie metálica sin pintura cuando manipule componentes.

- Desconectar las aplicaciones cuando no está requerido para el uso.
- Mantener todos los alambres, enchufes y equipo condiciones seguras.

### *4.3.3. CONDICIONES RECOMENDADAS PARA OPERAR*

#### *4.3.3.1. ELIJA UN SITIO QUE SEA:*

- Limpio y sin partículas en el aire (que no sea el polvo normal existente en el ambiente).
- Bien ventilado y lejos de fuentes de calor, incluyendo la luz del sol.
- Lejos de fuentes de vibración o impactos físicos.
- Aislado de campos electromagnéticos fuertes producidos por dispositivos eléctricos.
- En áreas susceptibles a tormentas eléctricas, se recomienda conectar el sistema a un supresor de sobre corriente/sobre voltaje.
- Provisto de un tomacorriente de pared con una conexión a tierra apropiada.
- No exponga el dispositivo a temperaturas extremas, ya que esto puede suponer la destrucción o daño del equipo o material.

### *4.3.3.2. PROTECCIÓN AL USUARIO Y AL TRABAJADOR*

A objeto de proteger al usuario y al trabajador se deben observarse todos los reglamentos y normas ordinarias en el funcionamiento de los diferentes equipos y maquinarias a utilizar en las distintas actividades.

- Se identifican los derechos y deberes del empleado y de la organización en materia de seguridad laboral.
- Se identifican los equipos y medios de seguridad más adecuados para cada actuación y su uso y cuidado es el correcto.
- Se vigila el cumplimiento de las normas de seguridad laboral, creando el ambiente necesario para su mantenimiento.
- Las zonas de trabajo de su responsabilidad permanecen en condiciones de limpieza, orden y seguridad.
- Se toman las medidas oportunas, y se avisa a quien corresponda, ante una situación de emergencia.
- Las precauciones de conexión y manipulación del equipo se especifican con claridad.
- Se informa debidamente a otras instancias, de la emergencia ocurrida, y en su caso se analizan las causas, proponiendo las medidas oportunas para evitar su repetición.

- Los equipos y medios utilizados para el mantenimiento de los equipos electrónicos y para el diagnóstico y localización de averías están calibrados y conservados convenientemente, aplicando los procedimientos normalizados.

- El diagnóstico y localización de la avería del equipo se realiza mediante la consulta de la documentación técnica del mismo, la utilización de las herramientas y los instrumentos de medida apropiados, aplicando el correspondiente procedimiento sistemático, en un tiempo adecuado.

- Las pruebas funcionales realizadas inicialmente permiten verificar los síntomas recogidos en el parte de averías y, en todo caso, precisar la sintomatología de la disfunción en el equipo.

- La hipótesis de partida y el plan de actuación elaborado permiten diagnosticar y localizar con precisión el tipo (mecánico y/o eléctrico) y el bloque funcional donde se encuentra la avería.

- Las operaciones de montaje, desmontaje y sustitución de componentes electrónicos (soldadura y desoldadura) de las tarjetas de circuito impreso se realizan mediante la utilización de componentes similares o equivalentes y con las herramientas apropiadas, aplicando los procedimientos normalizados y asegurando un buen contacto eléctrico y sujeción mecánica.

- Los ajustes de los subsistemas mecánicos de los equipos electrónicos se realizan mediante la utilización de herramientas y útiles específicos, con la precisión requerida, siguiendo los procedimientos documentados.

- La reparación del equipo se realiza respetando las normas de seguridad personal, de los equipos y materiales recomendadas en la documentación de los mismos y, en todo caso, siguiendo las pautas del buen hacer profesional.

- El informe de reparación de averías del equipo electrónico se realiza en el formato normalizado, recogiendo la información suficiente para realizar la facturación de la intervención y actualización del "Histórico" de averías del equipo.

#### *4.3.3.3. DE LOS AUXILIARES DE LABORATORIO*

- Al inicio de sus actividades el Auxiliar de laboratorio debe asistir a los estudiantes en sus inquietudes sobre ubicación de materiales y equipos, así como en las demás normas de seguridad dentro de laboratorio.

- Preparar el material y equipos solicitados para la realización de la práctica, siempre y cuando la requisición sea entregada con ocho (8) días de anticipación por los docentes de la ÚES-FMO.

- Verificar que se dé el uso adecuado del equipo, aparatos y material del laboratorio durante el desarrollo de la práctica.

- Apoyo a los alumnos en la utilización y manejo de equipo con seguridad, a fin de evitar accidentes o daños al mismo dentro del laboratorio.

- Podrá llamar la atención a los alumnos cuando éstos se encuentren haciendo mal uso del equipo de laboratorio e inclusive solicitar que abandonen el mismo cuando no tengan actividades académicas que realizar.

- Serán responsables de verificar el estado físico y buen funcionamiento de los equipos antes de que estos sean entregados a los alumnos y después de que se haya concluido la práctica, asegurándose que le sea devuelto en las mismas condiciones físicas y de funcionamiento en que se le entregó.

- Serán los responsables de los tratamientos especiales de limpieza que se requieran para el material y equipo proveniente de su uso en prácticas de laboratorio.

- No podrá abandonar el laboratorio durante el desarrollo de la práctica, sin antes dar aviso al profesor del laboratorio.

- En caso de no poder asistir a los laboratorios o tener que ausentarse durante el desarrollo de una práctica por razones externas deberán avisar con la mayor antelación posible, evitando hasta donde sea posible la solicitud de permisos durante el desarrollo de prácticas de laboratorio.

- Reportar el mantenimiento preventivo y correctivo de las instalaciones y equipos de laboratorio.

### *4.3.3.4. DE LOS PROFESORES.*

- Proporcionar al inicio del semestre, el manual de prácticas a realizar, así como verificar el calendario de las mismas a los estudiantes de acuerdo al Instructivo anual de la ÚES-FMO y al Auxiliar o persona encargada de atender el servicio en la entidad respectiva, con el tiempo de anticipación que estipulen.

- Entregar al Auxiliar o persona encargada de atender el servicio en la entidad respectiva su requisición de equipos y material de laboratorio por lo menos con 8 días de anticipación.

- Deberá conocer el uso de los aparatos y equipo que se requiera en la práctica para poder verificar que se dé el uso adecuado a éste y así evitar su deterioro.

- En caso de que el profesor no pueda asistir al laboratorio, se suspenderá la práctica. El profesor no podrá por ninguna razón abandonar el laboratorio durante el desarrollo de la práctica por más de 15 minutos.

- El Docente de cada clase será el responsable del uso adecuado del material asignado a los alumnos para realizar las prácticas en los laboratorios.

- Adicionar como información a la Entidad que prestará el servicio a) horarios en que se utilizarán los laboratorios b) nombre de los alumnos autorizados para trabajar en los laboratorios c) actividades que realizarán así como los materiales y equipos que serán utilizados.

- El profesor será totalmente responsable de la integridad física de los estudiantes y de los daños que se ocasionen a las instalaciones, aparatos y otros objetos del laboratorio durante el desarrollo de sus actividades en el laboratorio. Por lo tanto el profesor deberá asegurarse que sus estudiantes tienen experiencia en las labores que desempeñarán, que conocen las medidas de seguridad y precauciones que deben tener durante el desarrollo de sus actividades así como de sus reacciones en situaciones de emergencia.

#### *4.3.3.5. DE LOS ALUMNOS*

- Previo a su primera clase de laboratorio deberán conocer las medidas de seguridad y aprenderán el uso del equipo y elementos de protección disponibles en el laboratorio.
- Deben asistir al laboratorio en los horarios que correspondan a su sesión de laboratorio con los implementos de limpieza y seguridad necesarios de acuerdo al tipo de laboratorio y práctica a realizar.
- Es responsabilidad del alumno conocer y usar el equipo de seguridad necesario para el desarrollo de cada práctica.
- Para el préstamo de material y equipo de laboratorios necesario deberá presentar su carné vigente que lo acredite como alumno regular debidamente registrado en el listado enviado con anterioridad.

- Revisará el material, la cantidad y el estado físico, así como también el equipo que utilizará en la práctica, el cual debe estar en buenas condiciones.

- El material que se rompa o deteriore estando en poder los alumnos, deberá ser repuesto por otro de las mismas características a mas tardar al final del semestre académico de lo contrario no podrá presentar las evaluaciones finales correspondientes.

- Al final de la práctica entregar el material limpio y libre de marcas y etiquetas y reportar si hubo alguna anomalía.

- Al retirarse del laboratorio deberá dejar su área de trabajo en orden.

- Podrán permanecer en el laboratorio siempre y cuando estén dentro de sus horarios estipulados y con el consentimiento explícito del profesor responsable.

- No deberá fumar ni ingerir alimentos y bebidas dentro del laboratorio.

- Deberá guardar respeto y seguir las indicaciones del profesor y del Auxiliar responsable del área.

### *4.3.3.6. DEL FUNCIONAMIENTO DE LOS LABORATORIOS*

- Para llevar a cabo cualquier actividad dentro de los laboratorios, se debe llenar una requisición especificando el material y equipos necesarios para la realización de sus prácticas mínimo una semana antes de la fecha programada, con el propósito de preparar el material y equipo solicitados y en caso de no haber algún material o equipo, el profesor junto con la entidad que presta el servicio tenga tiempo de sustituirlo o realizar otra práctica.

- Todos los equipos deberán contar con una bitácora, en la cual, tanto profesores como alumnos deberán hacer las anotaciones correspondientes cuando utilicen dichos equipos.

- Analizar y definir procedimientos y útiles específicos para el diagnóstico y localización de averías en equipos electrónicos.

- Clasificar los distintos procedimientos generales utilizados para el diagnóstico y localización de averías (de naturaleza mecánica y/o eléctrica) en los equipos electrónicos.

- Definir un parte de averías estándar para el mantenimiento de equipos electrónicos, justificando la elección de cada uno de sus apartados.

- Explicar la utilidad que tiene el realizar un histórico de averías generales y de los equipos individuales en el servicio de mantenimiento de equipos electrónicos.

- Explicar la utilidad que tienen las hojas de servicio interno sobre estadística de averías en un taller de mantenimiento de equipos electrónicos.
- Explicar al menos un procedimiento general utilizado en la definición de útiles específicos diseñados para optimizar el proceso de mantenimiento de equipos electrónicos, basados fundamentalmente en la supervisión de los mismos mediante la utilización de sistemas micro programables.

#### *4.3.3.7. REGLAS BÁSICAS DE SEGURIDAD*

- Conservar todas las herramientas en buen estado con un mantenimiento periódico.
- Utilizar la herramienta adecuada para el desarrollo del laboratorio.
- Examinar todas las herramientas antes de utilizarlas para ver si están dañadas.
- Utilizar las herramientas de acuerdo con las instrucciones del fabricante.
- Seleccionar y utilizar equipos de protección adecuados

### *4.3.3.8. UBICACIÓN*

- No exponga el instrumento a polvo o vibraciones excesivas ni a temperaturas extremas (evite ponerlo al sol), para evitar así la posibilidad de que se deforme o se dañen los componentes internos.

- No utilice el instrumento cerca de aparatos de televisión, radios, equipos estereofónicos, teléfonos móviles ni dispositivos eléctricos de cualquier otro tipo. De hacerlo así, el instrumento, aparato de TV o radio podría generar ruido.

- No ponga el instrumento sobre superficies inestables, donde pueda caerse por accidente.

- Antes de cambiar el instrumento de lugar, desconecte todos los cables.

- No ponga los instrumentos pegados contra la pared (deje un espacio de por lo menos 3 cm/1 pulgada), ya que puede afectar la circulación de aire y hacer que el instrumento se caliente en exceso.

#### *4.3.3.9. CONEXIONES*

Antes de conectar el instrumento a otros componentes electrónicos, desconecte la alimentación de todos los componentes. Antes de apagar o encender los componentes, baje el volumen al mínimo. Asimismo, recuerde ajustar los componentes al nivel mínimo y subirlo gradualmente mientras ejecuta el instrumento, para establecer el nivel de escucha deseado.

### *4.3.3.10. PARA PROGRAMAR LOS MICRO CONTROLADORES*

Con respecto a la programación y manipulación de los micro controladores siempre sean precavidos.

Es casi inevitable, después de un par de semanas, caer en un exceso de confianza que tarde o temprano les pasara la cuenta (a todos nos ha ocurrido). Precauciones mínimas que siempre se deben observar:

- Des energizar el programador antes de colocar un dispositivo (PIC o EEPROM).
- Antes de energizar el programador, verificar que el dispositivo esta correctamente puesto (vean el Manual del Programador).
- Nunca usen los dedos para sacar un dispositivo desde el programador o protoboard.
- Deben utilizar pinzas especiales o hacer palanca desde un lado y otro con un objeto puntiagudo, para no romper las patas del dispositivo.
- Una vez programado el dispositivo, antes de energizar el protoboard verificar que esta correctamente colocado el dispositivo.
- Configurar como entradas los pines que no se utilizaran (en el caso de los micro controladores).
- Nunca conectar Leds u otros elementos de (relativamente) alto consumo directamente a los pines de un micro controlador. Siempre deben hacerlo

a través de resistencias (o transistores para dispositivos más exigentes como motores, por ejemplo).

Antes de manipular el micro controlador, toque una superficie de metal sin pintar y derivada a tierra o póngase una muñequera conductiva. No desconecte a el equipo al menos que se haya apagado la alimentación Antes de manipular un micro controlador, toque una superficie de metal sin pintar y derivada a tierra o póngase una muñequera conductiva.

Cuando sitúe el micro controlador en su zócalo y antes de ejercer presión para insertarlo, compruebe que todas las patillas están alineadas con el respectivo agujero, evitando en todo momento su posible doblegado. A medida que haga presión observe las patillas de cada hilera, alineándolas individualmente si es preciso

Inserte el micro controlador en su zócalo, asegurándose que la patilla 1 del mismo coincide con la etiqueta 1 impresa en la esquina inferior

### *4.3.4. LA ELECTRICIDAD ESTÁTICA*

Es una carga eléctrica causada por la acumulación de un exceso de electrones en la superficie de un material. Para la mayoría de las personas, la electricidad estática y ESD no son más que algo molesto. Por ejemplo, después de caminar sobre una alfombra descalzo, se acumulan electrones en el cuerpo y puede recibir una descarga al tocar un pomo metálico. Esta pequeña descarga libera la electricidad estática acumulada.

#### *4.3.4.1. PRECAUCIONES CONTRA LA ELECTRICIDAD ESTÁTICA*

Asegúrese de descargar cualquier electricidad estática que usted o los dispositivos electrónicos puedan haber acumulado antes de tocar cualquier dispositivo electrónico o de conectar un dispositivo a otro. La recomendación es que tome esta precaución antes de conectar al equipo, colocarla en una base o conectarla a cualquier otro dispositivo. Esto puede hacerlo de diversas maneras, incluidas las siguientes:

- Una superficie de metal que disponga de conexión a tierra. Por ejemplo, si su ordenador tiene una carcasa metálica y está conectado a un enchufe estándar de dos clavijas con toma de tierra, al tocar la carcasa debe descargar la electricidad estática de su cuerpo.

- Aumente la humedad relativa del entorno.

- Evite el contacto directo de las manos durante el transporte y el almacenamiento de productos en bolsas antiestáticas.
- No saque de las bolsas los componentes sensibles a la electricidad estática hasta que lleguen a entornos a prueba de este tipo de electricidad.
- Coloque los componentes en una superficie conectada a tierra antes de sacarlos de las bolsas.
- Procure no tocar las patillas, los contactos ni los circuitos.
- Utilice siempre un método de conexión a tierra adecuado cuando toque un componente o una unidad sensible a la electricidad estática.
- Instale elementos de prevención específicos para Electricidad estática, como alfombrillas de tierra.

#### *4.3.4.2. CONDICIONES QUE MEJORAN LA GENERACIÓN DE LA ELECTRICIDAD ESTÁTICA*

Entre las condiciones que pueden contribuir a la acumulación de electricidad estática en el entorno se Incluyen las siguientes:

- Baja humedad relativa.
- La rapidez con la que toca, conecta o desconecta dispositivos electrónicos.
- Aunque debe tomar siempre las precauciones apropiadas para descargar electricidad estática, si se encuentra en un entorno en el que percibe la existencia de Electricidad estática, deberá tomar precauciones adicionales para proteger el equipo electrónico.

### *4.3.5. CONSEJOS SOBRE SOLDADURA DE LOS INTEGRADOS*

Tras la experiencia de años anteriores, se ha comprobado que más de la mitad de los problemas que han surgido durante el concurso se han debido a malas conexiones, en general por malas soldaduras. Además son errores difíciles de encontrar. Pueden causar cortocircuitos que quemen componentes (como el micro) o falta de alimentación en ciertas partes del circuito (¿por qué el robot no anda si le digo que ande?) Aprender a soldar bien es imprescindible, y hay muchos manuales en la red que nos enseñaran. Es imprescindible leer uno antes de empezar (hay uno en la web). Además hay que usar el MULTIMETRO con frecuencia para comprobar que las soldaduras hagan buena conexión.

### *4.3.6. EVITE LOS SIGUIENTES LUGARES PARA LA OPERACIÓN E INSTALACIÓN DE LOS DISPOSITIVOS ELECTRÓNICOS.*

- Lugares en los que la temperatura ambiente pueda superar el rango comprendido entre 0 y 50 ºc
- Lugares en los que la humedad ambiente pueda superar el rango comprendido ente 45 y 85% de humedad relativa mientras el controlador este funcionando.
- Lugares en los que los cambios en la temperatura ambiente puedan producirse tan rápidamente como para provocar condensación.
- Lugares en los que el micro controlador este sometido directamente a vibraciones y golpes (provocan problemas en el funcionamiento).
- Lugares en este expuesto a polvo, aire salado o aire que contenga partículas de hierro.
- Lugares en los este expuesto a interferencias con la electricidad estática, magnetismo y ruidos.
- Lugares en los que este expuesto directamente a luz solar directa.
- Lugares en los que el calor pueda acumularse calor debido al a la radiación de calor del instrumento.

## 5. PRESUPUESTO DEL PROYECTO.

La tabla siguiente presenta los costos y gastos necesarios para la desarrollo del trabajo de grado.

| CANTIDAD. | DESCRIPCION. | P.U. | P.T. |
|---|---|---|---|
| Adquisicion de equipo: | | | |
| | Computadora COMPAQ DESKPRO (Caracteristicas ver tabla) | $ 100.00 | $ 100.00 |
| | Compra de equipo y dispositivos electronicos (ver tabla) | $ 48.22 | $ 48.22 |
| Microntroladores: | | | |
| 8 | Micro controlador Pic 16F84 | $ 15.00 | $ 120.00 |
| 3 | Micro controlador Pic 16F877 | $ 23.00 | $ 69.00 |
| Programadores de microcontroladores: | | | |
| 5 | Programador de Microcontroladores JDM | $ 25.00 | $ 125.00 |
| Costo promedio por Kit de entramiento (ver tabla) | | $ 7.00 | $ 7.00 |
| Costos de envio de equipo desde el exterior del pais: | | | |
| | costos de envio de los microcontroladores Pics (desde Peru y USA) | $ 60.00 | $ 60.00 |
| | costos de envio de los programadores JDM (desde Peru) | $ 60.00 | $ 60.00 |
| Gastos de papeleria: | | | |
| 3 | Resmas de papel bond | $ 5.00 | $ 15.00 |
| 3 | Marcadores | $ 2.00 | $ 6.00 |
| 3 | Liquidpaper | $ 2.00 | $ 6.00 |
| 3 | Libros acerca de Microcontroladores | $ 20.00 | $ 60.00 |
| | Fotocopias | $ 20.00 | $ 20.00 |
| | Impresiones | $ 10.00 | $ 10.00 |
| | Lapices y lapiceros | $ 2.00 | $ 2.00 |
| | Folders | $ 2.00 | $ 2.00 |
| | Grapas | $ 2.00 | $ 2.00 |
| | Fasters | $ 1.00 | $ 1.00 |
| Gastos consumibles: | | | |
| | Cartuchos de tinta | $ 30.00 | $ 30.00 |
| | Tonner de impresora LASER (Lexmark e120) | $ 15.00 | $ 15.00 |
| | CD's | $ 5.00 | $ 5.00 |
| Otros gastos realizados | | | |
| | Gasto en mantenimiento y reparacion de equipo | $ 15.00 | $ 15.00 |
| | Servicio de internet | $ 60.00 | $ 60.00 |
| | Telefono | $ - | $ - |
| | Contactar a Microperu (llamadas de larga distancia a Peru) | $ 16.00 | $ 16.00 |
| | Servicio de electricidad | $ 40.00 | $ 40.00 |
| | Viaticos | $ 25.00 | $ 25.00 |
| | Honorarios del grupo de trabajo | $ 600.00 | $ 600.00 |
| | | Total | $ 1,519.22 |

| CANTIDAD. | COMPONENTE. | DESCRIPCION. | P.U. | P.T. |
|---|---|---|---|---|
| Resistencias. | | | | |
| 7 | R11..R20 | Resistencias de 10KΩ por 1/4 W. | $0.25 | $1.75 |
| 5 | R1..R10 | Resistencia de 330Ω por 1/4 W. | $0.08 | $0.40 |
| Condensadores. | | | | |
| 4 | C1..C12 | Condensador de 22 µF. 25V. | $0.50 | $2.00 |
| Diodos. | | | | |
| 3 | DR1..DR5 | Diodo led (5mm) color rojo indica modo programación. | $0.12 | $0.36 |
| 7 | DC1..DC10 | Diodo led Común (5mm). | $0.12 | $0.84 |
| Integrados. | | | | |
| 4 | DIS1..DS7 | Display de 7 segmentos | $1.50 | $6.00 |
| Oscilador. | | | | |
| 3 | Q1..Q6 | Oscilador de cristal. Frecuencia de 4MHz. | $0.80 | $2.40 |
| Interruptores. | | | | |
| 6 | P1..P10 | Pulsadores Sw834 | $0.38 | $2.28 |
| 5 | S1..S7 | Switch CC2550 | $0.47 | $2.35 |
| Otros. | | | | |
| 2 | | Placa perforada CL034 | $1.71 | $3.42 |
| 2 | | Tableta cobreada CL110 (de 10cm x 10cm) | $0.67 | $1.34 |
| 3 | | Protoboard | $6.50 | $19.50 |
| 1 | | yda. De estaño | $0.16 | $0.16 |
| 1 | | Ext. p/tester NB426 | $1.58 | $1.58 |
| 6 | | Honz. de percloruro de hierro. | $0.64 | $3.84 |
| 1 | | | Total | $48.22 |

En la siguiente tabla se muestra una breve descripción de la computadora adquirida, para las prácticas de laboratorio de la FMO.

| DESCRIPCIÓN DE LA COMPUTADORA | | |
|---|---|---|
| Marca: | COMPAQ Deskpro | |
| Capacidad en: | | |
| | Disco Duro | 4 GB |
| | Memoria RAM | 64 MB |
| | Windows 98 | |
| Microprocesador: | | |
| | Pentium (400 Mhz) | |
| CD-ROM | Incluido | |

Nota: Ni incluye monitor, teclado ni mouse.

# 6. PROYECCIÓN DEL TIEMPO DE UTILIDAD DE LOS DISPOSITIVOS Y EL EQUIPO

El número de estudiantes que obtendran acceso al uso de estos equipos
se calcula que aproximadamente será de **370 estudiantes** de la carrera de Ingenieria en Sistemas.

Beneficios respecto al tiempo de vida de los Equipos (Hardware):

| | | |
|---|---|---|
| Nº de entrenadores | 5 | Kits de entrenamiento |
| Nº de Programadores | 5 | Quemadores JDM |
| Nº de Microcontroladores | 5 | Pics 16F84 |

Beneficio cuantificado en el tiempo de vida del equipo antes de ser reemplazado.

| | | |
|---|---|---|
| Nº de ciclos que posee un microcontrolador Pic 16F84 | 1000 | Ciclos de borrado/escritura |
| Nº de practicas de laboratorio de micro controladores en un semestre. | 10 | Nº de prácticas Laboratorio |
| Nº de grupos de estudiantes para cada laboratorio de practicas. | 5 | Nº total de grupos de estudiantes |
| Nº total de guias de laboratorio a realizar en un semestre. | **Total** | **50 Guias** |

Promedio de cuantas veces se necesita borrar/escribir por guia.
un microcontrolador antes de que el programa opere bien. 5 Veces por guia.

Nº total de veces de borrar/escribir un microcontrolador en un semestre. **Total 250 veces en un semestre**

| | | |
|---|---|---|
| Nº de semestres que durará el microcontrolador antes de su reemplazo. | 4 | Semestres |
| Nº de años que durará el microcontrolador antes de su reemplazo. | | **2 Años** |

**Es decir que aproximadamente la vida util de un microcontrolador será de 2 años por cada microcontrolador.**

Respecto a los programadores y entrenadores como estan construidos en base a elementos electronicos genericos:
Resistencias, diodos, filtros, transistores, etc. Son dispositivos muy sensibles y susceptibles a sufrir daños facilmente
**Se estima una vida util de aproximadamente será de 3 años por cada programador/entrenador.**

## *7. BENEFICIOS.*

### *BENEFICIOS A LA POBLACIÓN ESTUDIANTIL:*

Se beneficiaran directamente aun aproximado de 35 alumnos que cursan la asignatura de MICROPROGRAMACIÓN; a través de ofrecerles textos para la asignatura, guías practicas de laboratorios con micro controladores además de los medios tanto se software (de desarrollo de aplicaciones y simulación ) como hardware (quemadores y kits de entrenamiento) necesarios para el desarrollo de las practicas de laboratorio y la realización de nuevas aplicaciones con micro controladores, especialmente en el área de interés del alumno, con temarios flexibles y muy enfocados en el aspecto práctico que facilitarán una optima consolidación de los conocimientos teóricos impartidos en clase.

Para los estudiantes es una oportunidad de experimentar fuera del ambiente académico esto permitirá comprender sus limitaciones y proponer futuras extensiones o mejoras, además de clarificar la posibilidad de usar en forma integrada distintas herramientas.

Además se beneficiará indirectamente aun aproximado de 65 alumnos que cursan las asignaturas de SISTEMAS DIGITALES Y ARQUITECTURA DE COMPUTADORAS ya que se generaría una motivación y expectativa de estudio previo al de la asignatura de MICROPROGRAMACIÓN.

Desde el punto de vista didáctico, los beneficios han serán innegables pues el interés, la motivación, y el crecimiento de los estudiantes respecto a los propósitos planteados son evidentes al poseer laboratorio prácticos de micro controladores que cuentan con los recursos que permiten proporcionar a los estudiantes, conocimientos, habilidades y herramientas necesarias para implementar soluciones con micro controladores.

Finalmente se beneficiara a una población estudiantil de aproximadamente de 400 estudiantes de la carrera de INGENIERÍA EN SISTEMAS que podrán adquirir nuevas competencias para la inserción en el mercado laboral. Ya que el perfil de los egresados podrán estar involucrados en el desarrollo, dirección, manufactura, mantenimiento e implantación de soluciones basadas en micro controladores mediante los conocimientos de computación y electrónica digital

***BENEFICIOS AL DEPARTAMENTO DE INGENIERÍA Y LA FACULTAD:***

La Universidad ocupa un lugar especial dentro de la sociedad, tanto por la especificidad de su que hacer como por la manera en que éste se articula con la dinámica social, con su realidad presente y su anhelo futuro.

La universidad enseña, al poder ***promover o promocionar equipo y metodología nueva en la asignatura de MICROPROGRAMACIÓN*** como expectativa de motivación para tomar la asignatura para hacer uso de laboratorios dotados con equipos y dispositivos que tecnológicamente facilitan el aprendizaje a través de la práctica con herramientas de desarrollo de aplicaciones con micro controladores.

***Además ofrecer una formación adecuada que permita a los estudiantes y egresados*** adaptarse rápida y eficientemente a los futuros cambios en las tecnologías de la información. Es imposible negar que la informática es una disciplina que está en continuo cambio y modernización, nuestro objetivo no debe ser sólo que los estudiantes conozcan la tecnología actual, sino que adopten esquemas de trabajo que les permitan adaptarse fácilmente a los nuevas tecnologías que con seguridad surgirán en el futuro.

La Institución ejerce ***influencia positiva en el entorno*** debido al impacto de sus estudiantes y egresados; según la especificidad del programa, el plan de estudios incorpora el análisis de problemas del entorno; el egresado es reconocido por la calidad de la formación que reciben facilitando la consecución de ofertas laborales para estudiantes y egresados, permitiendo estar preparados para asumir los nuevos retos planteados por la mundo global en materia de tecnología.

***Estrategia de difusión***

Se espera poder publicar trabajos en concursos de proyectos, ferias educativas de proyectos de micro controladores, conferencias y medios arbitrados de relevancia en la temática correspondiente. Además con la impartición de diversos cursos y seminarios de capacitación y actualización de micro controladores. En este espacio los estudiantes que realizan proyectos de desarrollo e investigación de soluciones con micro controladores, ya que cuentan con equipo y herramientas que sirva para generar nuevos proyectos de grado basados micro controladores. Logrando que el DEPARTAMENTO Y LA FACULTAD sea una entidad académica con reconocimiento local y regional,

donde se formen profesionales capaces de identificar y resolver los problemas de su entorno.

La adquisición del hardware para llevar acabo las prácticas de laboratorio el DEPARTAMENTO DE INGENIERÍA Y LA FACULTAD ***mejorará su infraestructura educativa de sus laboratorios*** para garantizar la formación de ingenieros con alto desarrollo tecnológico. Laboratorios dotados con equipos y dispositivos tecnológicamente avanzados que facilitan el aprendizaje a través de la práctica con, herramientas de desarrollo de micro controladores, computadores con acceso permanente a Internet para desarrollos y simulación de circuitos entre otros.

Por ultimo promover, fomentar y desarrollar capacitaciones a los docentes que tengan la motivación de conocer está tecnología.

## *8. CONCLUSIONES.*

Considerando que el propósito principal de nuestro trabajo de graduación es la elaboración de la propuesta de implantación de prácticas de laboratorio con micro controladores a través de medios como hardware, software y la elaboración de las guías de práctica a fin de dar soporte a los conocimientos teóricos y convertirlos en un elemento útil en el proceso de enseñanza-aprendizaje. A fin de elevara sustancialmente la calidad académica de la facultad.

Inicialmente la comunidad de estudiantes de la facultad han manifestado su agrado frente a este tipo de laboratorio al poder aplicar de manera fácil y didáctica sus conocimientos teóricos.

Con la realización de las prácticas en el laboratorio sirvan para motivar la participación constante del estudiante, usando ayudas didácticas que permitan la fácil asimilación de los temas de la asignatura y utilizando los Kits de entrenamiento, Software y bibliografía para facilitar el proceso de aprendizaje.

Que los estudiantes no solo cuenten con la información teórica sino que también pueden desarrollar la aplicación de manera real, o sea, logran transitar desde el diseño y la simulación, hasta la comprobación práctica.

Brindar a los alumnos avanzados de la carrera, prácticas con herramientas de desarrollo para hardware y software en el campo de diseño con micro controladores. Este entrenamiento les permitirá utilizar el micro controlador para realizar aplicaciones, como: su tesis de grado, trabajos de

investigación, trabajos a terceros y le acreditará sólidos conocimientos prácticos para mejorar su oportunidad de inserción en el ámbito laboral.

Crear un punto de partida para generar soluciones tecnológicas a la sociedad. Además de crear una metodología innovadora en la enseñanza de lenguajes de programación para alumnos de ingeniería en sistemas informáticos, mediante el uso de micro controladores.

Incrementar en los estudiantes las capacidades para resolver problemas reales o cercanos a ellos con mayor independencia. Buscando fomentar iniciativas de desarrollo y mejoramiento de la calidad.

La elección del micro controlador PIC16F84 nos permitió ver el alcance de estos componentes en la soluciones a problemas específicos de ingeniería que integren hardware y software, que partiendo de una tecnología se tenga un valor agregado con ventajas de costo o facilidad de uso. En este caso se pudo ver la versatilidad y flexibilidad de este elemento en diversas tareas.

Nuestro trabajo deja creadas las bases para la implementación de otros proyectos de este campo y sentadas las condiciones para realizar simulaciones de procesos que irán aumentando en complejidad hasta alcanzar magnitudes similares a las que podamos encontrar en alguna de nuestras industrias.

Todo el estudiantado a partir de este momento contará con un instrumental de laboratorio muy novedoso y actual que le permitirá salir mejor formado para su futuro como ingeniero en sistemas informáticos.

La utilización del paquete de software de desarrollo integral gratuito (MPLAB), resuelve muchos de los problemas existente en el proceso de diseño, desarrollo y simulación de soluciones basadas en micro controladores, ya que las futuras investigaciones pueden emplearlo sin requerimientos de licencia de elevados costos, los alumnos cuentan con una herramienta de depuración y simulación en sus hogares sin necesidad de hacer uso de este paquete de desarrollo que no solo se encuentran en la sala de laboratorio. Sino que es accesible en todo momento a través del internet.

Esta investigación permite sentar las bases para próximas investigación en el área de hardware reconfigurable soportado con el diseño mediante lenguajes descriptores de hardware (Ensamblador), de manera que se promueva la introducción de nuevos contenidos en el área de microprogramación. Si bien este tipo de tecnología no tiene aún mucho auge en el ámbito local para aplicaciones de ingeniería, no debe ignorarse como una herramienta adicional de soporte para el diseño de soluciones basadas con micro controladores.

Si bien la presente experiencia va dirigida alumnos de Ingeniería en sistemas informáticos, ésta puede ser de utilidad para otras ingenierías, sobre todo si tenemos en cuenta que vivimos en la denominada “era de las comunicaciones”.

## *9. RECOMENDACIONES.*

1. A partir de la experiencia en la realización de este proyecto se pueden considerar algunas recomendaciones y sugerencias para seguir con trabajos en esta rama.

2. Que mediante la resolución de las prácticas de laboratorio, conduzcan a los alumnos a realizar los siguientes tipos de actividades:

3. La resolución de las guías prácticas, verificar las limitaciones y alcance del conjunto de instrucciones del micro controlador, interpretar alternativas en modos de direccionamiento, etc.

4. Elaboración de proyectos, a efectos de de integrar los conocimientos sobre las herramientas y tecnologías introducidas, los alumnos deberán realizar proyectos basados en el micro controlador. Cada grupo, deberá resolver problemas específicos propuestos por el catedrático de la asignatura y preparar informes respecto de los algoritmos desarrollados.

5. Tareas de investigación, en la resolución de los proyectos, los alumnos propondrán las metodologías adecuadas para la resolución óptima de problemas, dado que en desarrollo de software siempre hay muchos caminos para obtener un resultado, por lo que es imprescindible especificar los pro y los contra de las alternativas.

6. Aumentar y eficientizar la difusión de acciones y espacios vinculados a las aplicaciones con micro controladores (a través de mails, carteles más visibles, foro de discusión, espacio en la página web, referentes en las facultades, etc.). Queremos fortalecer el sentido de pertenencia por la institución, seguir promoviendo investigaciones de estudiantes y profesores a través de cursos, seminarios y talleres.

7. Los grupos de prácticas máximo 4 personas lo que permite al instructor brindar atención personalizada a los estudiantes. A cada grupo de estudiante se le asigna un computador, programador y micro controlador de modo que durante las prácticas los proyectos son puestos a prueba en un circuito real de manera inmediata.

8. La revisión de las prácticas de laboratorio por parte de los estudiantes con la antelación suficiente puede ayudar a obtener una visión global, que facilitará el desarrollo en cada una de las partes de que constan.

9. Es de suma importancia seguir los pasos del proceso de diseño y desarrollo de una aplicación basada con micro controladores, para comprobar que el diseño es algo muy simple a nivel teórico pero a la hora de implementarlo en la práctica es más complicado de lo que parece.

10. El PIC16F84 tiene 4 configuraciones para el oscilador externo, utilizar un oscilador externo XT de 20MHz es una solución efectiva y de bajo costo.

11. Es preciso que los estudiantes tengan conocimientos de algún lenguaje de programación de alto nivel, preferentemente C/C++. También resultan necesarios conocimientos básicos de arquitectura de computadores y nociones de programación en lenguaje ensamblador. Es conveniente que dominen conceptos de ingeniería del software, para la elaboración de diagramas de análisis (Flujo gramas), diseño programas. Además de dominar conceptos relacionados con los sistemas operativos.

12. Es importante que el alumno posea una base sólida sobre sistemas digitales (combi nacionales y secuenciales). Igualmente, es deseable el adecuado dominio de los fundamentos informáticos, lo relativo a programación y rudimentos de algoritmia, para dotar al alumnado de un bagaje y actitud mental que favorecerá la más rápida asimilación de la metodología de la programación de los sistemas basados en micro controladores.

13. Promover visitas industriales para observar aplicaciones con micro controladores.

14. Promover la implementación de proyectos afines a la asignatura.

15. Mejorar la calidad del cuerpo docente a través de la impartición de talleres de capacitación de Micro controladores para los docentes de manera que se eleve tanto su conocimiento sobre las materias que enseñan, como sus competencias educativas. Así estaremos contribuyendo a que los estudiantes gocen de la educación que merecen, una educación relevante, atractiva y de calidad creciente para todas y todos.

## *10. BIBLIOGRAFÍA.*

**Trabajos de Graduación:**

Argueta, Chinchilla, Reyes (2006). Propuesta para la implementación del laboratorio de hardware y elaboración de las guías de práctica de las asignaturas de especialidad de la carrera de Ingeniería de Sistemas Informáticos. Trabajo de graduación para optar al título de Ing. de Sistemas Informáticos, Departamento de Ingeniería y Arquitectura, Facultad Multidisciplinaria de Occidente, Santa Ana, El Salvador.

**Libros:**

1. Titulo: Micro controladores PIC, diseño práctico de aplicaciones.

   Autor: José Ma. Angulo Usategui, Ignacio Angulo Martínez.

   3ª Edición. 2002.

2. Título: Micro controladores PIC, diseño práctico de aplicaciones.

   Autor: José Ma. Angulo Usategui, Ignacio Angulo Martínez.

   3ª Edición. 2003.

3. Título: Programming and customizing PICmicro Microcontrollers.

   Autor: Myke Predko.

   2ª Edición. 2000.

4. Título: PIC microcontroller Project book.

   Autor: John Lovine.

   1ª Edición. 2000.

5. Título: PIC in Practice. A Project-based Approach 2Ed

   Autor: D. W. Smith.

   2ª Edición. 2006.

**E-books:**

1. Título: Microcontroller – Complete Guide to PIC.

2. Título: PIC microcontrollers, for beginners too.

**WEB:**


1. http://www.electron.es.vg
2. http://solelec.byethost17.com/index.php
3. http://www.in-circuitsolutions.com/index.htm
4. http://perso.wanadoo.es/luis_ju/
5. http://www.herostechnology.co.uk/default.html
6. http://www.ucpros.com
7. http://pikdev.free.fr
8. http://www.todorobot.com.ar
9. http://www.rentron.com/pic.htm
10. http://micropic.galeon.com/index.html
11. http://www.info-ab.uclm.es/asignaturas/42587/
12. http://electrolinks.blogspot.com

**Documentos:**


1. Título: PIC16F87X Data Sheet.

   Autor: Microchip Technology Inc.

2. Título: PIC16F87X Data Sheet Errata.

   Autor: Microchip Technology Inc.

3. Título: PIC16F8X Data Sheet.

   Autor: Microchip Technology Inc.

4. Título: PICDEM 2 Plus, Demonstration Board, User's Guide.

   Autor: Microchip Technology Inc.

5. Título: PICDEM-1, User's Guide.

   Autor: Microchip Technology Inc.

6. Título: Version 3 PICmicro microcontroller development board Technical data.

   Autor: Matrix Multimedia.

7. Título: PICmicro MCU Multiprogrammer, EB006-5, Technical data.

   Autor: Matrix Multimedia.

## *11. GLOSARIO.*

Glosario de términos usados en este proyecto.

A

***ALU:*** Unidad Aritmética y Lógica.
***A/D:*** Acrónimo de Analog to Digital [De analógico a digital] Expresa la conversión de un sistema al otro.

C

***CPU:*** Unidad de Proceso Central. Director y principal realizador de procesos de la computadora. Circuito microprocesador que realiza los procesos de datos básicos y controla el funcionamiento general de la computadora.

***CAN (Controller Área Network):*** para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE.UU. se usa el J185O.

D

***DIP:*** Tipo de encapsulado rectangular de circuito integrado. Dispone de hileras de patillas paralelas, distribuidas simétricamente de dos en dos.

***Display:*** Término inglés para definir determinados visualizadores, que mediante diodos de emisión luminosa (LED), o cristales líquidos (LC), permite la composición de dígitos o letras

***DsPICs (Digital Signal Controller):*** Son el último lanzamiento de Microchip, comenzando a producirlos a gran escala a finales de 2004. Son los primeros PICs con bus de datos inherente de 16 bits. Incorporan todas las posibilidades de los anteriores PICs y añaden varias operaciones de DSP implementadas en hardware, como multiplicación con suma de acumulador (multiply-accumulate, o MAC), barrel shifting, bit reversión o multiplicación 16x16 bits).

E

***ELECTRÓNICA:*** Rama de las ciencias físicas que se ocupa del estudio y manejo de los fenómenos electromagnéticos como medio de información.

***ENSAMBLADOR:*** Lenguaje de programación que utiliza símbolos y palabras, más difícil de manejar que los lenguajes de alto nivel, pero más fácil que el lenguaje máquina. También, programa que traduce el lenguaje ensamblador a lenguaje máquina.

***EPROM:*** Siglas de Erasable Programmable Read Only Memory, un mecanismo que contiene información bajo una forma semipermanente y que exige una exposición a la luz ultravioleta en el caso de querer borrar su contenidos.

***E/S:*** Acrónimo de 'Entrada/Salida'. Suele aplicarse al flujo de datos. También es conocido por su acrónimo inglés 'I/O'.

H

***HEXADECIMAL:*** Un sistema de contar muy apreciado por los programadores de códigos, ya que está en estrecha relación con el método de almacenaje utilizado por los ordenadores; está basado en el número 16 en vez del 10, que es el número en el que se basa nuestro sistema de contar ordinario.

I

***I/O:*** Acrónimo de Input/Output [Entrada/Salida]. Suele aplicarse al flujo de datos. También es conocido por su acrónimo castellano 'E/S' (Entrada/Salida)

***INSTRUCCIÓN:*** Elemento del código de programación que le dice al ordenador que lleve a cabo una tarea determinada. Una instrucción en lenguaje de ensamble, por ejemplo, podría ser ADD, con lo que le dice al ordenador que realice una suma.

L

***LSB:*** Bit menos significativo, es aquel que está más a la derecha y que tiene el menor valor posicional.

***LED:*** Light Emitting Diode. Diodo emisor de Luz

***LCD:*** Liquid Crystal Display. Pantalla de cristal líquido. Tecnología que permite la creación de pantallas planas.

***LPT:*** Forma que se denomina al puerto paralelo. Normalmente utilizado para la impresora. LPT1, LPT2.

M

***MEDICIÓN:*** Determinación del valor de una magnitud.

**MICRO CONTROLADOR:** Es un circuito integrado que contiene muchas de las mismas cualidades que
Una computadora de escritorio, tales como la CPU, la memoria, etc., pero no incluye ningún dispositivo de “comunicación con humanos”, como monitor, teclados o mouse. Los Micro controladores son diseñados para aplicación de control de máquinas, más que para interactuar con humanos.

***MULTÍMETRO:*** instrumento de múltiples propósitos, que se puede usar para medir resistencias, voltajes, corrientes, etc.

***MSB:*** Bit más significativo, es aquel que está más a la izquierda y que tiene el mayor valor posicional.

P

***PLC (Controlador lógico Programable):*** Dispositivo electrónico que controla máquinas y procesos. Utiliza una memoria programable para almacenar instrucciones y ejecutar funciones específicas que incluyen activación y desactivación, temporización, conteo, secuencia, aritméticos y manejo de datos. Los PLC incluyen componentes que hacen la interfaz con los dispositivos de control de entrada y salida. Como dispositivos de entrada se pueden tener botones, selectores, interruptores de límite y nivel, sensores, entre otros. Como dispositivos de salida se pueden tener válvulas, arrancadores de motores, bobinas de relevador, alarmas, luces, ventiladores, bocinas torretas, entre otros.

R

***RAM**:* Memoria de acceso aleatorio, es una memoria semiconductora volátil, en la cual se garantiza que los tiempos de acceso son iguales sin importar la ubicación de la información.

***RS-232:*** Norma de conexión estándar que conecta un Modem, o el equipo asociado a la terminal, a un ordenador.

T

***TTL:*** Lógica de transistor, son dispositivos electrónicos formados por transistores que utilizan niveles de voltaje binarios para su operación, donde 0V establece un nivel lógico de cero y 5V establece un nivel lógico de uno.

U

***USART: Acrónimo de Universal Synchronous/Asynchronous Receiver/Transmitter*** [Receptor/transmisor síncrono/asíncrono universal] Circuito integrado que controla el interfaz entre el módem y el ordenador. La 'USART' convierte los datos recibidos por el módem en caracteres y se los entrega al ordenador. Inversamente convierte los caracteres procedentes del ordenador y se los entrega al módem en forma de datos para que los envíe.

***UART: Acrónimo de Universal Asynchronous Receiver/Transmitter [Receptor/transmisor asíncrono universal]*** Circuito integrado que controla el interfaz entre el módem y el ordenador. La 'UART' convierte los datos recibidos por el módem en caracteres y se los entrega al ordenador. Inversamente convierte los caracteres procedentes del ordenador y se los entrega al módem en forma de datos para que los envíe.

# Anexos.

**Anexo 1**

Encuesta realizada a los alumnos de la carrera de ingeniería de sistemas informáticos

**Anexo 2**

Cuestionario utilizado en las visitas técnicas realizadas a las instituciones educativas

**Anexo 3**

Guías prácticas de laboratorio con micro controladores

**Anexo 4**

Visitas técnicas realizadas a las instituciones educativas

**Anexo 5**

Manuales del software de apoyo para prácticas

- Lenguaje Ensamblador
- MPLAB IDE
- WinPIC800
- Proteus

## ANEXO 1: FORMATO DE ENCUESTA DIRIGIDA A LOS ALUMNOS DE INGENIERIA DE SISTEMAS INFORMATICOS.


**UNIVERSIDAD DE EL SALVADOR**

**FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE**

**DEPARTAMENTO DE INGENIERIA**

**INGENIERIA EN SISTEMA INFORMATICOS**

**ENCUESTA DIRIGIDA A LOS ALUMNOS DE LA CARRERA DE INGENIERIA DE SISTEMAS INFORMATICOS.**

**OBJETIVO:** Obtener información que permita evaluar la calidad del desarrollo de los laboratorios prácticos, así como la infraestructura y el contenido de la asignatura de MICROPROGRAMACIÓN de la carrera de INGENIERÍA EN SISTEMAS INFORMÁTICOS.

**Indicaciones:** Lea detenidamente y conteste a continuación lo que se le solicita, marque una X el cuadro de la respuesta que considere apropiada, según sea el caso.

1. En que nivel de la carrera de INGENIERÍA DE SISTEMAS INFORMÁTICOS se encuentra actualmente.

3º ☐ 4º ☐ 5º ☐ Año

2. Especifique las asignaturas en las cuales a realizado prácticas de laboratorio y determine el grado en que las prácticas ayudaron a alcanzar los objetivos de la asignatura.

| | | Mucho | Regular | Poco | Nada |
|---|---|---|---|---|---|
| ☐ | Métodos Experimentales | ☐ | ☐ | ☐ | ☐ |
| ☐ | Física III | ☐ | ☐ | ☐ | ☐ |
| ☐ | Sistemas Digitales | ☐ | ☐ | ☐ | ☐ |
| ☐ | Arquitectura de Computadoras | ☐ | ☐ | ☐ | ☐ |
| ☐ | Microprogramación | ☐ | ☐ | ☐ | ☐ |
| ☐ | Comunicaciones I | ☐ | ☐ | ☐ | ☐ |
| ☐ | Protocolos de Comunicación | ☐ | ☐ | ☐ | ☐ |

Otras:

_____

3. Establezca a su criterio, la cantidad de prácticas de laboratorio por ciclo y tiempo en minutos de cada una.

☐ 1 - 4 Laboratorios ☐ 15 – 30 minutos

☐ 4 – 8 Laboratorios ☐ 30 – 45 minutos

☐ 8 – 10 Laboratorios ☐ 45 – 60 minutos

4. Posee conocimientos previos de Electrónica Digital?

   ☐ Bachillerato ☐ Cursos Independientes ☐ Otro

5. Que grado de motivación presenta para tomar la asignatura de Microprogramación?

   ☐ Muy motivado ☐ Motivado ☐ Poco ☐ Nada

6. Que expectativa en % considera que las practicas de laboratorio de Microprogramación podría consolidar el conocimiento impartido en el aula, en el tiempo de 1 ciclo de estudio.

   ☐ 100% ☐ 75% ☐ 50% ☐ 25% ☐ 0%

7. Conoce el programa de estudio de la asignatura de Microprogramación?

   ☐ SI ☐ NO

8. En que medida considera la combinación de prácticas de laboratorio y los temas expuestos en el plan de estudio puedan favorecer en un ciclo del año electivo en alcanzar los objetivos del plan de estudio

   ☐ 100% ☐ 75% ☐ 50% ☐ 25% ☐ 0%

9. Considera que las practicas de laboratorio de las asignaturas previas de Microprogramación, le ayudaron reforzar los conocimientos básicos de Electrónica Digital?

   ☐ Mucho ☐ Regular ☐ Poco ☐ Nada

10. Especifique su preferencia en la utilización de las herramientas de apoyo y trabajo en las prácticas de laboratorio de Electrónica Digital?

    ☐ Software – Simuladores ☐ Hardware ☐ Ambos

**ANEXO 2: FORMATO DEL CUESTIONARIO**

**UNIVERSIDAD DE EL SALVADOR.**

**FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE.**

**DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA.**

## CUESTIONARIO.

### DIRIGIDA A:

Personal responsable de la administración, operación e Impartición de las prácticas de laboratorio de Micro Controladores de las Instituciones seleccionadas para realizar visitas técnicas.

### OBJETIVO:

Obtener información sobre la infraestructura, equipo y normas de higiene y seguridad con la que cuentan otras instituciones que nos permita conocer los recursos utilizados para el proceso enseñanza-aprendizaje acerca de los micros controladores.

1. ¿A que carrera(s) pertenecen las prácticas de laboratorio acerca de micros controladores?
2. ¿A que asignatura(s) pertenecen las prácticas de laboratorio?
3. ¿A que nivel de la carrera(s) se imparte estas practicas de laboratorio?
4. ¿Cuál es la metodología de trabajo que utilizan para realizar las prácticas de laboratorio?
5. ¿Horarios?

6. ¿Equipo utilizado?
7. ¿Cuáles son las normas utilizadas dentro del laboratorio de prácticas de micro controladores?
8. ¿Cuáles son las precauciones mínimas que utilizan para la programación y manipulación de los micro controladores?
9. ¿Cuáles son los procedimientos para realizar el mantenimiento del equipo?
10. ¿Cantidad de alumnos que conforman los grupos de trabajo en los laboratorios?
11. ¿Cronología de trabajo que se lleva acabo en una practica de laboratorio?
12. ¿Cuáles son algunas ejemplos de practicas de la laboratorio que emplean?
13. ¿Cuáles son los errores más frecuentes que ocurren durante las prácticas de laboratorios de micro controladores?
14. ¿Cuáles son y si existen proyectos a desarrollar durante el desarrollo de la enseñanza de los micro controladores?
15. ¿Cuáles son las normas de manejo de los micros controladores?
16. ¿A cerca de la manipulación antes y después de la práctica de laboratorio?

17. ¿Cuál es el procedimiento de almacenamiento de los micros controladores y el resto del equipo utilizado en las prácticas de laboratorio?
18. ¿Cuál es el personal encargado del almacenamiento del equipo?
19. ¿Cuáles son los modelos que utilizan de micro controladores para realizar las prácticas?
20. ¿Por qué utilizan estos modelos?
21. ¿Qué software de programación de micros controladores utilizan?
22. ¿Bajo que plataforma lo ejecutan?
23. ¿Cuál entorno de desarrollo utilizan?
24. ¿Qué simulador(es) utilizan?
25. ¿Cual fue el procedimiento empleado para la adquisición de los micros controladores?
26. ¿Cuáles fueron las dificultades que encontraron para adquirir el equipo de práctica de laboratorio de micros controladores?
27. ¿Cuáles fueron sus proveedores?
28. ¿Poseen soporte técnico y en consiste?

***GRACIAS POR SU COLABORACIÓN***

# Guias Prácticas


UNIVERSIDAD DE EL SALVADOR

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE

DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA

INGENIERÍA EN SISTEMAS INFORMÁTICOS

MICROPROGRAMACIÓN

# *GUÍA DE PRÁCTICA Nº 1*

## *FAMILIARIZACIÓN CON EL AMBIENTE DE DESARROLLO DE MPLAB Y WINPIC*

### *OBJETIVOS.*

***En este laboratorio el alumno aprenderá:***

- Crear y editar el código fuente usando el editor built-in
- Trabajar con códigos ensamblador, compilador y enlazador.
- Realizar mediciones de tiempos con el simulador.
- Ver variables en las ventanas de visualización.

### *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

### *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB previamente instalado y funcionando correctamente.

***CÓDIGO FUENTE DE UN PROGRAMA QUE CUENTE EL NÚMERO DE VECES QUE SE PULSÓ LA TECLA CONECTADA AL TERMINAL RA4 Y QUE SAQUE ESE VALOR EN BINARIO NATURAL POR EL PUERTO B.***

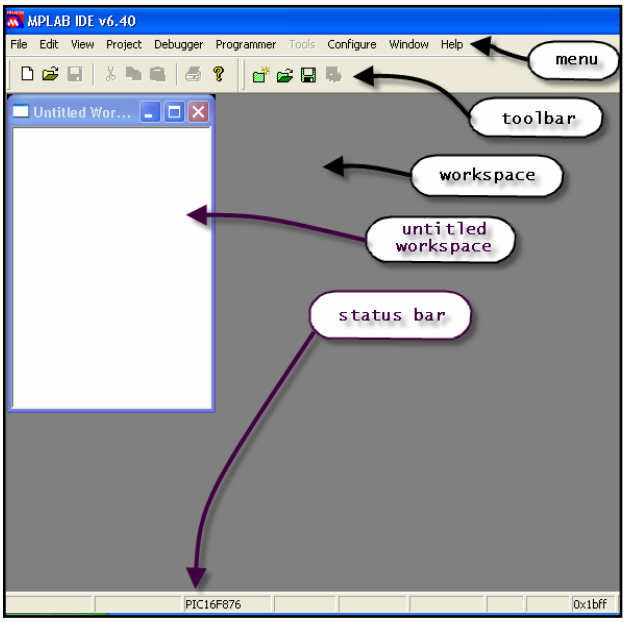
```
        LIST P=16F84
        INCLUDE "p16F84.inc"

        __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

        PAGE
CONTADOR EQU 0X20               ; variable que va a contar el número de Pulsaciones
                ORG 0X00
        GOTO INICIO
        ORG 0X05                ; saltamos el vector de interrupción
INICIO  CLRF PORTB              ; inicializamos PORTB
        BSF STATUS, RP0         ; Vamos al BANK  1 de la memoria
        MOVLW 0X00
        MOVWF TRISB^ 0x080      ; Se configura PORTB como Salida
        BCF STATUS, RP0         ; Volvemos al BANK 0
        CLRF CONTADOR           ; Inicializamos el CONTADOR de pulsaciones
ESPERA  BTFSS PORTA, 4          ; Esperamos A que RA4 pase de "1" a "0".
        CALL INCREMENTO         ; Se pulsó, vamos al subprograma de
incremento.
        GOTO ESPERA             ; No se pulsó, seguimos esperando
INCREMENTO INCF CONTADOR, 1     ; Aumentamos el CONTADOR
        MOVF CONTADOR, W        ; Sacamos el valor de esa cuenta
        MOVWF PORTB             ; Por el PORTB, y por tanto, por los Leds
SOLTAR  BTFSS  PORTA, 4         ; Esperamos a que se suelte el pulsador
        GOTO SOLTAR             ; Se soltó la tecla, RA4 pasa a 1
VOLVER  RETURN                  ; Volver al programa principal.

Finished
        END                     ; Fin del programa
```

## *PROCEDIMIENTO.*

A continuación se expondrán los pasos a seguir para utilizar MPLAB

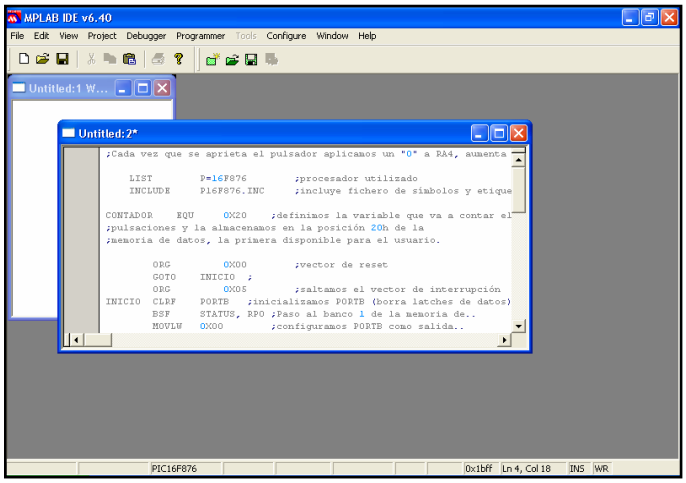
### *APERTURA DEL PROGRAMA*

**1.** Se abre el programa MPLAB IDE haciendo doble clic sobre el logo, entonces aparece la siguiente pantalla.

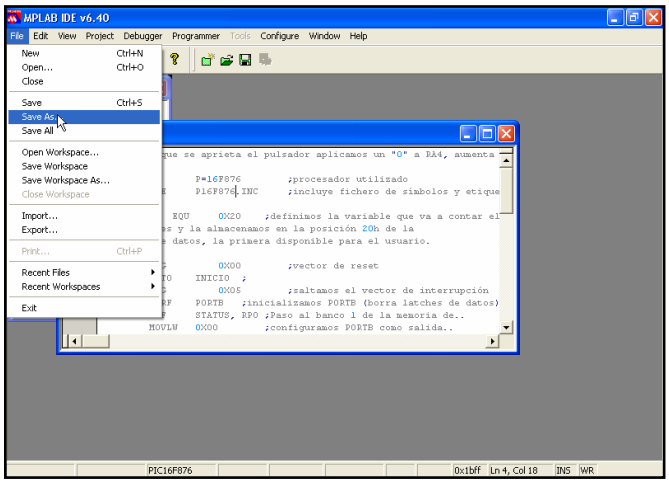



### 2. *Crear un código fuente.*

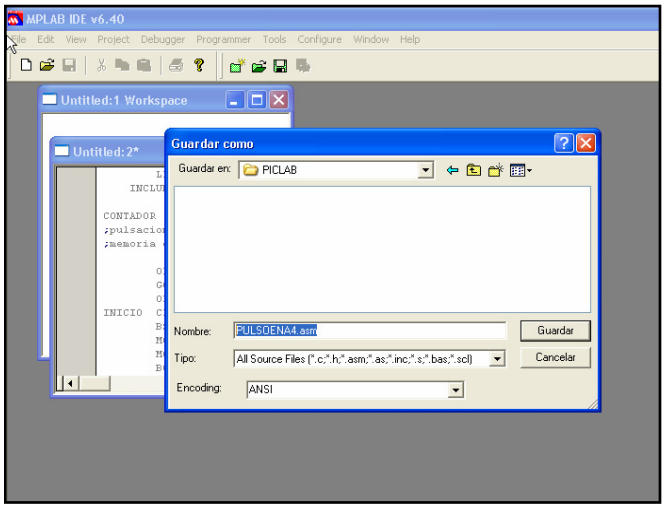
Escribe el código fuente del programa usando el editor de textos del MPLAB, para ello se deben de seguir los siguientes pasos: seleccionar ***File – New,*** se abre una ventana en blanco en la zona de trabajo (***Workspace***). Se puede escribir en ella el programa a simular o directamente “pegar” el programa si proviene de otro documento.




3. ***Guardar*** el documento en blanco utilizando ***File – Save as***




En el ejemplo lo guardamos en C:\PICLAB con el nombre PULSOENA4.asm

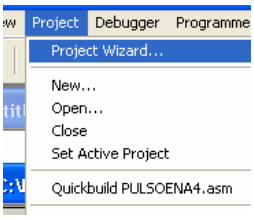



Se debe de tener en cuenta que sólo admite un path con un número menor de 64 caracteres. Una vez guardado, vemos en la pantalla que cambia el color del texto editado (etiquetas, operandos, comentarios, etc.).

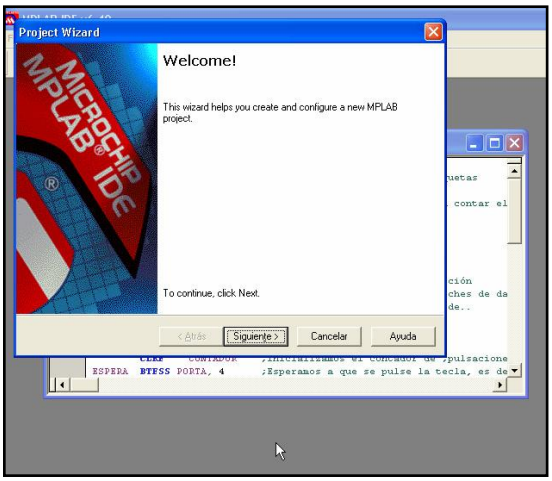

## 4. *Crear un proyecto*

Se debe preparar el proyecto, para no olvidar pasos intermedios es conveniente utilizar el ***MPLAB Wizard***, que llevará paso a paso la ejecución del mismo.

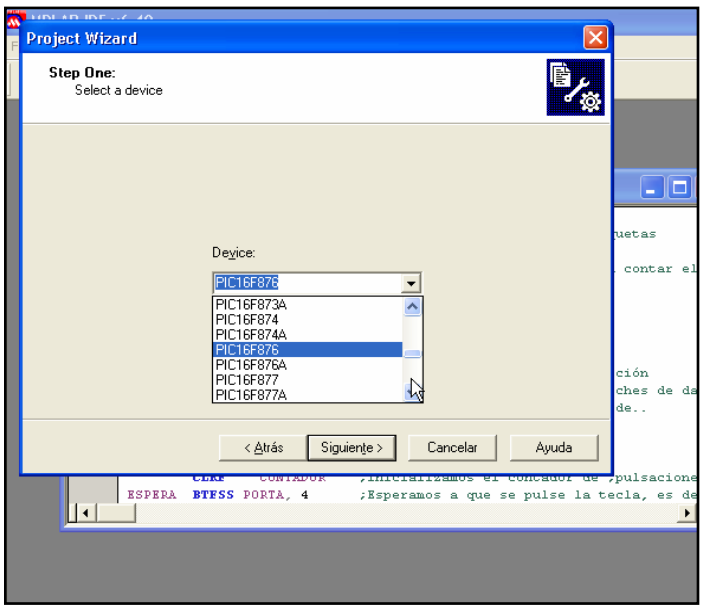
Abrir el Wizard. Para ello seleccionar ***Project – Project Wizard***.




Seleccionar ***Siguiente*** para continuar.

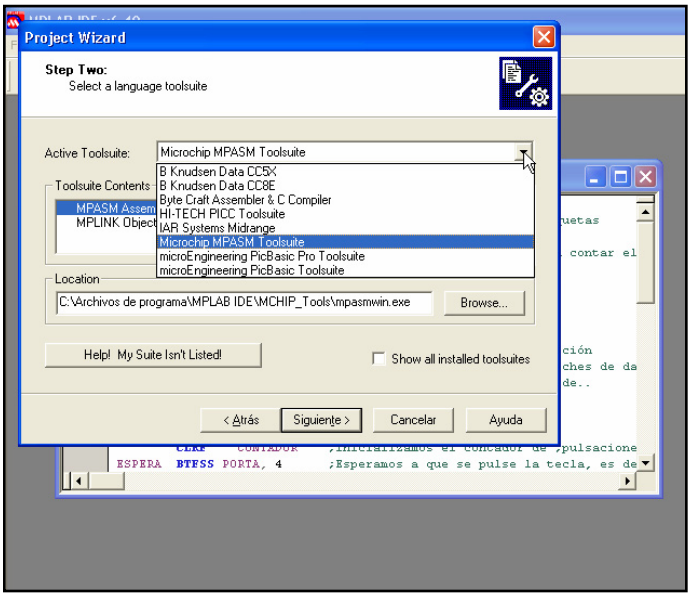



Seleccionar el dispositivo a utilizar. Abriendo la ***pestaña*** se elige el micro controlador, en nuestro caso, el ***PIC16F84***, luego seleccionamos ***Siguiente*** para continuar.

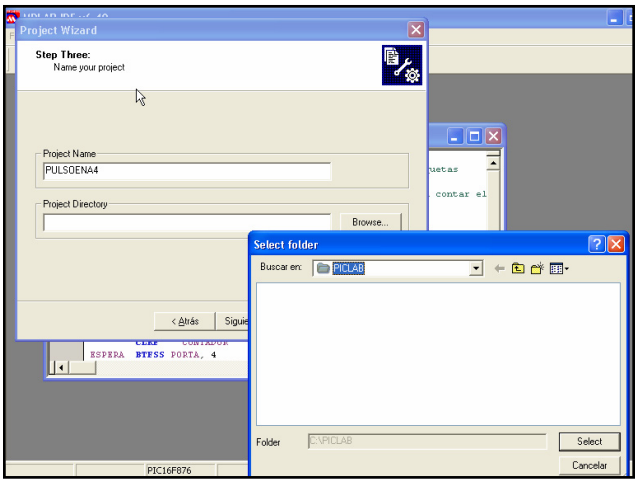


Seleccionar la ***Herramienta***. Abrir la pestaña en ***Active Toolsuit*** y elegir ***MPASM*** que es el ejecutable ensamblador. Si está instalado el programa con los valores por defecto, aparece la pantalla como en la vista, sino, habrá que buscarlo utilizando el ***Browse***.

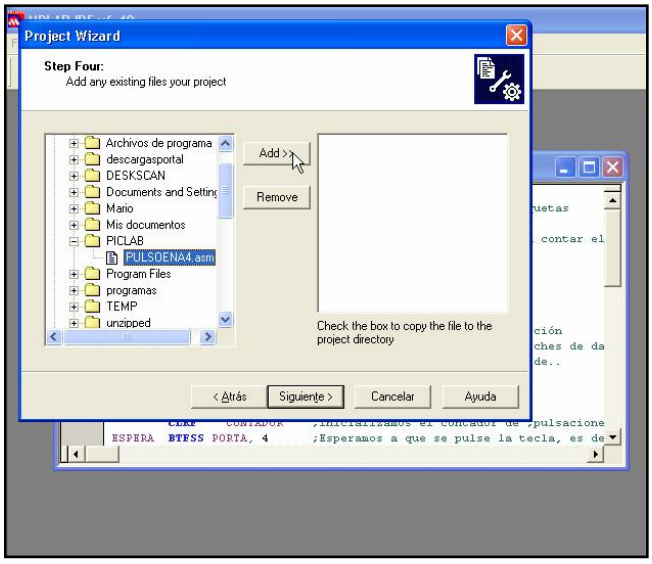
Seleccionar ***Siguiente*** para continuar.



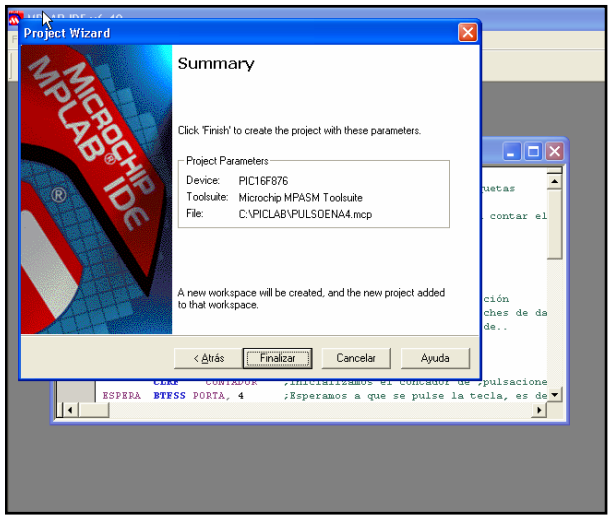

Poner nombre al proyecto. Le ponemos el mismo nombre que el ***asm*** creado, será entonces ***PULSOENA4*** y lo colocamos en C:\PICLAB. Seleccionamos ***Siguiente*** para continuar.



Agregar archivos al proyecto.  Buscar en C:\PICLAB el fichero ***PULSOENA4.asm*** grabado anteriormente  y agregarlo (***Add***).



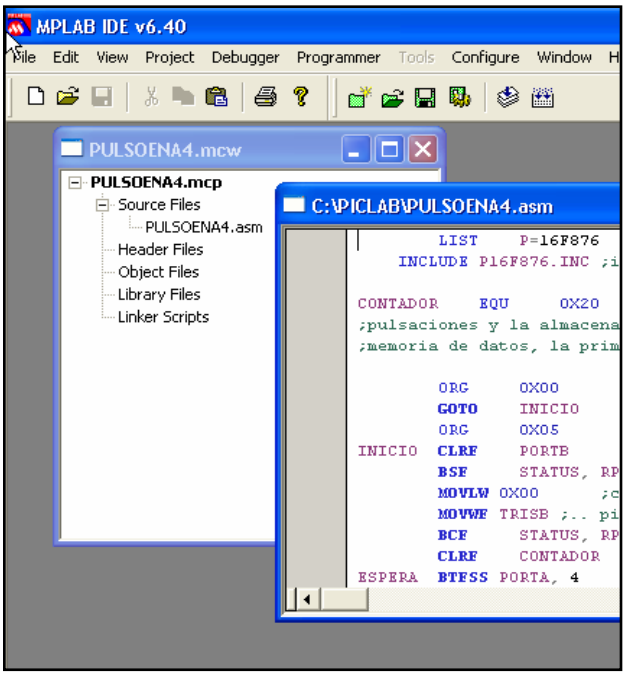

Se completa así la creación del proyecto dando la información sobre el mismo. Seleccionar ***Finalizar*** para salir del Wizard



***ENSAMBLAJE Y SIMULACIÓN.***

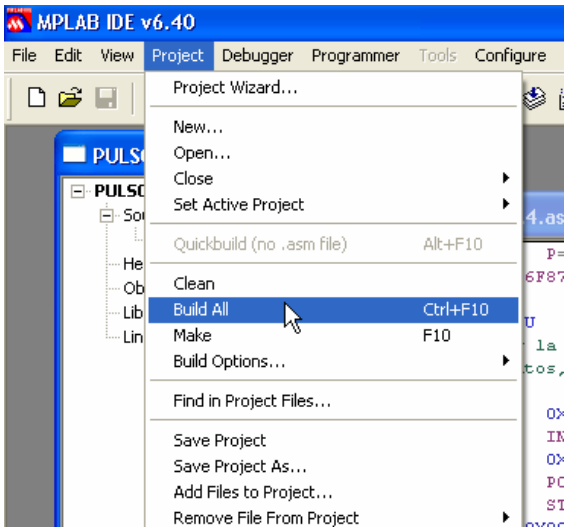
Una vez creado el proyecto se deben de realizar los siguientes pasos:

**5.** ***Ventana del proyecto***. Los archivos pueden agregarse, guardarse o borrarse en esta ventana. Para ello se hace clic con el botón derecho del ratón luego de seleccionarlo.

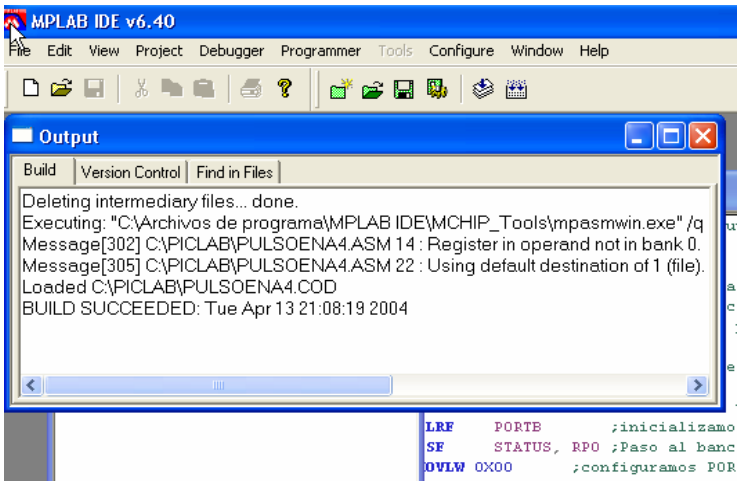



**6.** ***Construcción del proyecto.***

Una vez creado el proyecto, se debe de construir. Es decir, se debe ensamblar. Seleccionar ***Project – Build All*** para construir el mismo.




Si se efectuó de forma correcta, aparecerá la siguiente pantalla (***Ventana de salida***).

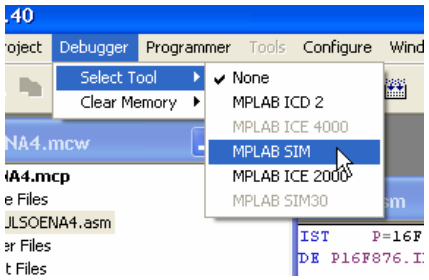



Se muestran en esta ventana los ***errores de sintaxis encontrados, los warnings (atenciones) y mensajes.*** Si hubiera errores habría que arreglarlos antes de proseguir. Haciendo doble clic sobre la línea que señala (y explica) el error, cambia la ventana al editor de texto y ubica una flecha verde en la línea donde se encuentra el error a corregir. Una vez corregido se debe volver a construir hasta que desaparezcan todos los errores y warnings.

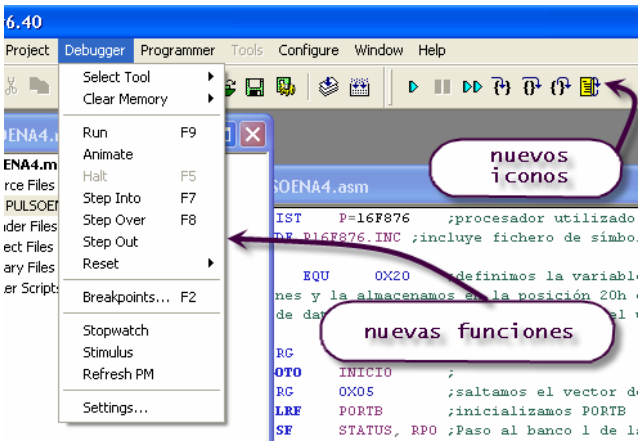
***UTILIZACIÓN DEL MPLAB SIM***

Una vez llegado a este último paso, se puede comenzar a verificar el funcionamiento del Programa. Para ello hay que cambiar la herramienta que utilizamos en el MPLAB, de ***MPASM (ENSAMBLADOR)*** al ***MPLAB SIM (SIMULADOR***).

Seleccionar ***Debugger – Select Tool – MPLAB SIM***.

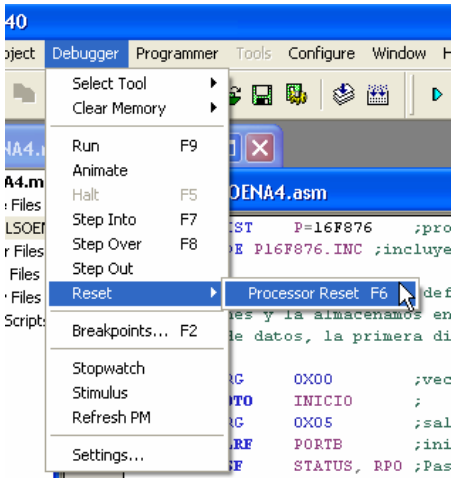



Una vez seleccionado el ***Simulador***, se habilitan otras funciones e iconos.

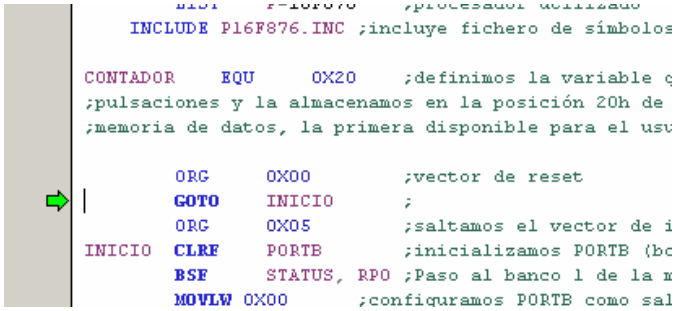


***SIMULACIÓN DEL PROGRAMA.***

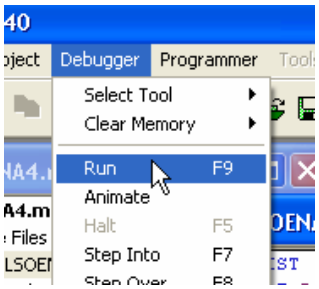
Antes de hacer correr el programa se debe resetear para indicar cual es la primera línea del mismo. Se puede hacer utilizando el menú o con el icono correspondiente. Con el menú se selecciona ***Debugger – Reset – F6***.




En la pantalla donde tenemos el texto del programa aparece una flecha verde indicando la posición de comienzo.

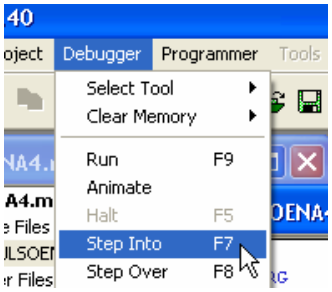



En estos momentos se puede hacer correr el programa, para ello seleccionamos ***Debugger – Run***.



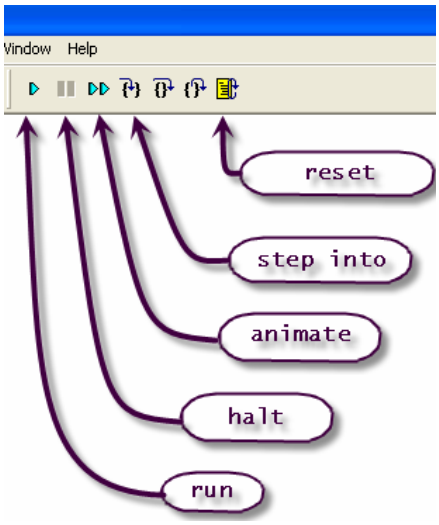
Aparece en el ***status bar*** (barra de estados) ***running***, indicando el estado actual.


Para parar el programa se realiza con ***Debugger – Halt***. Quedará señalizado el punto donde para con la flecha verde. Para realizar la simulación paso a paso se selecciona ***Debugger – Step Intro***. Es muy útil para la depuración del programa. Otra opción, es la utilización de la animación, para ello se selecciona ***Debugger – Animate***. Se observa en forma continua como se va ejecutando el programa a través del movimiento de la flecha verde en el lado izquierdo del programa.



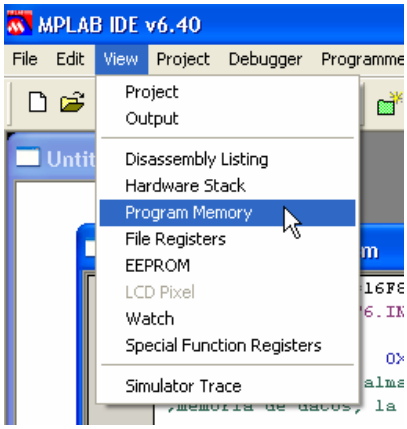

Para todos los casos anteriores se puede utilizar los iconos o las techas abreviadas.

| Debugger Menu | Toolbar Buttons | Hot Key |
|---|---|---|
| Run | | F9 |
| Halt | | F5 |
| Step Into | | F7 |
| Reset | | F6 |

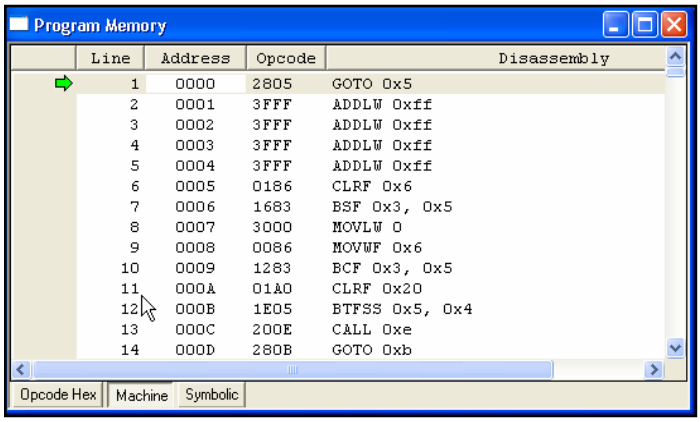



***OBSERVACIÓN DE MEMORIAS (EEPROM Y RAM).***

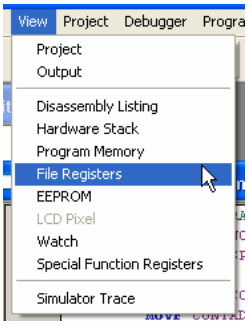
Si deseamos ver el contenido de la memoria de ***programa*** se puede seleccionar ***View – Program Memory***.



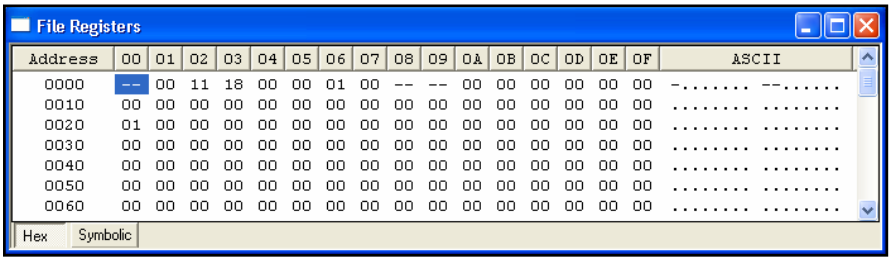

Aparece la ventana



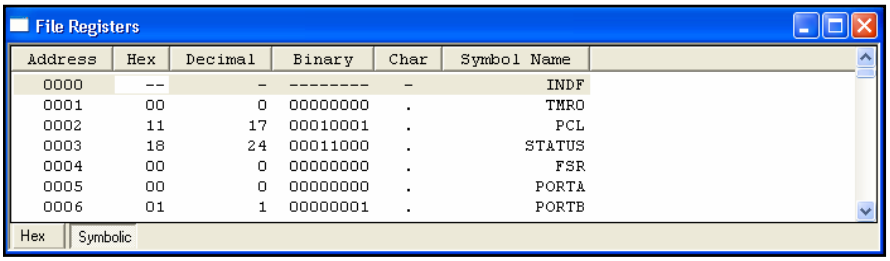

Si en cambio deseamos ver el contenido de la memoria ***RAM***, el procedimiento se similar, seleccionamos ***View – File Registers***.



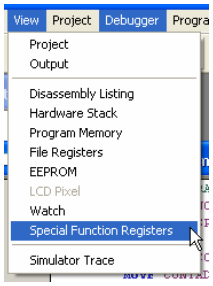

El resultado, será entonces, el mostrado por defecto con la opción ***Hex*** (hexadecimal).



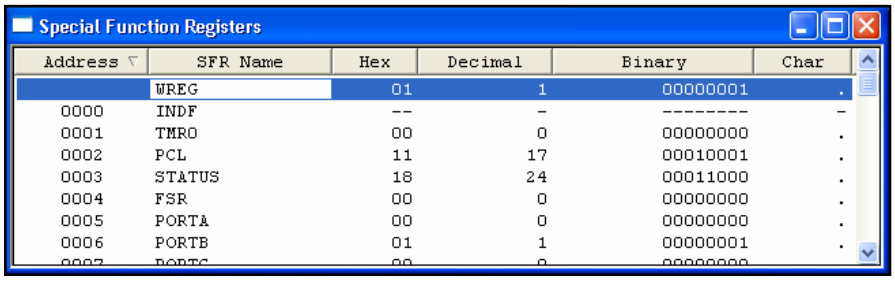

Si ponemos la opción ***Symbolic***, la información mostrada puede llegar a ser más útil, ya que es más fácil reconocer los registros al traer el nombre además de su dirección. Por supuesto, el número de registros mostrados es menor.




Podemos llegar a una ventana similar utilizando la opción ***View – Special Function Registers***. (Prácticamente es la misma tabla, pero tiene el registro de trabajo ***Work Register***)




La ventana que aparece en este caso, es muy similar a la anterior

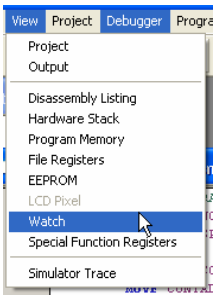



### *VENTANAS A MEDIDA*

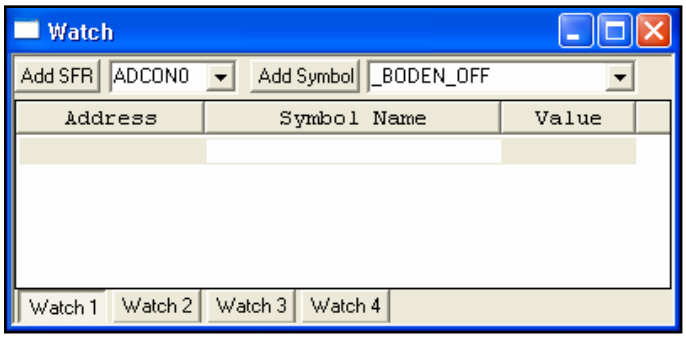
Normalmente, no es práctico tener en una ventana demasiada información, o por lo menos información que no es útil. Si será aconsejable tener una ventana donde podamos visualizar solo los Registros y / o Variables que se modifiquen en el programa y que nos sean de utilidad.

Entonces crearemos una única ventana de visualización a medida para nuestro proyecto. En ésta podremos visualizar Registros, Variables, Contadores, Puertos, etc.

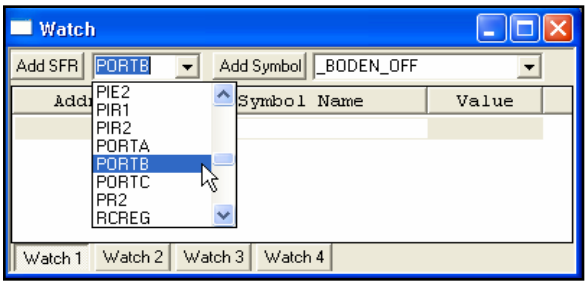
Para abrir esta ventana "***a medida***", seleccionamos ***View- Watch.***



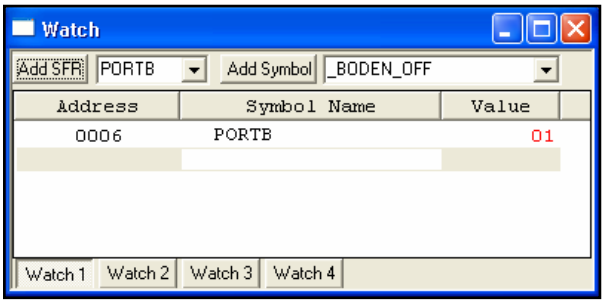

Aparece la ventana siguiente



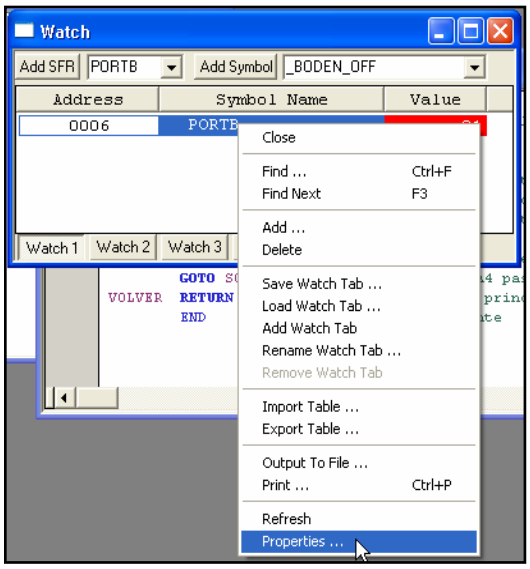

Si lo que deseamos visualizar son ***Variables***, la forma de hacerlo será abriendo la pestaña en ***Add SFR*** y seleccionando cada uno de los Registros que nos interesen



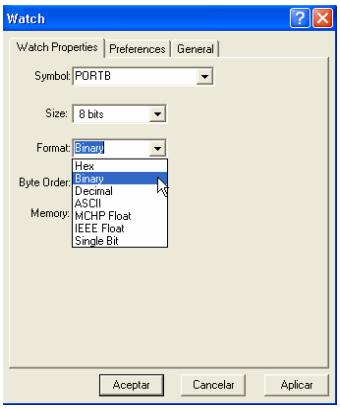

Al teclear ***Add SFR***, se añade el valor seleccionado a la ventana tal como vemos a continuación.



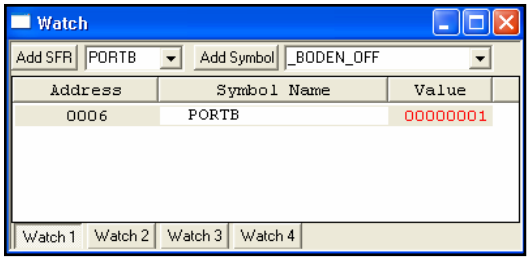

El valor que tiene el registro (señalado en color rojo) está dado en ***hexadecimal***. Si deseamos cambiarlo, por ejemplo en nuestro caso nos interesa tenerlo en binario para saber si los leds se encenderán o no, debemos seleccionar ***PORTB*** y luego el botón derecho del ratón. De la nueva lista abierta seleccionaremos ***Properties*** (Propiedades).



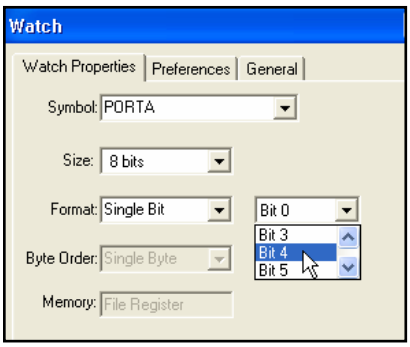

Ahora, cambiamos el valor que viene por defecto (Hex) a ***Binary*** (binario) y aceptamos.



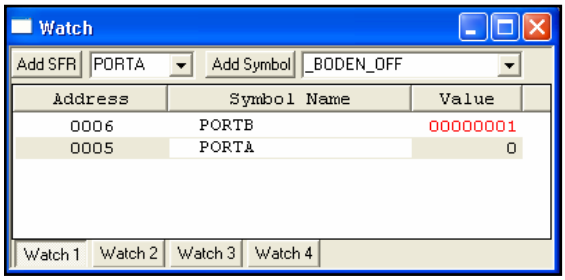

Al ***Aceptar***, aparece nuevamente la ventana de Visualización con el valor del ***Puerto B*** en binario.



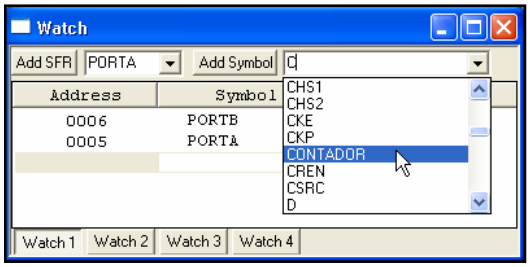

Haremos lo mismo pero ahora con el ***PORTA***, pero en este caso, sólo nos interesa el valor del bit 4. Elegimos la opción de ***Single Bit*** (único bit)



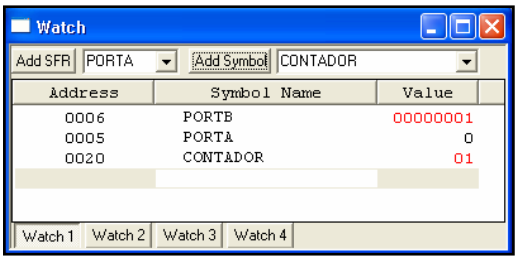

La pantalla de visualización quedará como sigue




Para agregar otras ***Variables***, como por ejemplo ***CONTADOR***, abrimos la pestaña en ***Add Symbol*** hasta encontrarlo, o podemos comenzar a escribir el nombre




Nuevamente lo seleccionamos y hacemos clic en ***Add Symbol***. El valor por defecto nos Lo agrega en Hexadecimal, y lo que haremos es cambiarlo a Decimal.

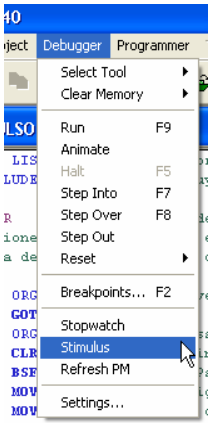



Siguiendo los mismos pasos, creamos la ventana con los valores que nos interesa. Si se mantiene esta ventana abierta mientras que se ejecuta el programa, los valores de los datos mostrados van actualizándose. Si en el paso anterior se modificó algún dato, éste se pone en rojo, en caso contrario permanece en negro.

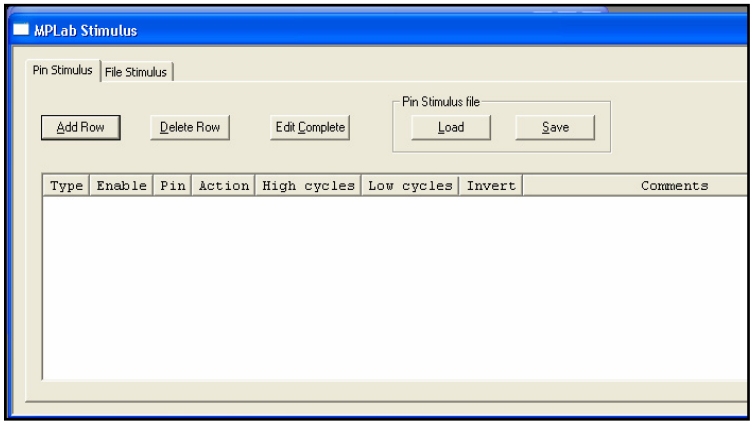
***ESTÍMULOS.***

Para simular mejor las condiciones reales, tales como entradas a nivel alto o bajo de un pin, el simulador provee mecanismos conocidos como estímulos.

El primer tipo de estímulo se llama "***estímulo de pin***" el cual provee estímulos síncronos o asíncronos a las entradas de E/S de los pines. Los ***síncronos*** están sincronizados con los ciclos de instrucción del dispositivo que se está simulando, y los estímulos ***asíncronos*** se aplican por el usuario en tiempo real mientras que se ejecuta la simulación. Solamente veremos los ***estímulos de pin asíncronos***. Seleccionar ***Debugger - Stimulus*** y hacer clic el ***Pin Stimulus tab***.

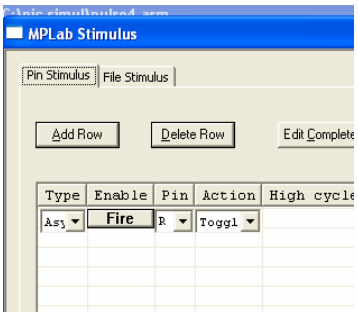



Hacer clic ***Add Row*** para agregar una fila para colocar el dato de entrada.



Si hacemos clic en la fila agregada (debajo de cada columna), seleccionamos las características del pulso, en nuestro caso deseamos poner un pulso asíncrono en la pata RA4 y que cada vez que se pulse cambie de estado.  Entonces seleccionamos

***Async*** para ***Type***, es decir, se elige el tipo de pulso (síncrono o asíncrono) ***Pin*** se abre la pestaña para elegir el RA4. ***Action*** con sus cuatro opciones (poner en uno, en cero, cambio y pulso)

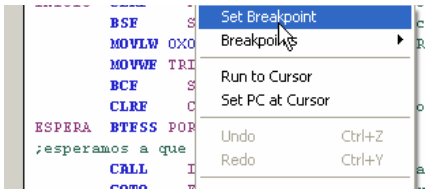



Cada vez que se haga clic sobre la posición ***Fire*** el valor del pulso cambiará de 0 a 1 y de 1 a 0. Si se tiene abierta la ventana de visualización se podrá comprobar los cambios en RA4.

***PUNTOS DE RUPTURA.***

Se pueden colocar puntos de ruptura para que el programa se detenga en otro momento que no sea el final del mismo. Para ello, seleccionar ***Debugger – Reset*** para iniciar la aplicación.

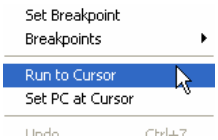
Luego buscar la línea donde se desea agregar el punto de ruptura, hacer clic con el botón derecho y seleccionar ***Set Breakpoint***.




Aparecerá una señal de ***Break (rotura)*** al margen izquierdo cerca de la línea.

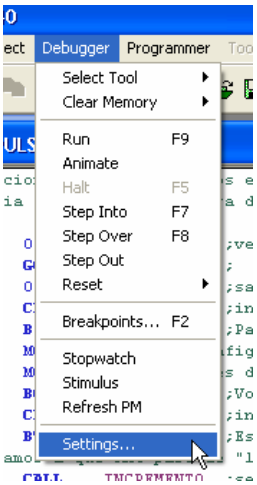

Si seleccionamos ***Debugger – Run***, el programa correrá hasta el punto de ruptura creado.

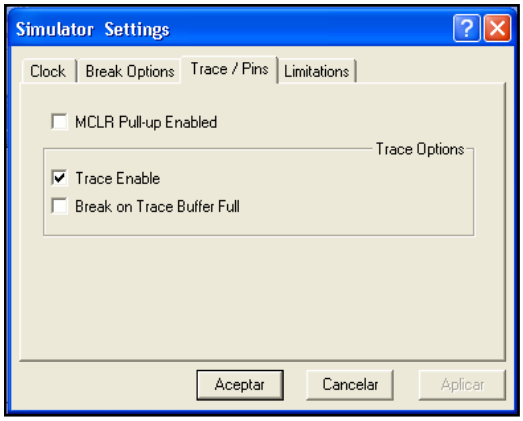

Para salir de esta situación, se coloca el cursor del ratón en alguna línea posterior y con el botón de la derecha del ratón se selecciona ***Run to Cursor***.




***TRACING CODE***

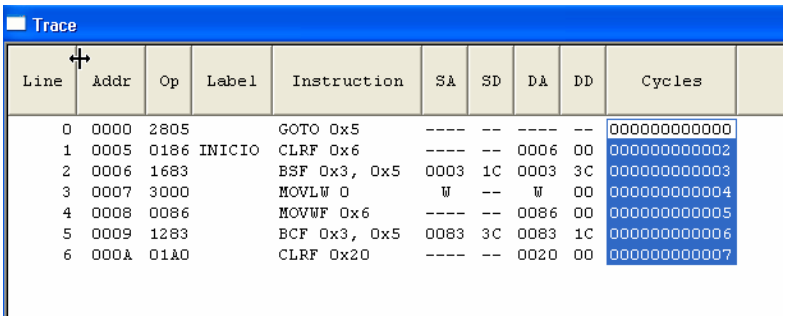
Se utiliza para grabar los pasos de la ejecución del programa. Para habilitarlo se debe seleccionar ***Debugger – Settings*** y luego elegir ***Trace / Pins***.




Hay dos opciones en el área ***Trace Options*** para controlar la simulación. Con la primera opción, ***Trace Enable***, el simulador coge los datos cuando funciona en modo ***Run***. Recoge datos hasta que se detiene en un punto de ruptura o manualmente. Muestra como máximos 8192 ciclos. Se utiliza para ver las instrucciones hasta un punto de ruptura.

Si se selecciona también la segunda opción, la memoria guardará otros 8192 ciclos más de datos, luego para de recogerlos y se detiene en un punto de ruptura. Este modo se utiliza para ver las instrucciones ejecutadas y grabadas luego de seleccionar un ***Run***. Para visualizar estas instrucciones se debe seleccionar ***View – Simulator Trace***.




Como se observa en la ventana, se pueden verificar otros datos:

***Line.*** Número de línea

***Addr.*** Dirección del contador de programa.

***Op.*** Código de la instrucción

***Label***. Etiqueta

***Instruction.*** Instrucción sin ensamblar

Valores leídos o escritos en los registros:

***SA Sourcce Address*** dirección del registro de operaciones de lectura

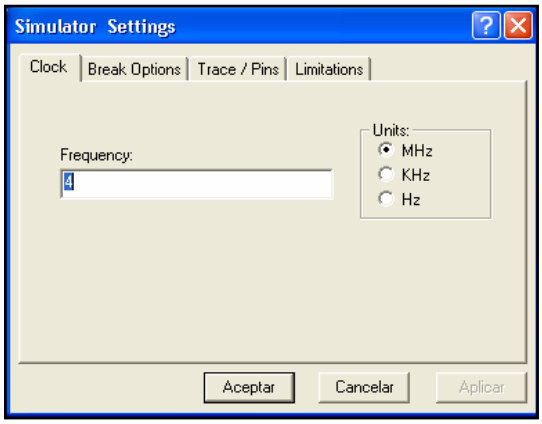
***SD Source Data*** dato leído del registro

***DA Destination Address*** destino del registro en operación de escritura

***DD Destination Data*** dato escrito en el registro

**--** No hubo acceso a ningún registro en la instrucción

***Cycles*** número de ciclos que ha llevado la ejecución del programa. Se utiliza para medir el tiempo de ejecución de las rutinas. El tiempo se calcula basado en la frecuencia de reloj que se colocó en ***Debugger – Settings – Clock Tab***.




Para guardar y cerrar el proyecto y campo de trabajo se deben seguir los siguientes pasos:

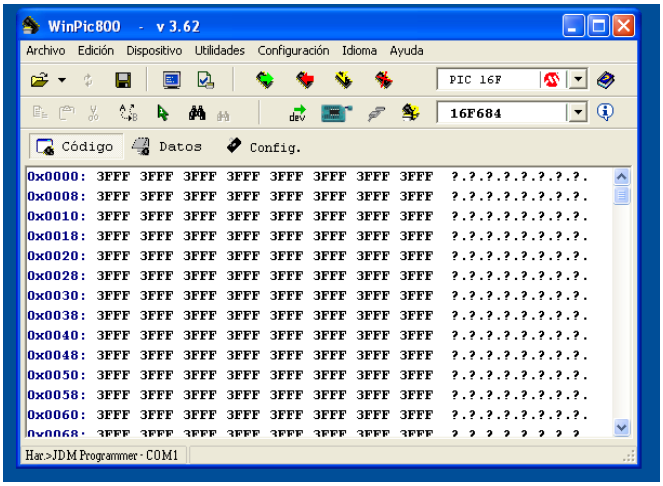
***Select Project - Save Project Select File – Workspace - Save Select File – Workspace - Close***

# Software WinPic800 Grabando un PIC.

El software WinPic800 (Se lo puede bajar de http://www.winpic800.com/ ), soporta los Pic nuevos, por ejemplo el Pic 18F2550, es freeware y lo podemos usar con este Programador de Pic y Memorias.

Para instalar el software WinPic800, bastará con ejecutar el archivo Winpic800.exe, se instalara en la carpeta C:\Winpic800.

Una vez hecho esto ejecutar el EXE con doble clic y establecer la configuración del hardware.




Lo primero es asegurarnos de que el WinPIC800 este correctamente configurado. Para ello dispone en el menú principal de la opción “Configuración”. En “Hardware” nos aseguraremos que el programador elegido sea el nuestro (figura 1).

Para configurar el programador vamos a Hardware del menú desplegable Configuración.

- En Selección del Hardware elegimos JDM Programmer.
- Sacamos la tilde de la casilla de Bloqueo configuración, para habilitar el acceso a la configuración.
- Vemos que este marcado la opción COM, que es el Puerto de comunicación.
- Ahora en Data tildamos la casilla Inv., cambiara de color el Estado. Se encenderá el Led Rojo en el Programador. Y pulsamos el botón Confirmar cambios.

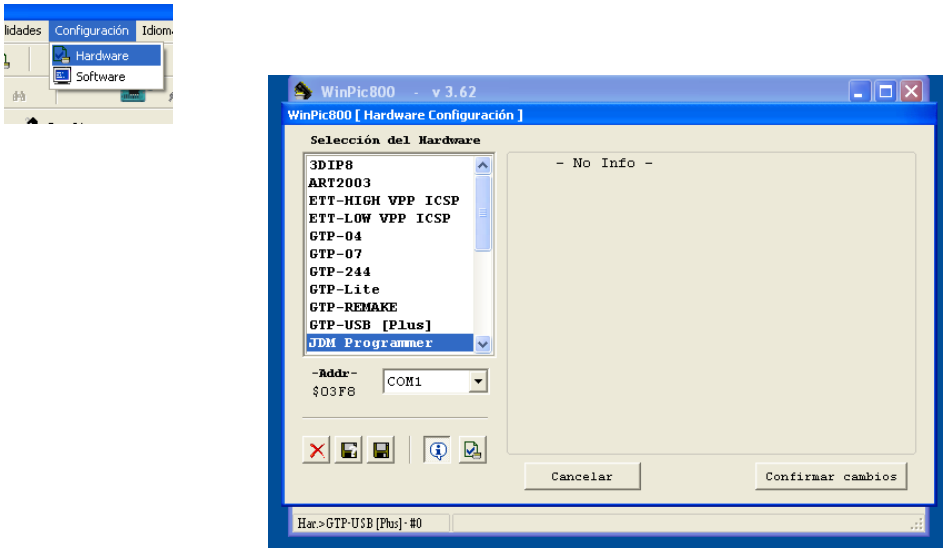



**Figura 1: Seleccionamos nuestro programador.**

En “Software” hay una serie de solapas y opciones (figura 2) que básicamente configuran los mensajes que recibiremos (o no) al utilizar el programa. En general, las opciones por defecto funcionarán correctamente para todos.

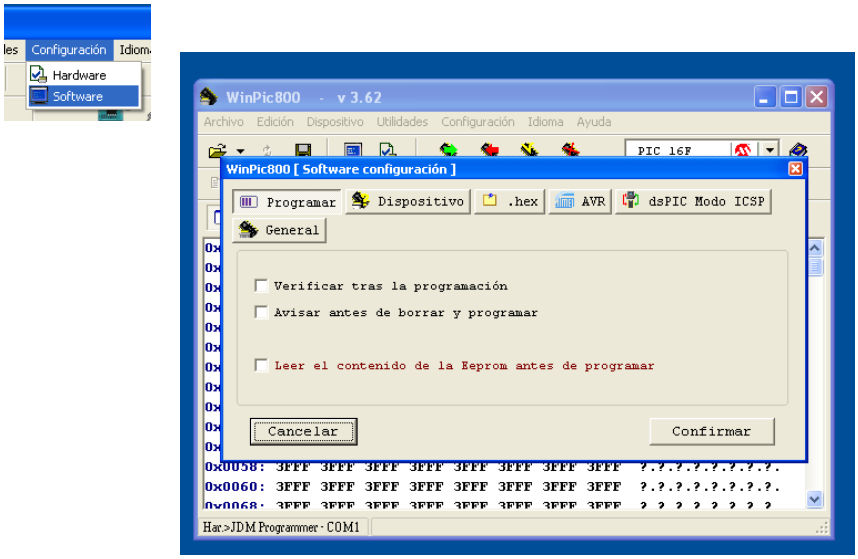



**Figura 2: las opciones por defecto funcionarán correctamente para todos.**

La figura 3 ilustra el paso siguiente: desde las listas que están a la derecha de la ventana principal del WinPIC800 seleccionamos la familia y modelo del micro controlador que vamos a utilizar. Este debe coincidir con el que seleccionamos en el MPLAB IDE, ya que el programa que se generó está especialmente concebido para ese modelo en particular. Como familia seleccionamos “PIC 16F” y como modelo “16F84”.

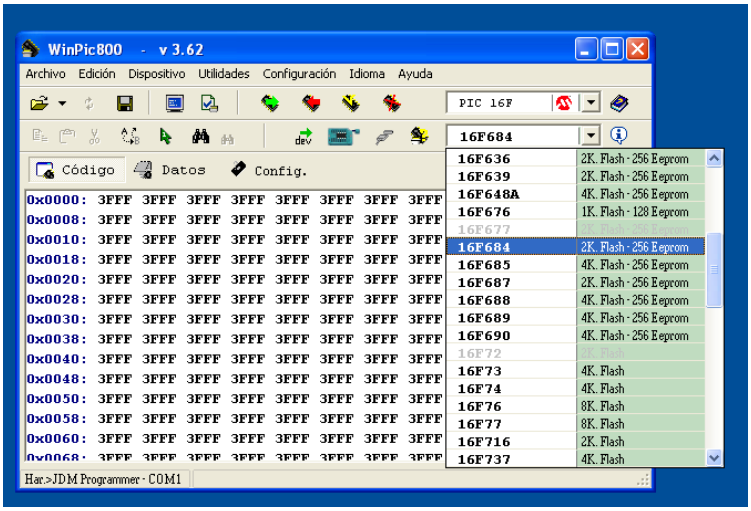



**Figura 3: seleccionamos la familia y modelo del micro controlador.**

Una vez que hemos hecho esto, WinPIC800 “sabe” como deberá enviar los datos al programador. Otro punto a tener en cuenta en esta etapa del proceso es la posición que debe ocupar el PIC en el zócalo ZIF del programador. Si tenemos dudas, podemos utilizar la ayuda incorporada en el programa, mediante la opción marcada con un círculo rojo en la figura 4. Luego, debemos ir al menú “Archivo” --> “Abrir” y cargar el fichero HEX que generamos con el MPLAB IDE.

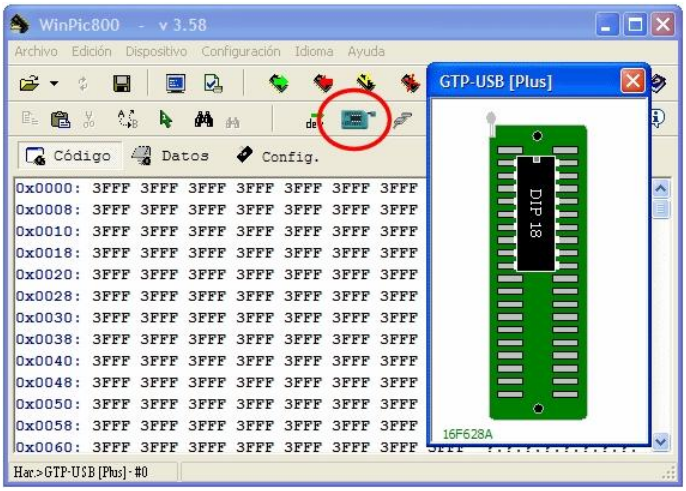



**Figura 4: Posición del PIC en el zócalo ZIF.**

Antes de proceder a realizar el grabado del programa en el micro controlador podemos configurar algunas de las características del mismo para su funcionamiento. Como por ejemplo: el tipo de reloj, activar/desactivar las alarmas de Reseteo del micro controlador, etc.

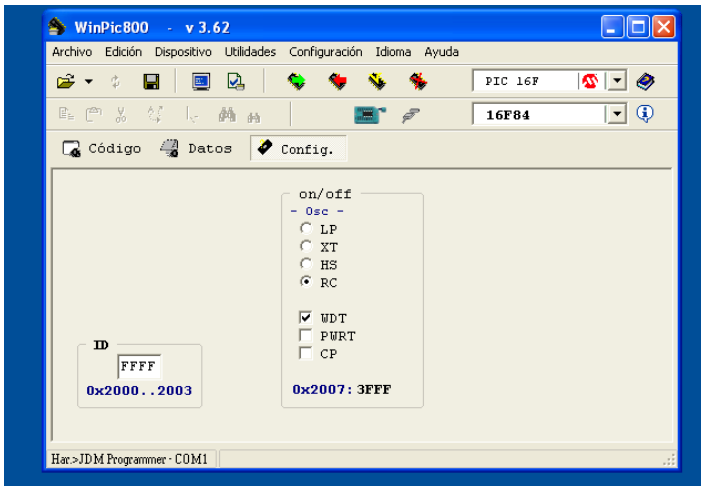



**Figura 5: Configuración de algunas características.**

El led del programador JDM estará en rojo si todo esta correctamente instalado, por lo que podemos proceder a enviar el fichero. Para ello, presionamos el icono “Grabar Todo” que se ve en la figura 6.

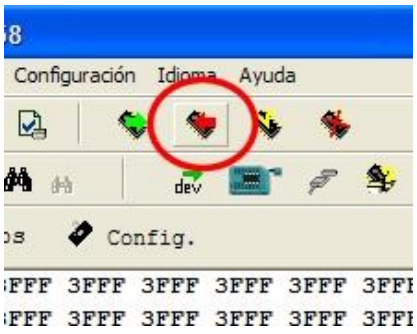



**Figura 6: el icono “Grabar Todo”.**

Y en un par de segundos tenemos nuestro PIC grabado. El mensaje que veremos será el de la figura 7.


**Figura 7: ya tenemos nuestro PIC grabado.**

Para probar que todo funciona, tenemos que armar el circuito y alimentarlo con 5V de corriente continua.

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
MICROPROGRAMACIÓN

# GUÍA DE PRÁCTICA Nº 2
# ENCENDER UN DIODO LED EN UNA SALIDA.

## *OBJETIVOS.*

- Utilización de un Diodo Led como dispositivo de visualización de salida.
- Verificar el modo en el que se debe programar el sentido de los puertos.
- Escribir sobre un puerto de salida visualizando sobre un diodo Led.

## *METODOLOGÍA.*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.
1 Vom (voltímetro, óhmetro y amperímetro).
1 Regleta protoboard.
1 Fuente de cinco voltios.
1 Micro controlador Pic 16F84.
1 Cristal de 4 Mhz
Resistencias (1KΩ - 5kΩ - 100Ω - 220Ω).
1 Diodo Led.

## *GENERALIDADES.*

### *PUERTOS DE ENTRADA/SALIDA*

Los puertos son entradas y salidas del micro controlador al exterior, por ellas enviamos o introducimos señales digitales TTL (5V) de forma que podemos comunicar el micro controlador con el exterior. En este caso utilizaremos 2 puertos uno de entrada y otro de salida (E/S). Sus nombres son RA y RB.

El puerto RA tiene 5 pines RA0-RA4, un caso particular es RA4/TOCK1 que puede actuar como pin de entrada o como entrada de impulsos para un contador denominado TMRO; El puerto B tiene 8 líneas que van desde RB0-RB7. Cada línea de los puertos que corresponden a las patillas RA0-RA4 y RBO-RB7 Pueden ser configurados como entradas o salidas mediante 2 registros llamados TRISA y TRISB.

PORTA (05H) BANK0 BITS RA4-RA3-RA2-RA1-RA0 --> TRISA BANK1

PORTB (06H) BANK0 BITS RB7-RB6-RB5-RB4-RB3-RB2-RB1-RB0 --> TRISB BANK1

Vemos como existen 2 bancos, BANK0 y BANK1 para pasar de un banco a otro utilizamos el registro STATUS (03H) mediante el bit 5 de ese registro, de forma que si el bit 5 es 0 estamos en el banco 0 y si el bit 5 es 1 pasaremos al banco 1.

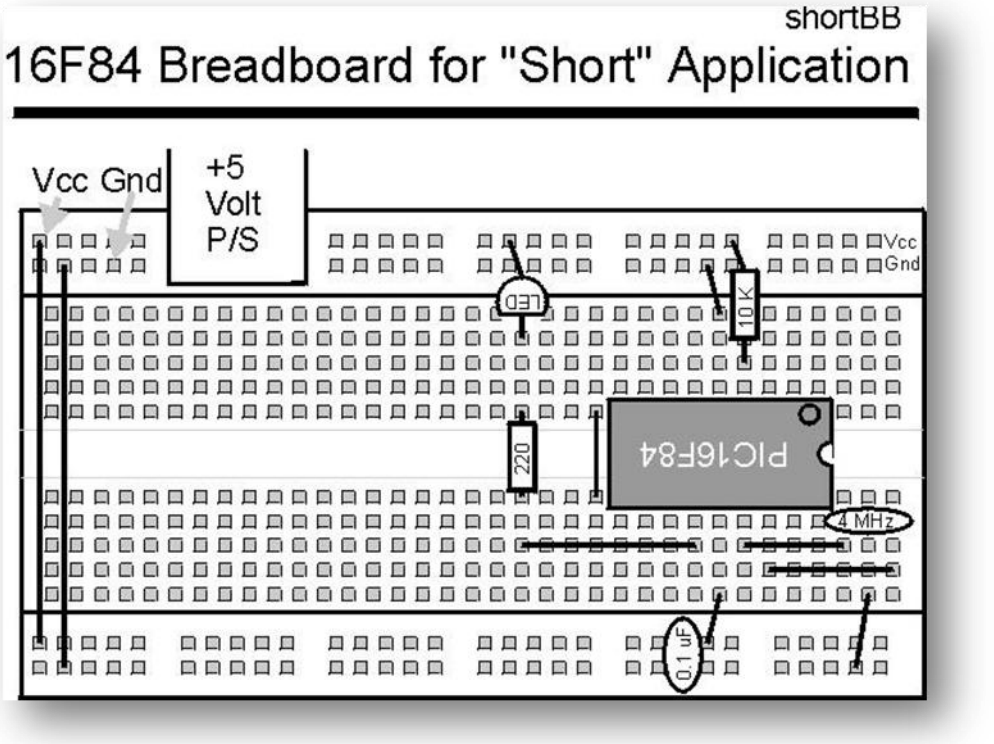
El proceso para configurar los puertos es el siguiente:

1. Al arrancar el micro controlador el banco por defecto es BANK0
2. Accedemos al banco1 BANK1 poniendo a 1 el bit5 del registro STATUS (03h)
3. Cargar en TRISA los valores adecuados (0 salidas 1 entradas)
4. Cargar en TRISB los valores adecuados (0 salidas 1 entradas)
5. Acceder al banco 0 BANK0 poniendo a 0 el bit 5 del registro STATUS
6. Realizar el programa para activar las entradas y salidas

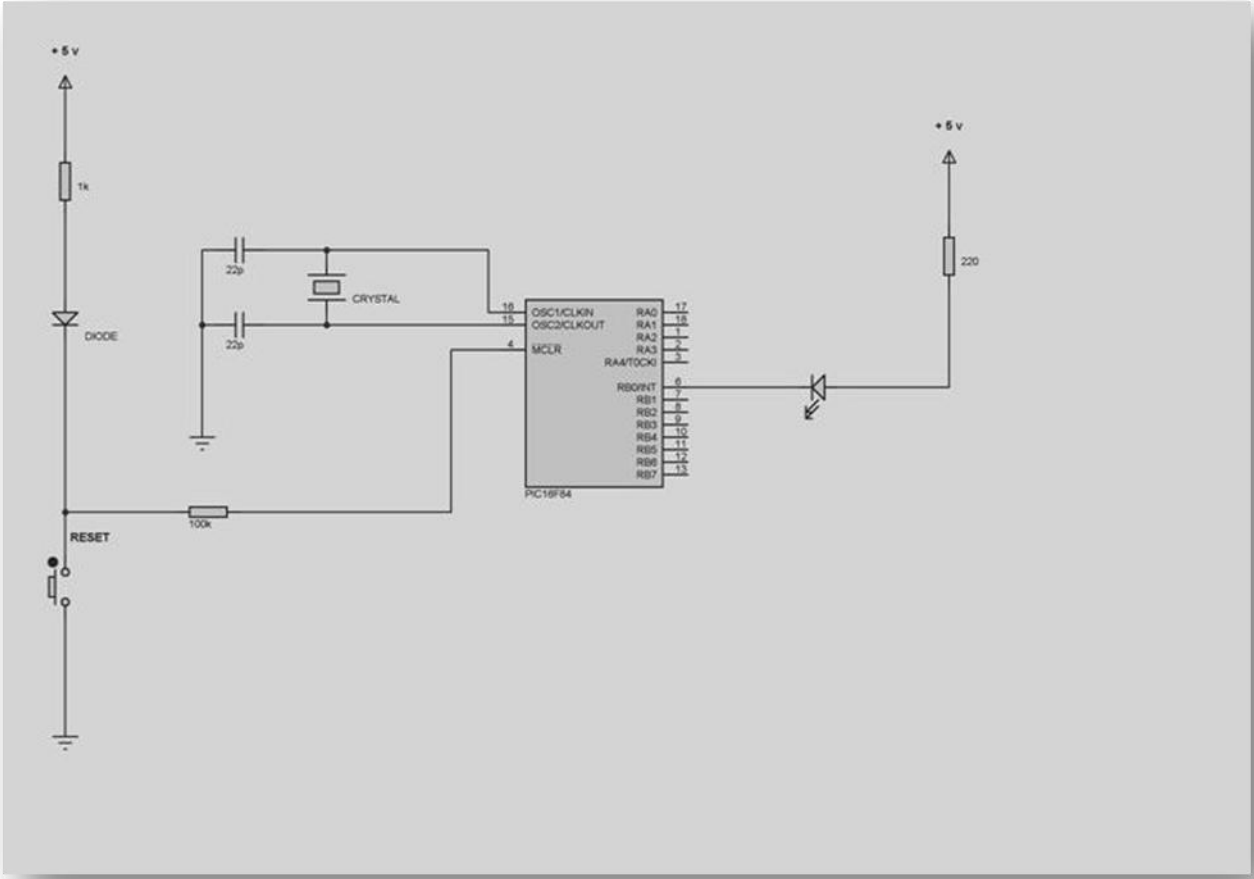
***PROCEDIMIENTO.***

Este es el primer programa ideal para quemar en un Pic y luego ejecutarlo en un entrenador, el programa simplemente configura el bit 0 del puerto "B" como salida.

Construya el siguiente esquema de conexión del circuito se activará un diodo LED conectado a RB0 como se ve en el y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construccion del proyecto (.Hex) utilizelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.



Esquema del circuito.



## *CODIGO FUENTE*

```
LIST P=16F84, R=DEC

INCLUDE "p16F84.inc"

__CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

PAGE

; Mainline of RegAddr

org 0

clrf PORTB ; limpia el Port "B"

bsf STATUS, RP0 ; Banco 1

bcf TRISB ^ 0x080, 0 ; configura RB0 como salida

bcf STATUS, RP0 ; Banco 0

Finished

goto $

End
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR.

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE.

INGENIERÍA EN SISTEMAS INFORMÁTICOS.

DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA.

MICROPROGRAMACIÓN.

**GUÍA DE PRÁCTICA Nº 3**

**ENCENDER UN LED EN UNA SALIDA A TRAVÉS DE UN PULSADOR EN UNA ENTRADA.**

## *OBJETIVOS.*

- Utilización de un Diodo Led como dispositivo de visualización de salida para que responda a los cambios de estado basado en sus entradas (Switch).
- Verificar el modo en el que se debe programar el sentido de los puertos.
- Realizar la entrada por un puerto mediante la lectura de interruptor-switch-pulsador.
- Escribir sobre un puerto de salida visualizando sobre un diodo Led.

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.

1 Vom (voltímetro, óhmetro y amperímetro).

1 Regleta protoboard.

1 Fuente de cinco voltios.

1 Micro controlador Pic 16F84.

1 Cristal de 4 Mhz

Resistencias (1KΩ - 5kΩ - 100Ω - 220Ω).

1 Diodo Led.

1 Switch (pulsador).

## *GENERALIDADES.*

### *PUERTOS DE ENTRADA/SALIDA*

Los puertos son entradas y salidas del micro controlador al exterior, por ellas enviamos o introducimos señales digitales TTL (5V) de forma que podemos comunicar el micro controlador con el exterior. En este caso utilizaremos 2 puertos uno de entrada y otro de salida (E/S). Sus nombres son RA y RB.

El puerto RA tiene 5 pines RA0-RA4, un caso particular es RA4/TOCK1 que puede actuar como pin de entrada o como entrada de impulsos para un contador denominado TMRO; El puerto B tiene 8 líneas que van desde RB0-RB7. Cada línea de los puertos que corresponden a las patillas RA0-RA4 y RBO-RB7. Pueden ser configurados como entradas o salidas mediante 2 registros llamados TRISA y TRISB.

PORTA (05H) BANK0 BITS RA4-RA3-RA2-RA1-RA0 --> TRISA BANK1

PORTB (06H) BANK0 BITS RB7-RB6-RB5-RB4-RB3-RB2-RB1-RB0 --> TRISB BANK1

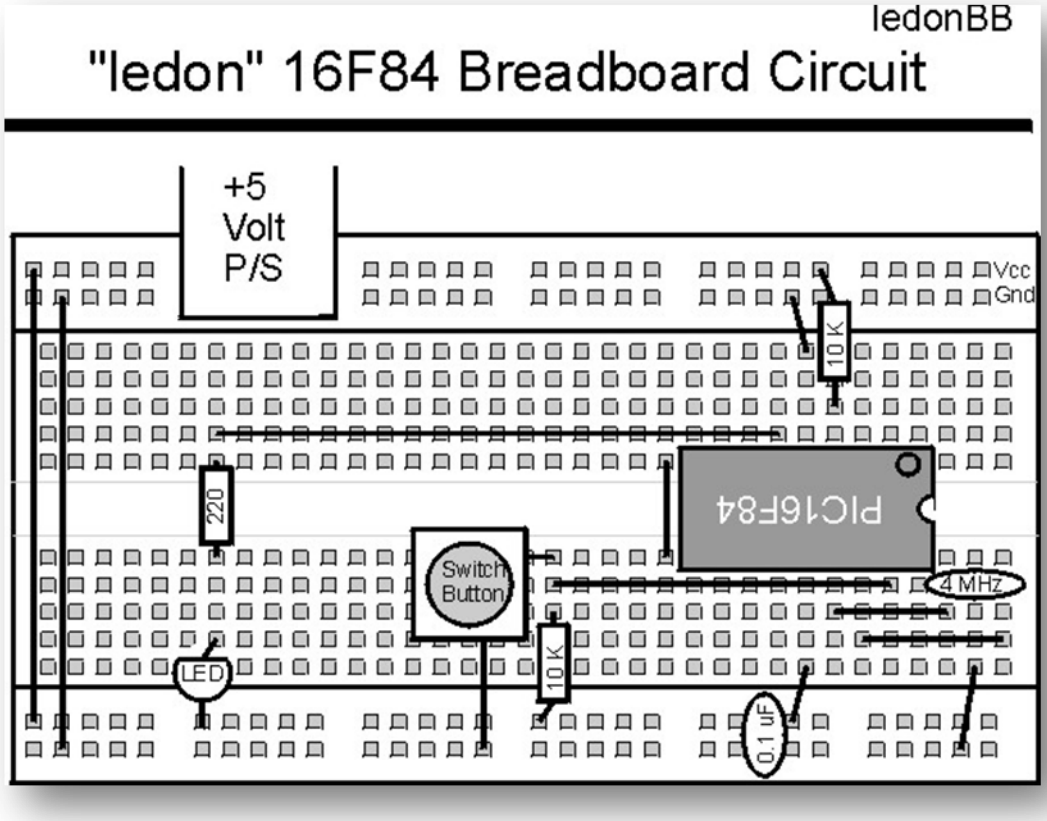
Vemos como existen 2 bancos, BANK0 y BANK1 para pasar de un banco a otro utilizamos el registro STATUS (03H) mediante el bit 5 de ese registro, de forma que si el bit 5 es 0 estamos en el banco 0 y si el bit 5 es 1 pasaremos al banco 1.

El proceso para configurar los puertos es el siguiente:

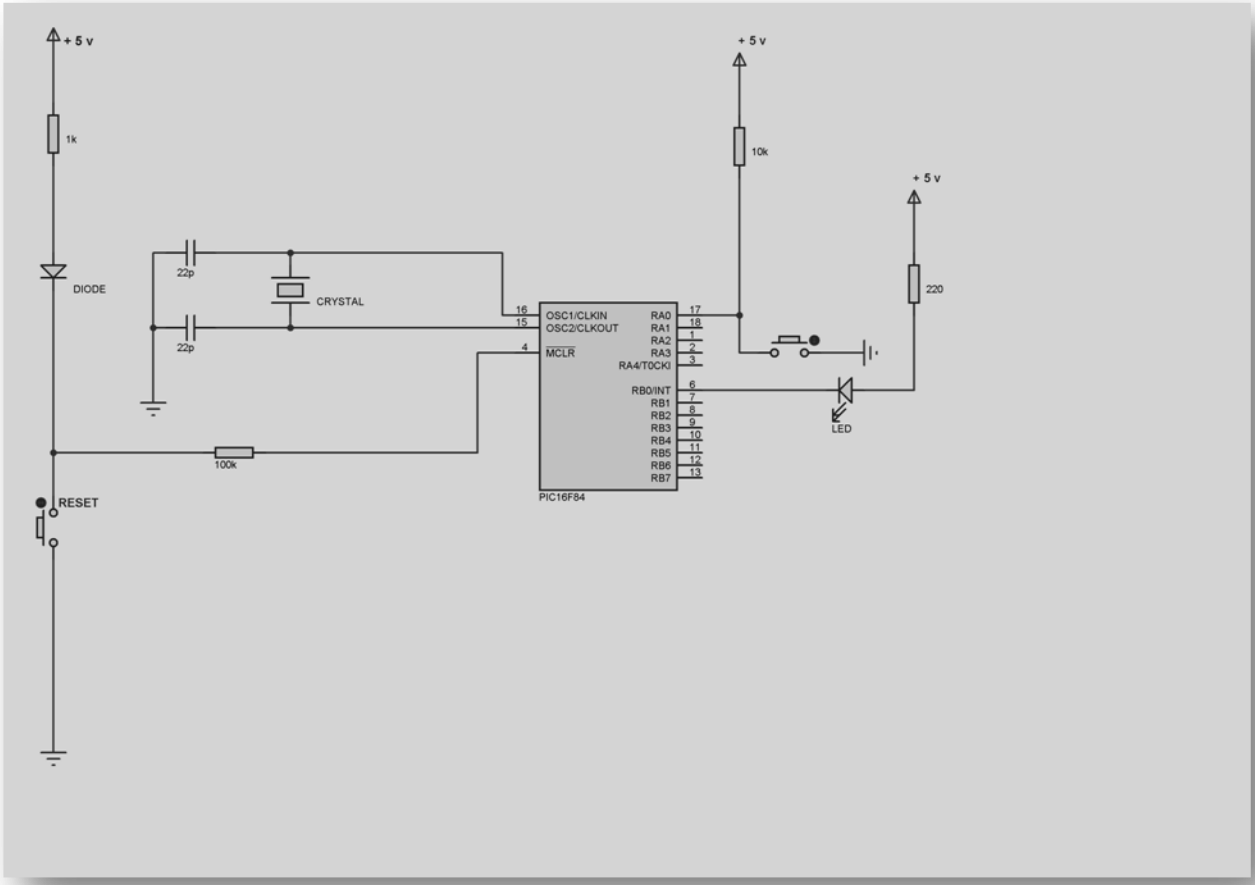
1. Al arrancar el micro controlador el banco por defecto es BANK0
2. Accedemos al banco1 BANK1 poniendo a 1 el bit5 del registro STATUS (03h)
3. Cargar en TRISA los valores adecuados (0 salidas 1 entradas)
4. Cargar en TRISB los valores adecuados (0 salidas 1 entradas)
5. Acceder al banco 0 BANK0 poniendo a 0 el bit 5 del registro STATUS
6. Realizar el programa para activar las entradas y salidas

***PROCEDIMIENTO.***

Construya el siguiente esquema de conexión del circuito se activará un diodo LED conectado a RB0 siempre que el Pulsador conectado a RA0 este cerrado. Utilizaremos el puerto A, y lo haremos con un solo pulsador el pulsador lo conectaremos a RA0 y el diodo LED a RB0, así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construccion del proyecto (.Hex) utilizelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.



Esquema del circuito.




En el circuito podemos ver como lo único que hemos añadido es un pulsador conectado al pin 17 (RA0), de forma que cuando lo pulsemos se introduzca un cero lógico en el pin y cuando no lo pulsemos se introduzca un uno lógico. Hemos añadido además un LED con su correspondiente resistencia limitadora de corriente en el pin 6 (RB0).

## *CÓDIGO FUENTE*

```
  LIST P=16F84, R=DEC
        INCLUDE "P16F84.inc"
        __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

 PAGE; Mainline of ledon
 Org     0
 Goto Reset
 Org 5
 Nop

Reset bsf    PORTB, 0              ; PORTB.0 "off"
       Bsf    STATUS, RP0          ; Bank 1
       Bcf    TRISB ^ 0x080, 0     ; RB0 to Output
       Bcf    STATUS, RP0          ; Bank 0
Loop
        Movf   PORTA, w             ; Transferencia PORTA.0 a PORTB.0
        Movwf PORTB
        Goto   Loop
       End
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR.

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE.

INGENIERÍA EN SISTEMAS INFORMÁTICOS.

DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA.

MICROPROGRAMACIÓN.

# GUÍA DE PRÁCTICA Nº 4
# ENCENDER CUATRO LEDS EN FORMA SECUENCIAL

## *OBJETIVOS.*

- Utilización de un Diodo Led como dispositivo de visualización de salida.
- Utilización de las instrucciones de manipulación y desplazamiento de bits de un registro.
- Utilización de las instrucciones de manipulación y control de secuencia del flujo del programa (salto a una Subrutina).
- Verificar el modo en el que se debe programar el sentido de los puertos.

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.

1 Vom (voltímetro, óhmetro y amperímetro).

1 Regleta protoboard.

1 Fuente de cinco voltios.

1 Micro controlador Pic 16F84.

1 Cristal de 4 Mhz

Resistencias (1KΩ - 5kΩ - 100Ω – 220RΩ).

4 Diodo Led.

## *GENERALIDADES.*

### *IMPORTANCIA DE LAS RUTINAS*

La mayoría de los micro controladores incluyen en su repertorio de instrucciones algunas que permiten saltar a una rutina y, cuando se complementa su ejecución, retornar al programa principal

El empleo de subrutinas aporta muchas ventajas entre las que se destacan las siguientes:

1. Se pueden escribir como subrutinas secciones de código y ser empleadas en muchos programas (por ejemplo, la subrutina de exploración de un teclado).
2. Dan a los programas un carácter modular, es decir, se pueden codificar diferentes módulos para usarlos en cualquier programa.
3. Se reduce notablemente el tiempo de programación, la detección de errores, usando repetidamente una subrutina.
4. El código es más fácil de interpretar, dado que las instrucciones de las subrutinas no aparecen en el programa principal. Solo figuran las llamadas CALLs.

### *LAS INSTRUCCIONES CALL Y RETURN*

La instrucción CALL (llamada la subrutina) consigue que la ejecución del programa continúe en la dirección donde se encuentra la subrutina a la que hace referencia. Es similar a GOTO pero coloca en la pila la dirección de la siguiente instrucción que se debe ejecutar después de la CALL.

La subrutina finaliza con la instrucción RETURN (Retorno de la subrutina) que retoma la dirección guardada en le pila y la coloca en el contador del programa PC continuando el flujo de control con la instrucción que sigue a la CALL.

### *INSTRUCCIONES DE ROTACIÓN DE BITS*

**RLF f, d; rotación a la izquierda, destino**

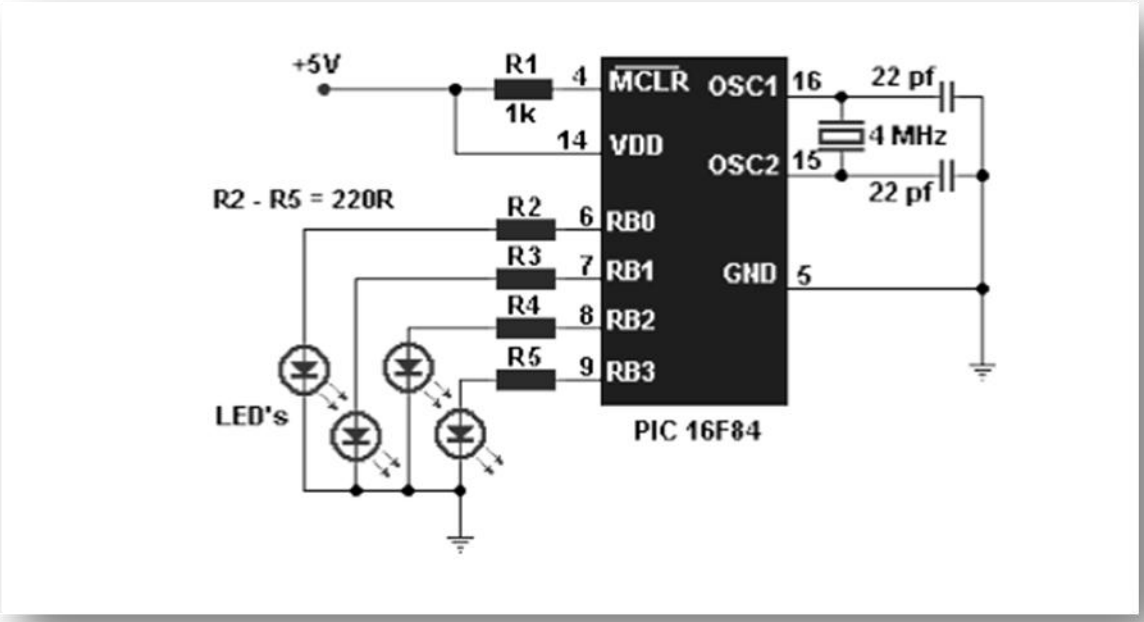
**RRF f, d; rotación a la derecha, destino**

En estas operaciones (Rotate Left File y Rotate Right File) los bits son desplazados de cada posición a la siguiente, en sentido derecho o izquierdo. El desplazamiento es cerrado, formando un anillo.

***PROCEDIMIENTO.***

Construya el siguiente esquema de conexión del circuito que encienda 4 diodos Led's en forma secuencial, y para ello recurriremos a una llamada de una rutina de retardo, así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construccion del proyecto (.Hex) utilizelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.

Esquema del circuito.




En el circuito podemos ver que hemos añadido 4 diodos leds (RB0-RB3) conectado al puerto B además hemos añadido a cada diodo led su correspondiente resistencia limitadora.

***CÓDIGO FUENTE***

```
; --------------- Encabezado ----------------------------------------------------
          LIST P=16F84, R=HEX
          INCLUDE "p16F84.inc"
         __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON
  PAGE
; ------------ Mapa de memoria ----------------------------------------------------
Rotar           equ 0x0A                    ; para desplazar el dato
reg1    equ 0x0C                    ; para hacer el retardo
reg2    equ 0x0D
reg3    equ 0x0E
; ------- Configuración de puertos ---------------------------------------------------
RESET  Org 0x00
        Goto INICIO                 ; salta a "INICIO"
        Org 0x05                    ; aquí comienza el programa
INICIO bsf    STATUS, RP0           ; Bank1
        Movlw b'00000000'
        Movwf TRISB ^ 0x080         ; configura RB como salida
        Bcf    STATUS, RP0          ;  banco 0

; ---- Realiza la secuencia de Led's ---------------------------------------------------
AHORA Movlw 0x01                    ; carga W con 00000001
          Movwf Rotar                       ; lo pasa al registro ROTAR
ROTANDO Movf Rotar, 0               ; pasa el contenido de ROTAR a W
        Movwf PORTB                 ; y de allí al puerto B
```

```
        Call RETARDO            ; llama a RETARDO
        Rlf Rotar, 1            ; desplaza un bit el contenido
                                ; De Rotar y lo guarda.
        Btfss Rotar, 4          ; prueba si se activa el 5to. Bit
                                ; Si es así saltea una línea
        Goto ROTANDO            ; sino, sigue ROTANDO
        Goto AHORA              ; repite todo de nuevo
; ----------- Rutina de Retardo ----------------------------------------
RETARDO Movlw 10                ; Carga los registros
        Movwf reg1              ; reg1, reg2 y reg3
                                ; Con los valores 10, 20 y 30
TRES    Movlw 20                ; respectivamente
        Movwf reg2
DOS     Movlw 30
        Movwf reg3
UNO     decfsz reg3, 1          ; Comienza a decrementar
        Goto UNO                ; cuando termine
        Decfsz reg2, 1          ; regresará a la siguiente
        Goto DOS                ; línea de código
        Decfsz reg1, 1          ; de donde fue llamado
        Goto TRES
        Retlw 00

Finished
        End
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
MICROPROGRAMACIÓN

# GUÍA DE PRÁCTICA Nº 5
# TRABAJANDO DIRECTAMENTE CON EL DISPLAY (SIN DECODIFICADOR)

## *OBJETIVOS.*

- Utilización de un Display de 7 segmentos (Ánodo común) como dispositivo de visualización.
- Utilización del registro **PLC (contador del programa)** para controlar el flujo de ejecución del programa (puesto que asigna un número a cada línea de código).
- Crear el Codificador de Binario a Decimal **(BCD)** por medio de código y el encendido de los segmentos del display, se hará activándolos desde el micro controlador.
- Utilización de una tabla de instrucciones para generar una rutina de visualización **(RETLW).**

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.
1 Vom (voltímetro, óhmetro y amperímetro).
1 Micro controlador Pic 16F84.
1 Cristal de 4 Mhz
1 Display de 7 segmentos (Ánodo común).
Resistencias (10K – 150R).

## *GENERALIDADES.*

### *El Registro PCL.*

Veamos como trabaja el Micro Controlador cuando se encuentra ante una serie de instrucciones. Existe un registro, llamado PCL, ubicado en la posición 0x02 en el banco de memoria, tiene mucho que ver con el flujo del programa, puesto que le asigna un número a cada línea de código. Todo empieza con la primera instrucción, esta tiene una posición indicada con un número en el registro PCL, cuando accede a esa posición, se lee la instrucción, se decodifica, y luego se ejecuta, una vez echo esto, el reloj del Micro controlador incrementa al contador de programa (PCL) en un unidad, esto hace que el PCL apunte a la segunda instrucción, ahora se lee esta segunda instrucción, se decodifica y también se ejecuta. Nuevamente, el reloj del sistema incrementa el PCL para que apunte a la tercera instrucción, la decodifique y la ejecute. Este proceso se repite hasta que termina el programa (es decir, cuando encuentra un END).

## *Las Tablas:*

Me imagino que sabes lo que es una tabla, bueno, una tabla es algo como esto...

| Cont. de Programa | ISNT. | DATO |
|---|---|---|
| **PCL=11 »** | RETLW | 11000000 |
| **PCL=12 »** | RETLW | 11100001 |
| **PCL=13 »** | RETLW | 00001111 |
| **PCL=14 »** | RETLW | 00111001 |

Te preguntarás por el contenido de esta tabla, bueno, en esta tabla, cada línea horizontal, es una línea de código, y la dirección de cada línea, está dada por el valor del PCL (el contador de programa), suponte ahora el siguiente código...RETLW 00001111

RETLW, es retornar cargando W con el Literal 00001111, el problema es que para llegar a esta instrucción deberías pasar por encima de las dos líneas anteriores.

La primera instrucción ADDWF PCL, F indica que se le debe sumar al registro PCL, lo que hay en W. Con F, le indicamos que guarde el resultado en el mismo registro PCL, es decir...

***PCL = PCL + W***

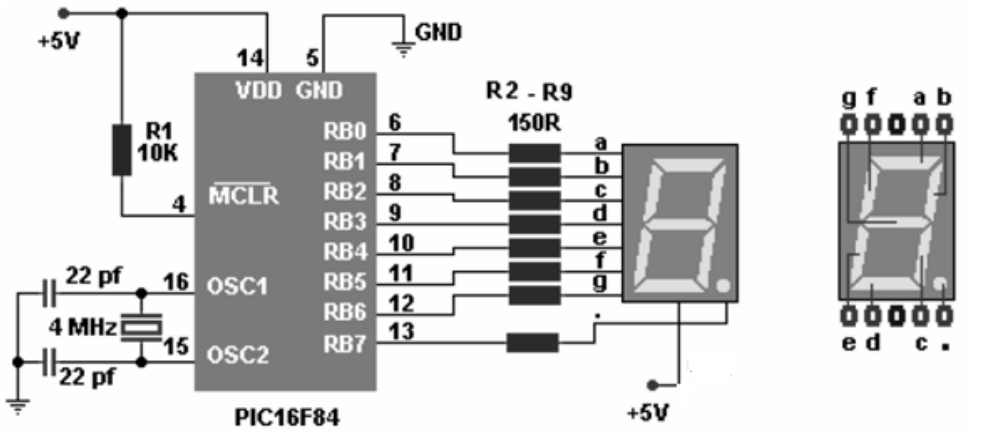
El acceso a la tabla lo haremos a través de W, le cargamos un valor y llamamos a la tabla, justo donde al PCL se le suma el valor de W.

## *PROCEDIMIENTO.*

Construya el siguiente esquema de conexión del circuito que haga el Micro Controlador envíe unas cuantas señales por su propia cuenta con un pequeño retardo, lo que haremos será una cuenta regresiva de 5 a 0 y luego haremos que escriba ÚES. (Con el puntito incluido), así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construcción del proyecto (.Hex) utilícelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.

Esta vez, el decodificador, deberemos crearlo nosotros, por medio de código, y el encendido de los segmentos del Display, se hará activándolos desde el micro controlador. Para que tengas una idea, cuando el Micro Controlador se encienda por primera vez, el display deberá encender los 5 segmentos que corresponden al número 5, y luego comenzar la secuencia, dejaremos el puerto A libre y del puerto B, utilizaremos los 7 pines más bajos (RB0 a RB6) para activar los segmentos del display, y RB7 para el punto. Bien, eso será para la configuración de los pines del Micro Controlador.

Esquema del circuito.



***CÓDIGO FUENTE***

```
LIST P=16F84
        #include <P16F84.INC>
        __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

reg1 equ 0x0D                       ; 3 registros para el retardo
reg2 equ 0x0E
reg3 equ 0x0F
Cont equ 0x10
            ORG 0x00
            GOTO INICIO
            ORG 0X05


INICIO BSF STATUS, RP0              ; configurando puertos
            CLRF TRISB^ 0x080       ; PORTB = SALIDA
            BCF STATUS, RP0

REINI       CLRF cont               ; pone el contador a 0
            MOVF Cont, W            ; pasa el contador a w (índice)
            CALL Tabla              ; llama a la Tabla
            MOVWF PORTB             ; pasa el dato obtenido a PORTB
            CALL Retardo
```

```
Disp_       MOVF Cont, W
            XORLW B'1000'          ; verifica si el contador llegó a 9
            BTFSC STATUS, Z        ; si no es así salta una línea
            GOTO REINI             ; si llegó a 9 lo atiende en REINI
            INCF Cont, F           ; incrementa el contador
            MOVF Cont, W           ; pasa el contador a w (índice)
            CALL tabla             ; llama a la tabla
            MOVWF PORTB            ; pasa el dato obtenido en la tabla a
PORTB
            CALL Retardo
            GOTO Disp_

Tabla       ADDWF PCL, F           ; se incrementa el contador de
programa

            RETLW B'10010010'      ; código para el 5
            RETLW B'10011001'      ; código para el 4
            RETLW B'10110000'      ; código para el 3
            RETLW B'10100100'      ; código para el 2
            RETLW B'11111001'      ; código para el 1
            RETLW B'11000000'      ; código para el 0
            RETLW B'11000001'      ; código para el U
            RETLW B'10000110'      ; código para el E
            RETLW B'00010010'      ; código para el S.
```

```
Retardo     Movlw 40                ; Aquí se cargan los registros
            Movwf reg1              ; reg1, reg2 y reg3

Tres        Movlw 30                ; con los valores 30, 20 y 35
            Movwf reg2

Dos         Movlw 45
            Movwf reg3

Uno         decfsz reg3, 1          ; Aquí se comienza a decrementar
            Goto Uno
            Decfsz reg2, 1
            Goto Dos
            Decfsz reg1, 1
            Goto Tres
            Retlw 00                ; regresare del retardo


            END
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
MICROPROGRAMACIÓN

# GUÍA DE PRÁCTICA Nº 6
# TRABAJANDO DIRECTAMENTE CON EL DISPLAY
# (CONTADOR 0 - F)

## *OBJETIVOS.*

- Utilización de las instrucciones para crear tablas y subrutinas **(CALL, RETURN, RETLW)**
- Utilización de un Display de 7 segmentos (Ánodo común) como dispositivo de visualización.
- Utilización del registro **PLC (contador del programa)** para controlar el flujo de ejecución del programa (puesto que asigna un número a cada línea de código).
- Crear el Codificador de Binario a Decimal **(BCD)** por medio de código y el encendido de los segmentos del display, se hará activándolos desde el micro controlador.
- Utilización de una tabla de instrucciones para generar una rutina de visualización **(RETLW).**

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.
1 Regleta protoboard.
1 Fuente de cinco voltios.
1 Micro controlador Pic 16F84.
1 Cristal de 4 Mhz
1 Display de 7 segmentos (Ánodo común).

## *GENERALIDADES.*

### *El Registro PCL.*

Veamos como trabaja el Micro Controlador cuando se encuentra ante una serie de instrucciones. Existe un registro, llamado PCL, ubicado en la posición 0x02 en el banco de memoria, tiene mucho que ver con el flujo del programa, puesto que le asigna un número a cada línea de código. Todo empieza con la primera instrucción, esta tiene una posición indicada con un número en el registro PCL, cuando accede a esa posición, se lee la instrucción, se decodifica, y luego se ejecuta, una vez echo esto, el reloj del Micro controlador incrementa al contador de programa (PCL) en un unidad, esto hace que el PCL apunte a la segunda instrucción, ahora se lee esta segunda instrucción, se decodifica y también se ejecuta. Nuevamente, el reloj del sistema incrementa el PCL para que apunte a la tercera instrucción, la decodifique y la ejecute. Este proceso se repite hasta que termina el programa (es decir, cuando encuentra un END).

## *Las Tablas:*

Me imagino que sabes lo que es una tabla, bueno, una tabla es algo como esto...

| Cont. de Programa | ISNT. | DATO |
|---|---|---|
| **PCL=11 »** | RETLW | 11000000 |
| **PCL=12 »** | RETLW | 11100001 |
| **PCL=13 »** | RETLW | 00001111 |
| **PCL=14 »** | RETLW | 00111001 |

Te preguntarás por el contenido de esta tabla, bueno, en esta tabla, cada línea horizontal, es una línea de código, y la dirección de cada línea, está dada por el valor del PCL (el contador de programa), suponte ahora el siguiente código...RETLW 00001111

RETLW, es retornar cargando W con el Literal 00001111, el problema es que para llegar a esta instrucción deberías pasar por encima de las dos líneas anteriores.

La primera instrucción ADDWF PCL, F indica que se le debe sumar al registro PCL, lo que hay en W. Con F, le indicamos que guarde el resultado en el mismo registro PCL, es decir...

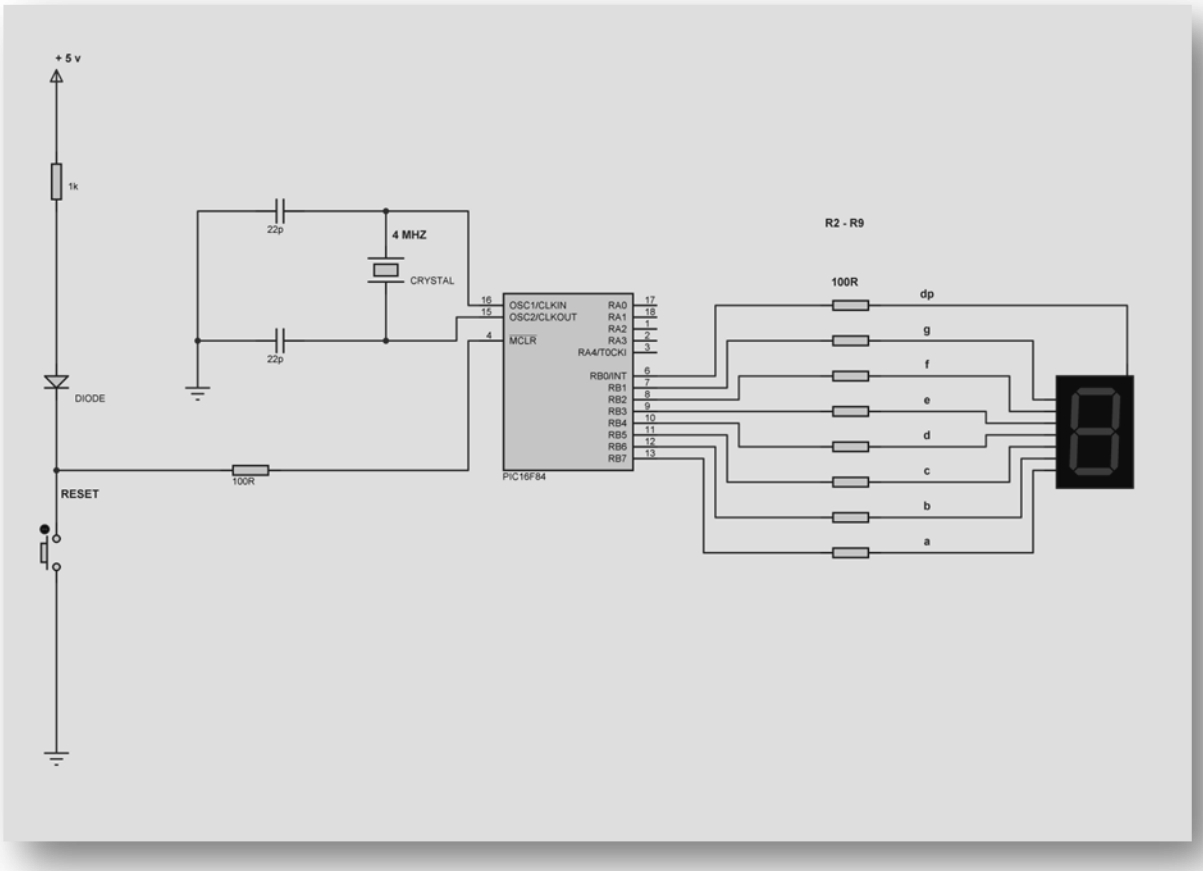
***PCL = PCL + W***

El acceso a la tabla lo haremos a través de W, le cargamos un valor y llamamos a la tabla, justo donde al PCL se le suma el valor de W.

## *PROCEDIMIENTO.*

Construya el siguiente esquema de conexión del circuito que cuenta desde 0 hasta F mostrando la cuenta en un display de 7 segmentos programa escrito en ensamblador. La finalidad del mismo consiste en contar en hexadecimal a través del puerto B (patitas RBO RB7), iluminando el encendido de los segmentos del display, se hará activándolos desde el micro, así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construcción del proyecto (.Hex) utilícelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.

Esquema del circuito.



## *CÓDIGO FUENTE.*

```
LIST P=16F84, R=DEC
          INCLUDE "p16F84.inc"
 __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

        UDATA
ValorActual RES 1
Aux1        RES 1
Aux2        RES 1
Aux3        RES 1


        ORG 0
        Goto INICIO
        ORG 5


INICIO
        Clrf ValorActual
        BANKSEL TRISB
        Clrf TRISB^ 0x080               ; Configurar pines de PORTB como salidas
        Clrf STATUS                     ; Apuntamos a la página 0


Bucle                                   ; Comienzo del bucle principal
        Movf ValorActual, W             ; Muevo ValorActual a W
        Call Tabla                      ; Llamamos a rutina de conversión
        Movwf PORTB                     ; ponemos resultado en PORTB (el display)
        Incf ValorActual, F             ; Incrementamos el valor para la siguiente
        Btfsc ValorActual, 4            ; iteración, y si se activa el bit 4, hemos
        Clrf ValorActual                ; llegado a 10H, y reseteamos la variable.
        Movlw 8                         ; Inicializamos Aux1 a 8 para conseguir
        Movwf Aux1                      ; una espera algo mayor que 1 segundo.
```

```
Espera
        Call Espera1                    ; Rutina con un retardo de 0,13 Seg. Aprox.
        Decfsz Aux1, F
        Goto Espera
        Goto Bucle                      ; Retornar al bucle principal

Espera1                                 ; Realiza 256 llamadas a la rutina Espera2
        Movlw 0xff
        Movwf Aux2

Bucle1
        Call Espera2
        Decfsz Aux2, F
        Goto Bucle1
        Return

Espera2
        Movlw 0xff
        Movwf Aux3

Bucle2
        Decfsz Aux3, F
        Goto Bucle2
        Return
```

```
Tabla
        ANDLW B'00001111'
        ADDWF PCL, F
        Retlw B'00000011'
        Retlw B'10011111'
        Retlw B'00100101'
        Retlw B'00001101'
        Retlw B'10011001'
        Retlw B'01001001'
        Retlw B'01000001'
        Retlw B'00011111'
        Retlw B'00000001'
        Retlw B'00011001'
        Retlw B'00010001'
        Retlw B'11000001'
        Retlw B'01100011'
        Retlw B'10000101'
        Retlw B'01100001'
        Retlw B'01110001'
        END
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
MICROPROGRAMACIÓN

# GUÍA DE PRÁCTICA Nº 7
# MANEJO DE INTERRUPCIONES
# (CONTADOR HEXADECIMAL)

## *OBJETIVOS.*

- Determinar la forma en que debe colocarse el código en el programa fuente para aceptar y atender una rutina de interrupción.
- Utilización del registro **INTCON (Registro Controlador Interrupciones)** para especificar la fuente de la interrupción producida.
- Que el alumno conozca que es una **ISR (Rutina de Servicio de Interrupción)** porción de código utilizada para atender a la interrupción producida.
- Utilización del registro **OPTION** (Registro que proporciona opciones para hacer mas eficiente tu aplicación y simplificar el código)
- Utilización de la instrucción **RETFIE** para retorno de una interrupción y activar a GIE

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.
1 Regleta protoboard.
1 Fuente de cinco voltios.
1 Micro controlador Pic 16F84.
1 Cristal de 4 Mhz
1 Display de 7 segmentos (Ánodo común).

## *GENERALIDADES.*

### *INTERRUPCIONES.*

Se trata de un acontecimiento que hace que el micro deje de lado lo que se encuentra realizando, atienda ese suceso y luego regrese y continúe con lo suyo. Pues eso son las interrupciones, pero veamos, hay dos tipos de interrupciones posibles, una es mediante una acción externa (es decir por la activación de uno de sus pines), la otra es interna (por ejemplo cuando ocurre el desbordamiento de uno de sus registros)

En el PIC 16F84 hay 4 fuentes de interrupciones, veamos cuales son:

- Por el pin RB0/INT, que regresa al PIC del modo SLEEP (interrupción externa).
- Por los pines RB4 a RB7, configurados como entrada y en caso de que alguno de ellos cambie de estado (interrupción externa).
- Por desbordamiento del registro TMR0, cuando este registro pasa de 255 a 0 en decimal ó 0xFF a 0x00 en hexadecimal (interrupción interna).
- Al completar la escritura de la EEPROM de datos (interrupción interna).

El tema es que, debe haber algo que nos indique la fuente de interrupción que se ha producido, y estas son las banderas de interrupciones, cada interrupción tiene su propia bandera y es un bit del registro **INTCON**, que cambia de estado de 0 a 1 cuando se produce la interrupción, salvo la última que se encuentra en el registro EECON1

| REGISTRO INTCON | | | | | | | |
|---|---|---|---|---|---|---|---|
| GIE | EEIE | TOIE | INTE | RBIE | TOIF | INTF | RBIF |

```
; -------  Habilitación de interrupciones  -------
BSF INTCON, GIE          ; habilitamos todas las interrupciones
BSF INTCON, INTE         ; que sean interrupciones externas
```

Si se ha producido una interrupción, obviamente una de las banderas del registro **INTCON** cambiará de estado y el micro irá a la dirección 0x04 como si se hubiera producido un CALL (una llamada) a esa dirección para ejecutar la ISR, por lo tanto la pila o STACK se carga una posición más, y el mecanismo de las interrupciones se deshabilita (es decir GIE=0) para dar lugar a la ISR.

Ahora bien, debes recuperar los registros importantes (lo que acabamos de ver), averiguar la fuente de interrupción, atender la interrupción, luego restaurar aquellos registros importantes, reponer el estado de las banderas del registro **INTCON** (aquellas que fueron modificadas por la interrupción) y regresar, pero no con un RETURN, ya que no estas regresando de una rutina cualquiera, sino de una interrupción, la cual está deshabilitada, y para habilitarla nuevamente es recomendable utilizar la instrucción **RETFIE**. Y ahora todas las interrupciones están habilitadas nuevamente, es decir GIE=1

Por cierto y antes de que lo olvide, si bien cada Flag cambia o se pone a 1 al producirse una interrupción, es tarea tuya borrarlo o ponerlo a cero nuevamente, ya que si no lo haces el micro estará siempre interrumpido o lo que es lo mismo, creerá que la interrupción se está produciendo continuamente.

Lo primero que debes saber, es que cuando una interrupción se produce, sea cual fuere la fuente de interrupción, el micro deja todo y salta a la dirección 0x04, éste es el vector de interrupción, si recuerdas de nuestro primer tutorial, siempre saltábamos por encima de esta dirección para iniciar nuestro programa, en esta dirección es donde escribiremos la rutina que dé servicio a todas las interrupciones, o bien haremos un salto a donde se encuentre ese trozo de código, el cual se conoce como ISR (Rutina de Servicio de Interrupción)

## *EL REGISTRO OPTION*

Este es otro de los registros que tienen mucho que ver con las interrupciones, algunos de sus Bits deben ser modificados, según la aplicación que estés realizando.

Por ejemplo; dijimos que por el **pin RB0/INT,** regresas al PIC del modo **SLEEP**, lo cual podría hacerse mediante un pulsador, suponte que el pulsador está al polo positivo (VCC) y con una resistencia a GND, de tal modo que la interrupción se produzca al enviar un 1 (presionando el pulsador), pero también podría hacerse enviando un 0 (liberando al pulsador). por lo tanto la interrupción debe ser sensible a un 1 o bien a un 0, como sabrá esto el micro?, pues muy fácil, hay que especificarlo, y esto se hace en el **Bit6 (INTDEG)** del registro **OPTION**, con un 1 será sensible al flanco ascendente, y en el momento que envíes un 1 por el pulsador se producirá la interrupción, si pones ese Bit a 0 será sensible al flanco descendente y la interrupción se producirá cuando liberes el pulsador, es decir enviando un 0.

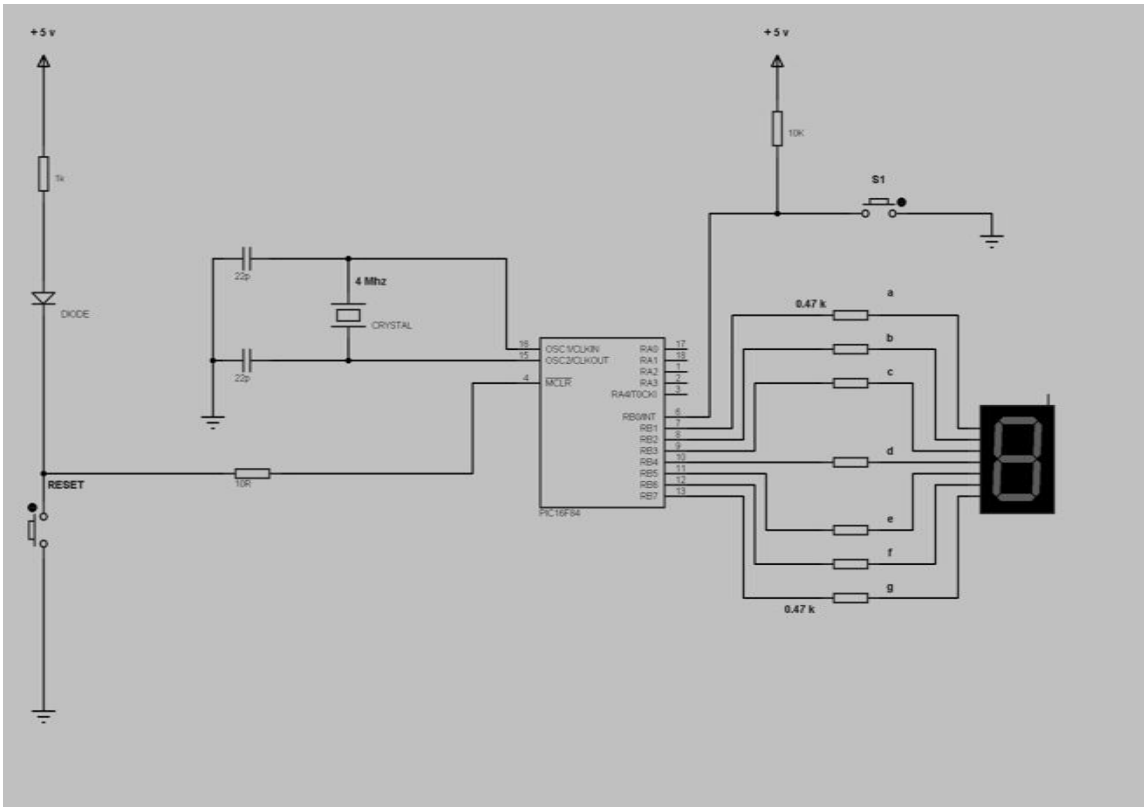
Este es el registro OPTION.

| REGISTRO OPTION | | | | | | | |
|---|---|---|---|---|---|---|---|
| RBPU | INTDEG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |

***PROCEDIMIENTO.***

Construya el siguiente esquema de conexión del circuito con el micro controlador utilizando la interrupción por el pin RB0, y en él estará conectado el pulsador, lo que vamos a hacer es un contador. La finalidad del mismo consiste en contar en hexadecimal a través del puerto B (patitas RB1- RB7), iluminando el encendido de los segmentos del display, se hará activándolos desde el micro controlador, así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construcción del proyecto (.Hex) utilícelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.

Esquema del circuito.



***CÓDIGO FUENTE.***

```
        LIST P=16F84, R=DEC
                INCLUDE "p16F84.inc"
         __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON



CONTA  equ     0x0C                                    ; Registro auxiliar para el
RETARDO
DIGITO equ     0x0D                                    ; lleva el conteo del digito a
visualizar
#define  BANK1        bsf STATUS, 5                   ; Macro para abreviar el
BANCO 1
#define  BANK0        bcf STATUS, 5                   ; Macro para abreviar el
BANCO 0
                      Org     0                       ; Posición 0 de la Memoria de
Programa
                      Goto    INICIO                  ; Va a la etiqueta INICIO
                      Org     4                       ; Vector de Interrupción
                              Goto    ISR                     ; el programa se
direcciona a Org 4
                      Org     5


;*********************************************************************
***
;                             INICIO DEL PROGRAMA
*
;               CONFIGURACIÓN DE PUERTOS E INICIO DE VARIABLES
*
;*********************************************************************
***
```

```
INICIO
                    BANK1                       ; Selección del Banco 1
                    Movlw  0x01                 ; w = 0x01
                    Movwf  TRISB^0x080          ; Bit 0 en "1" RB0 = IN; RB1-
RB7 = 0
                    Clrf   TRISA^0x080          ; trisa = 0, Todo el Puerto RA
= Salida
                    BANK0                       ; selección del Banco 0
                    Clrf   DIGITO               ; DIGITO = 0
                    Call   TABLA
                    Movwf  PORTB                ; Porta= w, Visualizar el digito
0



;*********************************************************************
***
;                   PROGRAMACIÓN DE LA INTERRUPCIÓN
*
;*********************************************************************
***
                    BANK1                       ; Selección del Banco 1
                    Movlw  0x80                 ; w = 0x80
                    Movwf  OPTION_REG^0x080     ; Option_Reg = w
                    BANK0                       ; Selección del Banco 0
                    Movlw  0x90                 ; w = 0x90
                    Movwf  INTCON               ; Intcon = w
WAIT                SLEEP                       ; Dormir (Esta instrucción
permite que
                                                ; El micro controlador entrar en
                                                 ; un estado Pasivo donde
                                            consume muy
                                                 ; Poca potencia).
                    Goto   WAIT                 ; Va a WAIT
```

```
;*****************************************************************************
;                          EN ESTA PARTE DEL PROGRAMA SE ATIENDE
*
; LA INTERRUPCIÓN QUE EN NUESTRO CASO ES POR FLANCO DE BAJADA EN EL PIN RB0/INT
*
;*****************************************************************************

ISR             Btfss   INTCON, 1       ; Fue Interrupción?
                Retfie                  ; no, retorne y habilite
interrupciones
                                        ; Globales (Gie = 1)
                Movlw   .15             ; si, carga a w = 15
                Xorwf   DIGITO, 0       ; exor entre w y DIGITO (Para
                                        ; determinar que el Conteo sólo
                                        ; va hasta F)
                Btfss   STATUS, 2       ; DIGITO es 15?
                Goto    UP              ; NO, va a UP
                Clrf    DIGITO          ; SI, DIGITO = 0
                Goto    VISUAL          ; va a VISUAL
UP              Incf    DIGITO, 1       ; Incrementa al registro
DIGITO
VISUAL          Call    TABLA           ; Llama la TABLA
                Movwf   PORTB           ; el contenido de w lo pasa al
Puerto B
                                        ; Para visualizar el digito
                                        ; Correspondiente


SOLTAR          Call    RETARDO         ; llama la rutina de retardo
para evitar
                                        ; rebotes
                Btfss   PORTB, 0        ; Soltaron el interruptor?
                Goto    SOLTAR          ; NO, va a SOLTAR
```

```
                        Bcf     INTCON, 1                   ; SI, borra la bandera de
Interrupción
                                                            ; Por cambio en el RB0/INT
                        Retfie                              ; Retorna y habilita las
interrupciones
                                                            ; Globales (GIE = 1)
;-------------------------------------------------------------------------------------------------
        ----
;               Si el usuario no desea utilizar la instrucción "RETFIE", también se puede
          ;             reemplazar por las dos siguientes líneas que equivalen a lo mismo:

;                       Bsf     Intcon, 7       ; Habilita Interrupciones globales (Gie =1)
;                       Return                  ; retorna del llamado
;-------------------------------------------------------------------------------------------------
---


;*****************************************************************************
***
;                               TABLA PARA DISPLAY DE ÁNODO COMÚN
*
;*****************************************************************************
***

TABLA   Movf    DIGITO, 0                               ; Mueve el contenido de
DIGITO a w
                Addwf   PCL, 1                          ; suma a pcl el contenido de w
                Retlw   0x80                            ; Cuando el digito es 0
                Retlw   0xF2                            ; Cuando el digito es 1
                Retlw   0x48                            ; Cuando el digito es 2
                Retlw   0x60                            ; Cuando el digito es 3
                Retlw   0x32                            ; Cuando el digito es 4
                Retlw   0x24                            ; Cuando el digito es 5
                Retlw   0x04                            ; Cuando el digito es 6
                Retlw   0xF0                            ; Cuando el digito es 7
                Retlw   0x00                            ; Cuando el digito es 8
```

```
                Retlw   0x20                    ; Cuando el digito es 9
                Retlw   0x10                    ; Cuando el digito es A
                Retlw   0x06                    ; Cuando el digito es B
                Retlw   0x8C                    ; Cuando el digito es C
                Retlw   0x42                    ; Cuando el digito es D
                Retlw   0x0C                    ; Cuando el digito es E
                Retlw   0x1C                    ; Cuando el digito es F
```

```
;*******************************************************************************
; ESTA RUTINA GENERA UN TIEMPO PRUDENTE PARA EVITAR QUE LOS REBOTES QUE GENERA
; EL SWITCH AL SER PRESIONADO CONFUNDAN AL MICRO CONTROLADOR.
; Total tiempo de Retardo 772 uS (Micro Segundos)
;*******************************************************************************

RETARDO         Clrf    CONTA                   ; CONTA = 0
INCREM          Incfsz  CONTA, 1                ; Incrementa a CONTA y salta si
                                                ; CONTA= 0? (255 -> 0)
                Goto    INCREM                  ; Va a la Etiqueta INCREM
                Return                          ; Retorna del llamado
                End                             ; Fin del Programa.
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
MICROPROGRAMACIÓN

# GUÍA DE PRÁCTICA Nº 8
# TRABAJANDO DIRECTAMENTE CON EL DISPLAY
# (CONTADOR 0 - 9)

## *OBJETIVOS.*

- Utilización de las instrucciones para crear tablas y subrutinas **(CALL, RETURN, RETLW)**
- Utilización de un Display de 7 segmentos (Ánodo común) como dispositivo de visualización.
- Utilización del registro **PLC (contador del programa)** para controlar el flujo de ejecución del programa (puesto que asigna un número a cada línea de código).
- Crear el Codificador de Binario a Decimal **(BCD)** por medio de código y el encendido de los segmentos del display, se hará activándolos desde el micro controlador.
- Utilización de una tabla de instrucciones para generar una rutina de visualización **(RETLW).**

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.
1 Regleta protoboard.
1 Fuente de cinco voltios.
1 Micro controlador Pic 16F84.
1 Cristal de 4 Mhz
1 Display de 7 segmentos (Ánodo común).

## *GENERALIDADES.*

### *El Registro PCL.*

Veamos como trabaja el Micro Controlador cuando se encuentra ante una serie de instrucciones. Existe un registro, llamado PCL, ubicado en la posición 0x02 en el banco de memoria, tiene mucho que ver con el flujo del programa, puesto que le asigna un número a cada línea de código. Todo empieza con la primera instrucción, esta tiene una posición indicada con un número en el registro PCL, cuando accede a esa posición, se lee la instrucción, se decodifica, y luego se ejecuta, una vez echo esto, el reloj del Micro controlador incrementa al contador de programa (PCL) en un unidad, esto hace que el PCL apunte a la segunda instrucción, ahora se lee esta segunda instrucción, se decodifica y también se ejecuta. Nuevamente, el reloj del sistema incrementa el PCL para que apunte a la tercera instrucción, la decodifique y la ejecute. Este proceso se repite hasta que termina el programa (es decir, cuando encuentra un END).

## *Las Tablas:*

Me imagino que sabes lo que es una tabla, bueno, una tabla es algo como esto...

| Cont. de Programa | ISNT. | DATO |
|---|---|---|
| **PCL=11 »** | RETLW | 11000000 |
| **PCL=12 »** | RETLW | 11100001 |
| **PCL=13 »** | RETLW | 00001111 |
| **PCL=14 »** | RETLW | 00111001 |

Te preguntarás por el contenido de esta tabla, bueno, en esta tabla, cada línea horizontal, es una línea de código, y la dirección de cada línea, está dada por el valor del PCL (el contador de programa), suponte ahora el siguiente código...RETLW 00001111

RETLW, es retornar cargando W con el Literal 00001111, el problema es que para llegar a esta instrucción deberías pasar por encima de las dos líneas anteriores.

La primera instrucción ADDWF PCL, F indica que se le debe sumar al registro PCL, lo que hay en W. Con F, le indicamos que guarde el resultado en el mismo registro PCL, es decir...

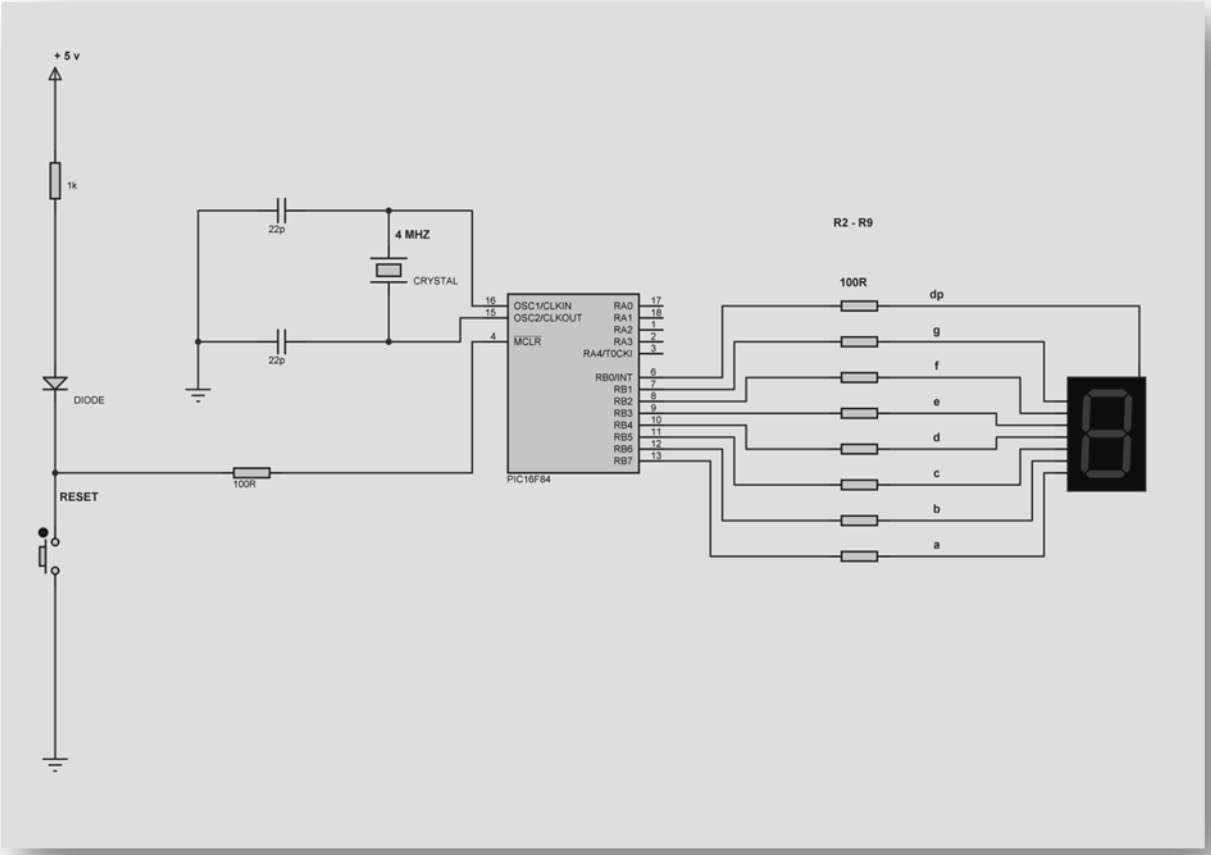
***PCL = PCL + W***

El acceso a la tabla lo haremos a través de W, le cargamos un valor y llamamos a la tabla, justo donde al PCL se le suma el valor de W.

## *PROCEDIMIENTO.*

Construya el siguiente esquema de conexión del circuito que cuenta desde 0 hasta 9 mostrando la cuenta en un display de 7 segmentos programa escrito en ensamblador. La finalidad del mismo consiste en contar en hexadecimal a través del puerto B (patitas RBO RB7), iluminando el encendido de los segmentos del display, se hará activándolos desde el micro, así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construcción del proyecto (.Hex) utilícelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.

Esquema del circuito.



## *CÓDIGO FUENTE.*

```
LIST P=16F84, R=DEC
          INCLUDE "p16F84.inc"
 __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON


        UDATA
ValorActual RES 1
Aux1        RES 1
Aux2        RES 1
Aux3        RES 1


        ORG 0
        Goto INICIO
        ORG 5


INICIO
        Clrf ValorActual
        BANKSEL TRISB
        Clrf TRISB^ 0x080               ; Configurar pines de PORTB como salidas
        Clrf STATUS                     ; Apuntamos a la página 0


Bucle                                   ; Comienzo del bucle principal
        Movf ValorActual, W             ; Muevo ValorActual a W
        Call Tabla                      ; Llamamos a rutina de conversión
        Movwf PORTB                     ; ponemos resultado en PORTB (el display)
        Incf ValorActual, F             ; Incrementamos el valor para la siguiente
        Btfsc ValorActual, 4            ; iteración, y si se activa el bit 4, hemos
        Clrf ValorActual                ; llegado a 10H, y reseteamos la variable.
        Movlw 8                         ; Inicializamos Aux1 a 8 para conseguir
        Movwf Aux1                      ; una espera algo mayor que 1 segundo.
```

```
Espera
        Call Espera1                    ; Rutina con un retardo de 0,13 Seg. Aprox.
        Decfsz Aux1, F
        Goto Espera
        Goto Bucle                      ; Retornar al bucle principal

Espera1                                 ; Realiza 256 llamadas a la rutina Espera2
        Movlw 0xff
        Movwf Aux2

Bucle1
        Call Espera2
        Decfsz Aux2, F
        Goto Bucle1
        Return

Espera2
        Movlw 0xff
        Movwf Aux3

Bucle2
        Decfsz Aux3, F
        Goto Bucle2
        Return
```

```
Tabla
        ANDLW B'00001111'
        ADDWF PCL, F
        Retlw B'00000011'
        Retlw B'10011111'
        Retlw B'00100101'
        Retlw B'00001101'
        Retlw B'10011001'
        Retlw B'01001001'
        Retlw B'01000001'
        Retlw B'00011111'
        Retlw B'00000001'
        Retlw B'00011001'
        Retlw B'00010001'
        Retlw B'11000001'
        Retlw B'01100011'
        Retlw B'10000101'
        Retlw B'01100001'
        Retlw B'01110001'
        END
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR.

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE.

INGENIERÍA EN SISTEMAS INFORMÁTICOS.

DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA.

MICROPROGRAMACIÓN.

# GUÍA DE PRÁCTICA Nº 9
# ENCENDER OCHO LEDS EN FORMA SECUENCIAL

## *OBJETIVOS.*

- Utilización de un Diodo Led como dispositivo de visualización de salida.
- Utilización de las instrucciones de manipulación y desplazamiento de bits de un registro.
- Utilización de las instrucciones de manipulación y control de secuencia del flujo del programa (salto a una Subrutina).
- Verificar el modo en el que se debe programar el sentido de los puertos.

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.

1 Vom (voltímetro, óhmetro y amperímetro).

1 Regleta protoboard.

1 Fuente de cinco voltios.

1 Micro controlador Pic 16F84.

1 Cristal de 4 Mhz

Resistencias (1KΩ - 5kΩ - 100Ω – 220RΩ).

8 Diodos Leds.

## *GENERALIDADES.*

### *IMPORTANCIA DE LAS RUTINAS*

La mayoría de los micro controladores incluyen en su repertorio de instrucciones algunas que permiten saltar a una rutina y, cuando se complementa su ejecución, retornar al programa principal

El empleo de subrutinas aporta muchas ventajas entre las que se destacan las siguientes:

5. Se pueden escribir como subrutinas secciones de código y ser empleadas en muchos programas (por ejemplo, la subrutina de exploración de un teclado).
6. Dan a los programas un carácter modular, es decir, se pueden codificar diferentes módulos para usarlos en cualquier programa.
7. Se reduce notablemente el tiempo de programación, la detección de errores, usando repetidamente una subrutina.
8. El código es más fácil de interpretar, dado que las instrucciones de las subrutinas no aparecen en el programa principal. Solo figuran las llamadas CALLs.

### *LAS INSTRUCCIONES CALL Y RETURN*

La instrucción CALL (llamada la subrutina) consigue que la ejecución del programa continúe en la dirección donde se encuentra la subrutina a la que hace referencia. Es similar a GOTO pero coloca en la pila la dirección de la siguiente instrucción que se debe ejecutar después de la CALL.

La subrutina finaliza con la instrucción RETURN (Retorno de la subrutina) que retoma la dirección guardada en le pila y la coloca en el contador del programa PC continuando el flujo de control con la instrucción que sigue a la CALL.

### *INSTRUCCIONES DE ROTACIÓN DE BITS*

**RLF f, d; rotación a la izquierda, destino**

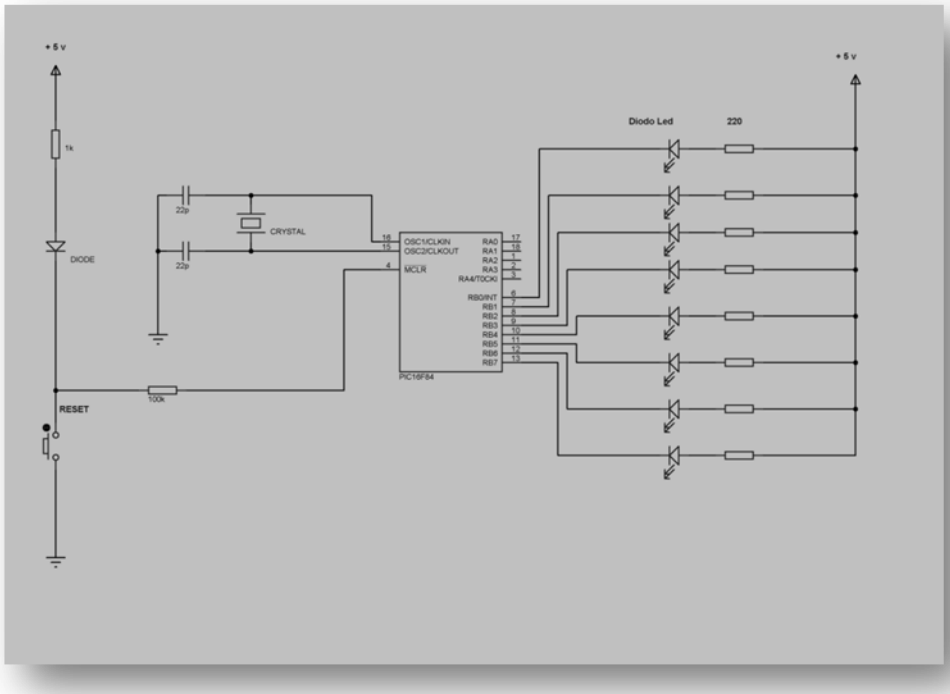
**RRF f, d; rotación a la derecha, destino**

En estas operaciones (Rotate Left File y Rotate Right File) los bits son desplazados de cada posición a la siguiente, en sentido derecho o izquierdo. El desplazamiento es cerrado, formando un anillo.

***PROCEDIMIENTO.***

Construya el siguiente esquema de conexión del circuito que encienda 8 diodos Led's en forma secuencial, y para ello recurriremos a una llamada de una rutina de retardo, así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construccion del proyecto (.Hex) utilizelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.

Esquema del circuito.




En el circuito podemos ver que hemos añadido 8 diodos leds (RB0-RB7) conectado al puerto B además hemos añadido a cada diodo led su correspondiente resistencia limitadora.

## *CÓDIGO FUENTE*

```
LIST p=16f84
                INCLUDE "p16f84.inc"

J               equ             h'1F'
K               equ             h'1E'

                __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON
                Org     5

                Bsf             STATUS, RP0          ;  Ir al banco 1
                Movlw           b'00000000'
                Movwf           TRISB ^0x080         ;  Portb como salida
                Bcf             STATUS, RP0          ;  Regreso al Banco 0

                Movlw           b'11111111'
                Movwf           PORTB
                Goto            start
; Retardo
Delay           Movlw           d'255'
                Movwf           j
Jloop           Movwf           k
Kloop           Decfsz          k, f
                Goto            Kloop
                Decfsz          j, f
                Goto            Jloop

Start
                Bcf             PORTB, 0            ; Encender el LED en PORTB.0
                Call            delay
                Bsf             PORTB, 0            ; Apagar el LED en PORTB.0
                Call            delay
                Bcf             PORTB, 1            ; Encender el LED en PORTB.1
                Call            delay
                Bsf             PORTB, 1            ;  Apagar el LED en PORTB.1
                Call            delay
                Bcf             PORTB, 2            ;  Encender el LED en PORTB.2
                Call            delay
                Bsf             PORTB, 2            ;  Apagar el LED en PORTB.2
                Call            delay
                Bcf             PORTB, 3            ;  Encender el LED en PORTB.3
                Call            delay
                Bsf             PORTB, 3            ;  Apagar el LED en PORTB.3
                Call            delay
                Bcf             PORTB, 4            ;  Encender el LED en PORTB.4
                Call            delay
```

```
Bsf     PORTB, 4        ; Apagar el LED en PORTB.4
Call    delay
Bcf     PORTB, 5        ; Encender el LED en PORTB.5
Call    delay
Bsf     PORTB, 5        ; Apagar el LED en PORTB.5
Call    delay
Bcf     PORTB, 6        ; Encender el LED en PORTB.6
Call    delay
Bsf     PORTB, 6        ; Apagar el LED en PORTB.6
Call    delay
Bcf     PORTB, 7        ; Encender el LED en PORTB.7
Call    delay
Bsf     PORTB, 7        ; Apagar el LED en PORTB.7
Goto    Start
End
```

CONCLUSIONES:_____

RECOMENDACIONES:_____


UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
MICROPROGRAMACIÓN

# GUÍA DE PRÁCTICA Nº 10
# UN SEMÁFORO CON EL PIC16F84

## *OBJETIVOS.*

- Utilización de un Diodo Led como dispositivo de visualización (salida).
- Utilización de las instrucciones de manipulación y desplazamiento de bits de un registro (**BSF, BCF**).
- Utilización de las instrucciones de manipulación y control de secuencia del flujo del programa (salto a una Subrutina **CALL**, **INCFSZ f, d DECFSZ f, d)**.
- Verificar el modo en el que se debe programar el sentido de los puertos.

## *METODOLOGÍA*

Cada estudiante, previo estudio de la guía propuesta, procederá a desarrollarla de manera grupal, durante el tiempo programado y asignado para las prácticas correspondientes dentro del laboratorio, asistido por al menos un instructor responsable de dicha práctica.

## *MATERIAL Y EQUIPO.*

1 Computadora de escritorio con MPLAB y WINPIC previamente instalado y funcionando correctamente.

1 Vom (voltímetro, óhmetro y amperímetro).

1 Regleta protoboard.

1 Fuente de cinco voltios.

1 Micro controlador Pic 16F84.

1 Cristal de 4 Mhz

Resistencias (1KΩ - 5kΩ - 100Ω – 220RΩ).

6 Diodos Leds.

## *GENERALIDADES.*

### *IMPORTANCIA DE LAS RUTINAS*

La mayoría de los micro controladores incluyen en su repertorio de instrucciones algunas que permiten saltar a una rutina y, cuando se complementa su ejecución, retornar al programa principal

El empleo de subrutinas aporta muchas ventajas entre las que se destacan las siguientes:

9. Se pueden escribir como subrutinas secciones de código y ser empleadas en muchos programas (por ejemplo, la subrutina de exploración de un teclado).

10. Dan a los programas un carácter modular, es decir, se pueden codificar diferentes módulos para usarlos en cualquier programa.

11. Se reduce notablemente el tiempo de programación, la detección de errores, usando repetidamente una subrutina.

12. El código es más fácil de interpretar, dado que las instrucciones de las subrutinas no aparecen en el programa principal. Solo figuran las llamadas CALLs.

### *LAS INSTRUCCIONES CALL Y RETURN*

La instrucción CALL (llamada la subrutina) consigue que la ejecución del programa continúe en la dirección donde se encuentra la subrutina a la que hace referencia. Es similar a GOTO pero coloca en la pila la dirección de la siguiente instrucción que se debe ejecutar después de la CALL.

La subrutina finaliza con la instrucción RETURN (Retorno de la subrutina) que retoma la dirección guardada en le pila y la coloca en el contador del programa PC continuando el flujo de control con la instrucción que sigue a la CALL.

***INSTRUCCIONES DE ROTACIÓN DE BITS***

**RLF f, d; rotación a la izquierda, destino**

**RRF f, d; rotación a la derecha, destino**

En estas operaciones (Rotate Left File y Rotate Right File) los bits son desplazados de cada posición a la siguiente, en sentido derecho o izquierdo. El desplazamiento es cerrado, formando un anillo.

**INCFSZ f, d increment file skip if 0**.

Incrementa el contenido del registro, si el resultado NO es 0 la siguiente instrucción se ejecuta normalmente, pero si es 0 la instrucción siguiente se ignora y se salta.

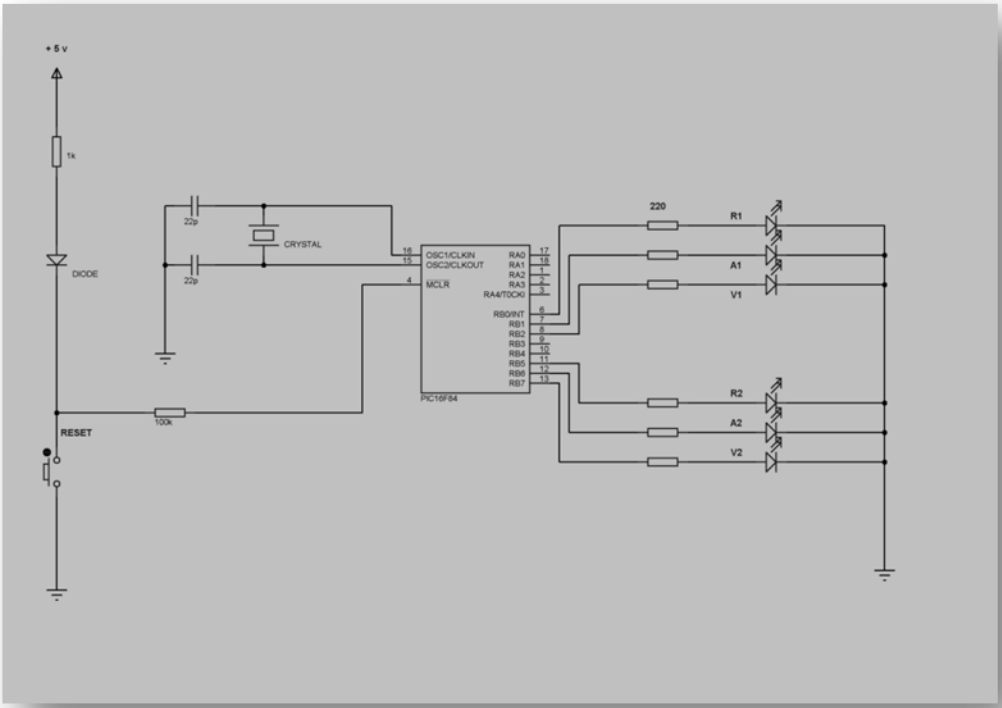
**DECFSZ f, d Increment file skip if 0**.

Incrementa el contenido del registro, si el resultado NO es 0 la siguiente instrucción se ejecuta normalmente, pero si es 0 la instrucción siguiente se ignora y se salta.

***PROCEDIMIENTO.***

Construya el siguiente esquema de conexión del circuito que simule el funcionamiento de un semáforo, así se ve en el siguiente esquema y lleve acabo el proceso de construir el proyecto (con el código fuente que se le proporciona) y simularlo en MPLAB, ahora con el archivo resultante de la construccion del proyecto (.Hex) utilizelo para programar el micro controlador con WINPIC para luego proceda a realizar pruebas del funcionamiento del circuito.

Esquema del circuito.



## *CÓDIGO FUENTE*

```
LIST p=16f84
                INCLUDE "p16f84.inc"

J               equ             h'1F'
K               equ             h'1E'

                __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

                Org     5

                Bsf             STATUS, RP0         ;  Ir al banco 1
                Movlw           b'00000000'
                Movwf           TRISB ^0x080        ;  Puerto B como salida
                Bcf             STATUS, RP0         ;  Ir al banco 2
                Movlw           b'00000000'
                Movwf           PORTB

Start
                Bsf             PORTB, 2
                Bsf             PORTB, 5
                Call            delay
                Call            delay
                Call            delay
                Bcf             PORTB, 2
                Bsf             PORTB, 1
                Call            delay
                Bcf             PORTB, 1
                Bsf             PORTB, 0
                Call            delay
                Bcf             PORTB, 5
                Bsf             PORTB, 7
                Call            delay
                Call            delay
                Call            delay
                Bcf             PORTB, 7
                Bsf             PORTB, 6
                Call            delay
                Bcf             PORTB, 6
                Bsf             PORTB, 5
                Call            delay
                Bcf             PORTB, 0
                Bcf             PORTB, 5
                Goto            start
```

```
; Retardo
Delay       Movlw       d'255'
            Movwf       j
Jloop       Movwf       k
Kloop       Decfsz      k, f
            Goto        kloop
            Decfsz      j, f
            Goto        jloop

            Movlw       d'255'
            Movwf       j
Jloop1      movwf       k
Kloop1      decfsz      k, f
            Goto        kloop1
            Decfsz      j, f
            Goto        jloop1

            Movlw       d'255'
            Movwf       j
Jloop2      movwf       k
Kloop2      decfsz      k, f
            Goto        kloop2
            Decfsz      j, f
            Goto        jloop2

            Movlw       d'255'
            Movwf       j
Jloop3      movwf       k
Kloop3      decfsz      k, f
            Goto        kloop3
            Decfsz      j, f
            Goto        jloop3

            Movlw       d'255'
            Movwf       j
Jloop4      Movwf       k
Kloop4      Decfsz      k, f
            Goto        kloop4
            Decfsz      j, f
            Goto        jloop4
```

```
            Movlw       d'255'
            Movwf       j
Jloop5      Movwf       k
Kloop5      Decfsz      k,f
            Goto        kloop5
            Decfsz      j, f
            Goto        jloop5

            Movlw       d'255'
            Movwf       j
Jloop6      Movwf       k
Kloop6      Decfsz      k, f
            Goto        kloop6
            Decfsz      j, f
            Goto        jloop6

            Movlw       d'255'
            Movwf       j
Jloop7      Movwf       k
Kloop7      Decfsz      k, f
            Goto        kloop7
            Decfsz      j, f
            Goto        jloop7
            End
```

CONCLUSIONES:_____

RECOMENDACIONES:_____

**ANEXO 4:** VISITAS TÉCNICAS REALIZADAS A LAS INSTITUCIONES EDUCATIVAS

**INSTITUCIÓN:** UNIVERSIDAD DE EL SALVADOR, FACULTAD DE INGENIERÍA Y ARQUITECTURA.

**UBICACIÓN:** SAN SALVADOR, SAN SALVADOR.

**CARRERA:** INGENIERÍA ELÉCTRICA.

Gracias a la entrevista realizada al Ing. Wilber Calderón, docente de la Universidad de El Salvador, Facultad de Ingeniería y Arquitectura, Escuela de Ingeniería Eléctrica, se obtuvo una idea respecto a las decisiones que debíamos tomar acerca de componentes, software, reglamentos y procedimientos a utilizarse en las practicas de laboratorio para PICs de la Facultad Multidisciplinaria de Occidente. A continuación se detallan los resultados obtenidos de esta entrevista.

- Las practicas con Micro controladores de la Facultad de Ingeniería y Arquitectura, pertenecen solamente a la carrera de Ingeniería Eléctrica.
- La duración de la práctica en el laboratorio es de una hora clase, este lapso de tiempo no es una regla, a veces la práctica puede extenderse, la mayor parte de las veces la práctica termina hasta que el alumno la finaliza.
- El modelo de Micro controlador utilizado es el PIC16F877 (40 pines), programador PIPO2 (variación del JDM), entrenador diseñado en la Escuela de Ingeniería y los circuitos son montados en breadboards.

- Es responsabilidad de los alumnos adquirir sus propios elementos incluyendo el Micro controlador y la breadboard.
- Por lo anteriormente mencionado, cada alumno es el encargado y responsable del cuidado del Micro controlador.
- Se forman grupos de dos alumnos cada uno, para realizar las prácticas de laboratorio.
- Al momento de trabajar con los Micro controladores, no se presentan muchos problemas, la única recomendación cuando se fabrican circuitos en placas de cobre, es la de no soldar el Micro controlador directamente en la placa, esto puede dañar el Micro controlador o borrar la información almacenada en él.
- Los alumnos tienen la libertad de elegir un proyecto, el cual debe ser presentado al finalizar el ciclo.
- No existe un software específico a utilizar en las prácticas de laboratorio, cada alumno es encargado de introducir el programa en el Micro controlador con el software de su preferencia.
- También es decisión del alumno el sistema operativo a utilizar, no se hace uso de simuladores dentro en las prácticas.
- Los Micro controladores son adquiridos en el extranjero, ya que conseguirlos en el país es una de las mayores dificultades encontradas.

**INSTITUCIÓN:** INSTITUTO TECNOLÓGICO CENTROAMERICANO.

**UBICACIÓN:** SANTA TECLA, SAN SALVADOR.

**CARRERA:** TÉCNICO EN INGENIERÍA ELÉCTRICA, TÉCNICO EN INGENIERÍA DE SISTEMAS.

En el Instituto Tecnológico Centroamericano (ITCA), el Ingeniero Aguiluz nos proporciono parte de su tiempo en la visita realizada en esta institución. En esta entrevista no se pudo lograr mucho, ya que la institución no posee hasta el momento con un laboratorio de Microcontroladores.

Para el próximo año esperan contar con un laboratorio con Microcontroladores, un proyecto que a este momento ya se encuentra aprobado. Al momento de la visita el Instituto no tenía una idea clara acerca de los modelos de componentes que se pretenden adquirir para instalar el laboratorio.

Actualmente el ITCA posee un laboratorio completo, en el cual existen varios entrenadores, unos son para memorias (EPROM, RAM) y otros para el microprocesador 8085. Algunos de los entrenadores han sido fabricados por ex-alumnos del Instituto. Los principales proveedores de componentes electrónicos para el ITCA son Josnab y Casa Rivas, los dos ofrecen descuentos a estudiantes.

# Manuales

*EL LENGUAJE ENSAMBLADOR Y EL PIC16F84.*

**INSTRUCCIONES ORIENTADAS A REGISTROS.**

- ADDWF f, d Suma W y el registro f, el resultado lo guarda según d (si d=0 se guarda en W y si d=1 se guarda en f).
- ANDWF f, d Realiza la operación AND lógica entre W y f, el resultado lo guarda según d.
- CLRF f Borra el registro f (pone todos sus bits a cero).
- CLRW -Borra el acumulador. ? COMF f, d Calcula el complementario del registro f (los bits que están a "0" los pone a "1" y viceversa. Resultado según d? DECF f, d Decrementa f en uno (le resta uno). Resultado según d.
- DECFSZ f, d Decrementa f y se salta la siguiente instrucción si el resultado es cero. Resultado según d? INCF f, d Incrementa f en uno (le suma uno). Resultado según d.
- INCFSZ f, d Incrementa f y se salta la siguiente instrucción si el resultado es cero (cuando se desborda un registro vuelve al valor 00h). Resultado según d.
- IORWF f, d Realiza la operación lógica OR entre W y f. Resultado según d.
- MOVF f,d Mueve el contenido del registro f a W si d=0 (si d=1 lo vuelve a poner en el mismo registro)
- MOVWF f mueve el valor de W a f. Por ejemplo, si queremos copiar el valor del registro "REG1" al registro "REG2" (ya veremos como ponerles nombres a los registros) escribiremos:

        - MOVF REG1,0 ;mueve el valor de REG1 a W
        - MOVWF REG2 ; mueve el valor de W a REG2
    - Lo que va después del; son comentarios

- NOP -No hace nada, solo pierde el tiempo durante un ciclo.
- RLF f, d Rota el registro f hacia la izquierda a través del bit CARRY (todos los bits se mueven un lugar hacia la izquierda, el bit 7 de f pasa al CARRY y el bit CARRY pasa al bit 0 de f). Resultado según d.
- RRF f, d Lo mismo que RLF pero hacia la derecha.
- SUBWF f, d Resta f y W (f - W). Resultado según d.
- SWAPF f, d intercambia los 4 primeros bit de f por los otros cuatro. Resultado según d.
- XORWF f, d Realiza la operación lógica XOR (OR exclusiva) entre W y f. Resultado según d.

**INSTRUCCIONES ORIENTADAS A BITS.**

- BCF f,b Pone a "0" el bit b del registro f
- BSF f,d Pone a "1" el bit b del registro f
- BTFSC f,b Se salta la siguiente instrucción si el bit b del registro f es "0"
- BTFSS f,b Se salta la siguiente instrucción si el bit b del registro f es "1"

**INSTRUCCIONES ORIENTADAS A CONSTANTES Y DE CONTROL.**

- ADDLW k Le suma el valor k al acumulador (W).
- ANDLW k Operación lógica AND entre W y el valor k (resultado en W).
- CALL k Llamada a subrutina cuyo inicio esta en la dirección k
- CLRWDT -Borra el registro Watchdog
- GOTO k Salta a la dirección k de programa.
- IORLW k Operación lógica OR entre W y el valor k (resultado en W)
- MOVLW k carga el acumulador con el valor k. Por ejemplo, si queremos cargar el valor 2Ah en el registro "REG1" escribiremos:
    - MOVLW 2AH ; carga el acumulador con el valor 2Ah
    - MOVWF REG1 ; mueve el valor de W a "REG1"
- RETFIE Instrucción para volver de la interrupción
- RETLW k carga el valor k en W y vuelve de la interrupción
- RETURN vuelve de una subrutina.
- SLEEP El PIC pasa a modo de Standby.

**INSTRUCCIONES PARA EL ENSAMBLADOR.**

Existen una serie de instrucciones que son para el ensamblador y nos hacen la tarea de programación más sencilla y más legible.

- EQU: Un ejemplo de esto son las etiquetas, podemos poner un nombre a un registro de memoria, esto se hace mediante la instrucción EQU. Por ejemplo:

```
VARIABLE1       EQU       0CH
```

A partir de ahora en lugar de escribir 0CH podemos escribir VARIABLE1. Con EQU también podemos poner nombre a constantes de la misma forma.

- #DEFINE: Otra instrucción para el ensamblador que usaremos será la instrucción #DEFINE. Es parecido a EQU, solo que aquí no ponemos etiquetas a un registro, podemos ponerla a una instrucción entera, Por ejemplo:

```
#DEFINE    BANCO1    BSF   STATUS, 5

#DEFINE    BANCO0    BCF   STATUS, 5
```

A partir de ahora, cuando escribamos BANCO1 se pondrá a "1" el bit de selección de banco y pasaremos al banco 1, al escribir BANCO0 pasaremos al banco 0

- ORG: Indica al ensamblador la dirección (de memoria de programa) donde se guardará la instrucción que vaya a continuación. Por ejemplo:

```
        ORG       00H

        CLRF      VARIABLE1
```

La instrucción CLRF está en la dirección de memoria de programa 00H (será la primera instrucción en ser ejecutada por el PIC)

- END: Se escribe al final del programa para indicar que ya ha acabado. (Es obligatorio, si no da error).

Etiquetas a direcciones de Programa: muy útiles para usar con instrucciones CALL (Llamada a subrutina) o GOTO (Salto). Por ejemplo:

```
.........
[Hay programa anterior]
.........
        BTFSC       VARIABLE1, 0    ; Si el bit 0 de VARIABLE1 es
                                    ;"0" se salta la siguiente
                                    ; instrucción
        GOTO            ESUNO       ; Salta a ESUNO solo si el bit 0
                                    ; De VARIABLE1 es "1"
        BSF         VARIABLE1, 0    ; Si el bit 0 de VARIABLE1 es 0
                                    ; Se ejecuta esta instrucción y el
                                    ; Programa sigue por aquí
.............
[Continúa el programa]
.............
ESUNO                               ; Etiqueta a una dirección de
                                     ; programa
        BCF         VARIABLE1, 0    ; Si el bit 0 de VARIABLE1 es
                                    ;"1" se ejecuta esta otra
                                     ; Instrucción y el programa
                                     ; sigue por aquí
......
[Continúa el programa]
```

**UN POCO DE ORDEN**

Es importante llevar un poco de orden a la hora de escribir el programa, nos ayudará mucho:

Al principio van los EQU y los #DEFINE, después comenzamos con el programa.

El programa se escribe en cuatro columnas separadas por tabuladores:

En la primera columna se ponen las etiquetas a direcciones de programa.

En la segunda columna se ponen las instrucciones (BSF, CLRF, BTFSC... etc.).

En la tercera columna se ponen Los registros o parámetros a los que afecta la instrucción.

En la cuarta columna se ponen los comentarios que creas pertinentes (cuantos más mejor) seguidos de un punto y coma.

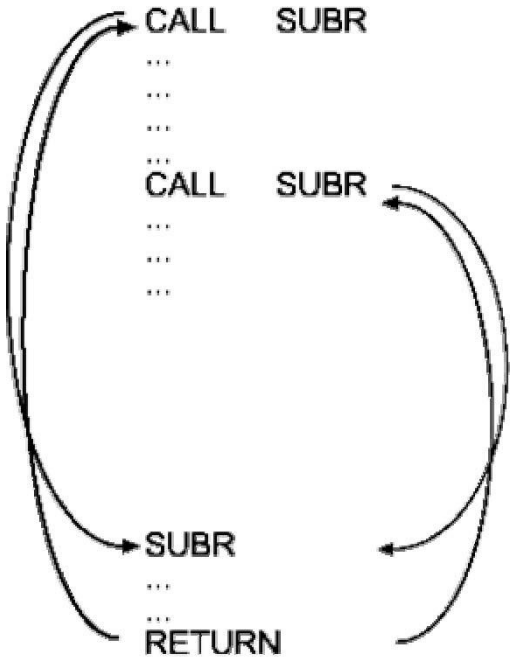
Ejemplo de programa bien ordenado:

```
;**********************************
;* PROGRAMA BIEN ORDENADO         *
;**********************************
;* El siguiente programa configura *
;* RA1 como entrada y RA0 como     *
;* Salida y hace que la salida (RA0) *
;* Sea la inversa de la entrada    *
;* (RA1)                           *
;**********************************

;(Conviene poner título y una
; Pequeña explicación de lo que
; hace el programa)
```

```
;(Primero los EQU y los #DEFINE)

STATUS      EQU         03H
TRISA       EQU         05H
PORTA       EQU         05H

#DEFINE     BANCO0      BCF     STATUS, 5
#DEFINE     BANCO1      BSF     STATUS, 5

;(Después empezamos con el programa)
ORG         00H                 ; Empezamos siempre a escribir en esta dirección
            BANCO1              ; Pasamos al banco 1 para hacer algunas
                                ; Configuraciones
            BCF     TRISA, 0    ; Configuramos RA0 como salida
            BSF     TRISA, 1    ; Configuramos RA1 como entrada
            BANCO   0           ; Volvemos al banco 0

INICIO      BTFSC   PORTA, 1    ; Comprueba la entrada (RA1), si es "0" se
                                ; salta la siguiente instrucción
            GOTO    ESUNO       ; si la entrada (RA1) es "1" va a ESUNO
            BSF     PORTA, 0    ; Pone a "1" la salida RA0. Ejecuta esta
                                ; instrucción porque la entrada RA1 era "0"
            GOTO    INICIO      ; Vuelve otra vez a comprobar el estado de
                                ; la entrada RA1
ESUNO       BCF     PORTA, 0    ; Pone a "0" la salida RA0. Ejecuta esta
                                ; instrucción porque la entrada RA1 era "1"
            GOTO    INICIO      ; Vuelve otra vez a comprobar el estado de
                                ; la entrada RA1
            END                 ; Indica final del programa
```

**SUBRUTINAS.**

Una subrutina o subprograma es una parte de programa que hace algo concreto y se repite a menudo, para ahorrar memoria y esfuerzo y para hacer el programa más comprensible se agrupa en forma de subrutina. Una subrutina se debe ejecutar siempre llamándola con la instrucción CALL y al final de dicha subrutina debe haber siempre un RETURN. El esquema de la derecha muestra cómo funcionan las subrutinas:




Durante el programa principal se llama varias veces a la subrutina SUBR (el nombre es lo de menos) con la instrucción CALL. Cuando el PIC ejecuta una instrucción CALL se guarda en memoria la dirección de código de programa a la que tiene que retornar de tal forma que cuando se encuentra con la instrucción RETURN vuelve al programa principal donde lo dejó.

Una subrutina no solo puede ser llamada desde el programa principal, también puede hacerse desde otra subrutina (una subrutina que llama a otra subrutina) o desde una interrupción (enseguida las veremos).

El siguiente ejemplo muestra un programa que utiliza una subrutina de retardo a la que llama DELAY. Esta subrutina de retardo se hace decrementando el registro CUENTA2 desde FFh hasta 00h 16 veces (las veces que se decrementa CUENTA2 son contadas hacia atrás por CUENTA1) De esta forma se consigue perder tiempo (el tiempo perdido con esta subrutina depende de la frecuencia a la que opere el PIC).

```
;****************************************
;*        USO DE SUBRUTINAS             *
;****************************************
;* Este programa configura RB0 como salida      *
;* Y genera una intermitencia en dicha          *
;* Salida                        *
;****************************************

STATUS      EQU         03H
TRISB       EQU         06H
PORTB       EQU         06H

CUENTA1     EQU         0CH         ; Las variables que usemos siempre a
                                    ; partir de la dirección 0Ch
CUENTA2     EQU         0DH

F           EQU         1
W           EQU         0
            ORG         00H
            BSF         STATUS, 5   ; banco 1
            BCF         TRISB, 0    ; RB0 como salida
            BCF         STATUS,5    ;banco 0

INICIO      BSF         TRISB,0     ;Pone a "1" RB0 (enciende)
            CALL        DELAY       ; Llama a la subrutina de retardo
            BCF         TRISB,0     ;Cuando vuelve del retardo pone
                                    ; A "0" RB0 (apaga)
            CALL        DELAY       ; llama a la subrutina de retardo
            GOTO        INICIO      ; cuando vuelve del retardo
                                    ; ejecuta el GOTO

;=================================================
;= DELAY:    Subrutina de retardo
;=          Modifica los siguientes registros:

;=                  CUENTA1
;=                  CUENTA2
;=                  ACUMULADOR
;=                  STATUS
```

```
DELAY   MOVLW   010H        ; Carga el acumulador con el valor
                            ; 10H (16 en decimal)
        MOVWF   CUENTA1     ; Mueve el contenido del acumulador
                            ; A CUENTA1
ACA1    MOVLW   0FFH        ; Carga el acumulador con el valor FFH
        MOVWF   CUENTA2     ; Mueve el contenido del acumulador
                            ; A CUENTA2
ACA     DECFSZ  CUENTA2, F  ; Decrementa CUENTA2, guarda el
                            ; resultado en f, y si es cero se salta la
                            ; siguiente instrucción
        GOTO    ACA         ; vuelve a decrementar mientras
                            ; CUENTA2 no sea cero
        DECFSZ  CUENTA1, F  ; Se decrementa CUENTA1 cada vez que
                            ; CUENTA2 llega a cero
        GOTO    ACA1        ; mientras CUENTA1 no llegue a cero
                            ; recarga CUENTA2 y repite el proceso
        RETURN              ; retorna al programa principal

; FIN DE LA SUBRUTINA DELAY

        END                 ; Fin del programa
```

**INTERRUPCIONES.**

Cuando se produce una interrupción el PIC deja automáticamente lo que esté haciendo, va directo a la dirección 04h de programa y ejecuta lo que encuentre a partir de allí hasta encontrarse con la instrucción RETFIE que le hará abandonar la interrupción y volver al lugar donde se encontraba antes de producirse dicha interrupción.

Para que se pueda producir una interrupción hay que habilitar las interrupciones globalmente y la interrupción en concreto que queremos utilizar (con el registro INTCON). Este PIC tiene 4 tipos de posibles interrupciones:

1. Por cambio en los bits RB4-RB7
2. Por el estado de RB0
3. Por desbordamiento del Timer-contador
4. Por fin de ciclo de escritura de la EEPROM de datos

Mientras se está ejecutando una interrupción no se puede producir otra interrupción, el PIC no lo permite. Una cosa importante a tener en cuenta al usar interrupciones es que cuando estas se producen podríamos estar trabajando con registros que pueden ser modificados en la propia interrupción, como el acumulador o el STATUS.

Para que la interrupción no eche a perder el buen funcionamiento del programa principal conviene guardar los valores de estos registros en otras variables que no vayamos a modificar. Antes de salir de la interrupción volvemos a restaurar los valores guardados y todo solucionado.

El siguiente ejemplo muestra un programa que usa la interrupción por cambio en el puerto B (En pines RB4 a RB7)

```
;****************************************
;*    USO DE INTERRUPCIONES             *
;****************************************
;* Este programa invierte el estado del pin      *
;* RA0 cada vez que se modifica el estado        *
;* De alguno de los pines RB4, RB5, RB6 o        *
;* RB7. Para ello habilita la interrupción       *
;* Por cambio de RB4-RB7              *
;****************************************

STATUS      EQU         03H
TRISA       EQU         05H
PORTA       EQU         05H
TRISB       EQU         06H
PORTB       EQU         06H
INTCON          EQU         0BH

ACUM        EQU         0CH
STAT        EQU         0DH

F           EQU         1
W           EQU         0

#DEFINE     BANCO0      BCF     STATUS, 5
#DEFINE     BANCO1      BSF     STATUS, 5

            ORG         00H
            GOTO        INICIO      ; ponemos este GOTO al principio para
                                    ; poder poner el subprograma de las
                                    ; Interrupciones a partir de
                                     ; La dirección 04h

                                    ; Comienza la interrupción:
                                    ;=======================

            ORG         04H         ; El PIC salta a esta dirección cuando se
                                    ; produce una interrupción
            BCF         INTCON, 0   ; bit que indica un cambio en RB4-RB7,
```

```
                                        ; recuerda que hay que ponerlo a "0" por
                                        ; programa, este es el momento

                                        ; comenzamos guardando el contenido del
                                        ; acumulador y del STATUS para
                                        ; restaurarlos antes de salir de la
                                         ; Interrupción (es recomendable hacer esto
                                        ; Siempre que se usen interrupciones)
                MOVWF     ACUM          ; Copia el acumulador al registro ACUM
                MOVF      STATUS, W     ; Guarda STATUS en el acumulador
                BANCO0                  ; Por si acaso, nunca se sabe en que parte
                                        ; de programa principal salta la
                                         ; Interrupción

                MOVWF     STAT          ; Copia el acumulador al registro STAT
                                        ; Invertimos el estado de RA0:
                                        ;==========================
                BTFSC     PORTA,0       ;si RA0 es "0" salta la siguiente instrucción
                GOTO      ESUNO         ; vete a ESUNO
                BSF       PORTA,0       ;Pon a "1" RA0 (porque era "0")
                GOTO      HECHO         ; ya está invertido RA0, vete a HECHO

ESUNO           BCF       PORTA,0       ;Pon a "0" RA0 (Porque era "1")

                                        ; Ya se ha invertido el estado de RA0
                                        ;==================================

                                        ; Ahora hay que restaurar los valores del
                                        ; STATUS y del acumulador antes de salir
                                         ; De la interrupción:


HECHO           MOVF      STAT,W        ;Guarda el contenido de STAT en el
                                         ; Acumulador
                MOVWF     STATUS        ; Restaura el STATUS
                SWAPF     ACUM,F        ;Da la vuelta al registro ACUM
                SWAPF     ACUM,W        ;Vuelve a dar la vuelta al registro ACUM y
                                        ; restaura el acumulador (Con la instrucción
                                        ; SWAPF para no alterar el STATUS, la
                                        ; Instrucción MOVF altera el bit 2 del
                                        ; STATUS)
                RETFIE                  ; fin de la interrupción
```

```
INICIO      BANCO1                  ; Pasamos al banco 1
            MOVLW       0FFH        ; Todos los bits del acumulador a "1"
            MOVWF       TRISB       ; configuramos todo el puerto B como
                                     ; entradas
             BCF        TRISA,0     ;RA0 como salida
            BANCO0                  ; Volvemos al banco 0

                                    ; Configuración de las interrupciones:
                                    ;================================

            BSF         INTCON, 7   ; Habilita las interrupciones globalmente
            BSF         INTCON, 3   ; Habilita la interrupción por cambio de
                                     ; Puerto B

                                     ;================================
                                    ; ya están configuradas las interrupciones, a
                                    ; partir de ahora cuando haya un cambio en
                                    ; RB4-RB7 saltará la interrupción (a la
                                    ; Dirección 04h de programa)

NADA        GOTO        NADA        ; En este ejemplo no se hace nada en el
                                    ; programa principal, simplemente se
                                    ; espera a que salte la interrupción. La
                                    ; verdadera utilidad de las interrupciones
                                     ; es que se pueden hacer "cosas"
                                    ; Mientras sin preocuparse de la
                                     ; Interrupción

            END                     ; FIN DE PROGRAMA
```

## TIMER - CONTADOR TMR0.

El registro TMR0 puede contar ciclos de instrucción interna o pulsos de entrada por RA4 según el valor del bit 5 del registro OPTION (TOCS). Si este bit está a "1" TMR0 cuenta pulsos por RA4 y se le llama Contador; si el bit está a "0" cuenta ciclos de instrucción interna y se le llama Timer.

Cada ciclo de instrucción dura 4 veces el ciclo del reloj del PIC (para un reloj de 4MHz ==> Ciclo reloj=0,25 µSeg ==> Ciclo instrucción = 4 X 0,25 = 1µSeg).

Cuando lo usamos como contador (Por RA4) podemos determinar si el incremento se hará por flanco ascendente o descendente con el bit 4 del registro OPTION (TOSE)

Podemos leer o escribir el registro TMR0 en cualquier momento. Cuando escribamos en él deja de contar durante dos ciclos, cuando lo leamos no pasa nada.

Podemos asignar el prescaler al TMR0, si hacemos esto podemos elegir el factor en el que se verá dividido el conteo mediante los bits del 0 al 2 del registro OPTION según la tabla del factor de división. Por ejemplo, si elegimos un factor de división de 1/2 tienen que entrar 2 pulsos para que TMR0 se incremente en uno, si está a 1/4 tienen que entrar 4... etc.

También podemos utilizar la interrupción que se produce cuando se desborda el TMR0, es decir, cuando pasa de FFh a 00h. (se configura desde el registro INTCON)

El siguiente ejemplo usa la interrupción por desbordamiento de TMR0 para obtener una onda cuadrada a la salida RB0:

```
;*****************************************
;*       EJEMPLO 4: USO DEL TMR0         *
;*****************************************
;* Este programa crea una señal cuadrada a              *
;* La salida RB0, para ello utiliza el TMR0                     *
;* Y la interrupción por desbordamiento del             *
;* Mismo. Se le asignará el prescaler con               *
;* Un factor de división de 1/2. De esta                *
;* Forma las interrupciones saltarán a                  *
;* Intervalos fijos de tiempo. Invirtiendo              *
;* El estado de RB0 durante las                 *
;* Interrupciones se conseguirá una onda                *
;* Cuadrada perfecta                    *
;*****************************************


OPTIONR         EQU             01H             ; Registro para configuración del
                                                 ; TMR0
STATUS          EQU             03H
TRISB           EQU             06H
PORTB           EQU             06H
INTCON                  EQU             0BH

ACUM            EQU             0CH
STAT            EQU             0DH

F               EQU             1
W               EQU             0

#DEFINE         BANCO0          BCF     STATUS, 5
#DEFINE         BANCO1          BSF     STATUS, 5

                ORG             00H

                GOTO            INICIO          ; ponemos este GOTO al principio
                                                 ; Para poder poner
                                                ; El subprograma de las
                                                 ; Interrupciones a partir de
                                                ; La dirección 04h
                                                ; Comienza la interrupción
                ORG             04H             ; El PIC salta a esta dirección cuando
                                                 ; Se produce una interrupción
```

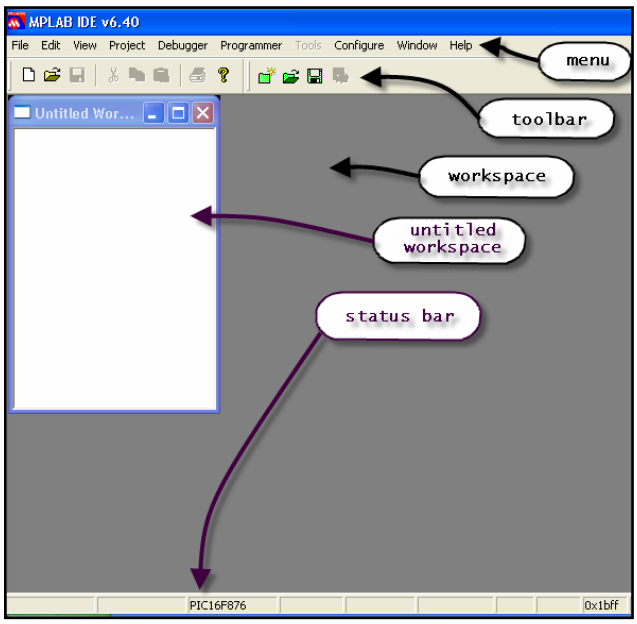
```
                BCF     INTCON, 2       ; bit que indica desbordamiento de
                                        ; TMR0, recuerda que hay que
                                        ; ponerlo a "0" por programa, este
                                        ; es el momento

                                        ; comenzamos guardando el
                                        ; contenido del acumulador
                                        ; Y del STATUS para restaurarlos
                                        ; Antes de salir de
                                        ; La interrupción (es recomendable
                                        ; hacer esto siempre que se usen
                                        ; Interrupciones)

                MOVWF   ACUM            ; Copia el acumulador al registro
                                        ; ACUM
                MOVF    STATUS, W       ; Guarda STATUS en el acumulador
                BANCO0                  ; Por si acaso, nunca se sabe en que

                                        ; Parte de programa principal salta
                                        ; La interrupción
                MOVWF   STAT            ; Copia el acumulador al registro
                                        ; STAT

                                        ; Para generar una onda cuadrada
                                        ; Invertimos el estado de RB0 cada
                                        ; Vez que salta una interrupción
                                        ;====================


                BTFSC   PORTB, 0        ; si RB0 es "0" salta la siguiente
                                        ; Instrucción
                GOTO    ESUNO           ; vete a ESUNO
                BSF     PORTB, 0        ; Pon a "1" RB0 (porque era "0")
                GOTO    HECHO           ; ya está invertido RB0, vete a
                                        ; HECHO

ESUNO           BCF     PORTB, 0        ; Pon a "0" RA0 (Porque era "1")
                                        ; Ya se ha invertido el estado de
                                        ; RB0
                                        ; Ahora hay que restaurar los
                                        ; Valores del STATUS y del
                                        ; Acumulador antes de salir de la
                                        ; Interrupción:
```

```
HECHO       MOVF        STAT, W     ; Guarda el contenido de STAT en el
                                    ; Acumulador
            MOVWF       STATUS      ; Restaura el STATUS
            SWAPF       ACUM, F     ; Da la vuelta al registro ACUM
            SWAPF       ACUM, W     ; Vuelve a dar la vuelta al registro
                                    ; ACUM y restaura el acumulador
                                    ;(Con la instrucción SWAPF para no
                                    ; alterar el STATUS, la instrucción
                                    ; MOVF altera el bit 2 del STATUS)
            RETFIE                  ; fin de la interrupción

                                    ; Fin de la interrupción

INICIO      BANCO1                  ; Pasamos al banco 1
            BCF         TRISB, 0    ; RB0 como salida
            BCF         OPTIONR, 3  ; Asignamos el Preescaler a TMR0
            BCF         OPTIONR, 0  ; \
            BCF         OPTIONR, 1  ;} Prescaler a 1/2
            BCF         OPTIONR, 2  ; /
            BCF         OPTIONR, 5  ; Entrada de TMR0 por ciclo de
                                    ; Instrucción interna (se pone a
                                    ; contar)
            BANCO  0                ; Volvemos al banco 0
                                    ; Configuración de las
                                    ; Interrupciones:
            BSF         INTCON, 7   ; Habilita las interrupciones
                                    ; Globalmente
            BSF         INTCON, 5   ; Habilita la interrupción por
                                    ; Desbordamiento de TMR0
                                    ; Ya están configuradas las
                                    ; Interrupciones, a partir de ahora
                                    ; Cuando cuando se desborde TMR0
                                    ; saltará la interrupción (a la
                                    ; Dirección 04h de programa)
NADA        GOTO        NADA        ; En este ejemplo no se hace nada
                                    ; En el programa principal,
                                    ; Simplemente se espera a que salte
                                    ; La interrupción. La verdadera;
                                    utilidad de las interrupciones es
                                    ; Que se pueden hacer "cosas"
                                    ; Mientras sin preocuparse de las
                                    ; Interrupciones

            END                     ; FIN DE PROGRAMA
```

## *MPLAB IDE*

### *ABRIR EL PROGRAMA*

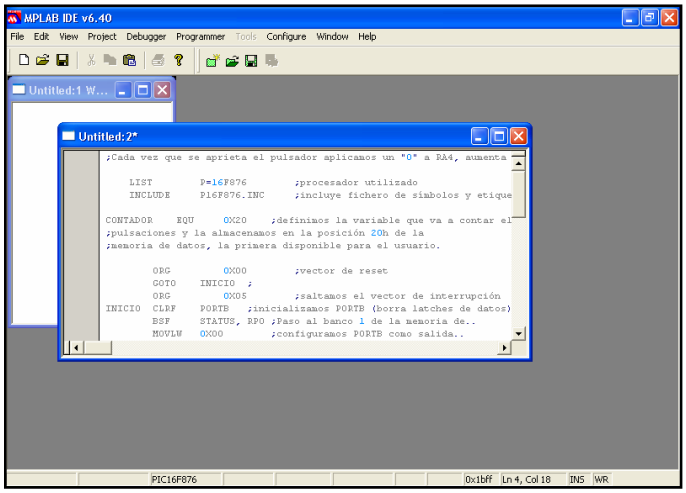
**1.** Se abre el programa MPLAB IDE haciendo doble clic sobre icono de MPLAB en el escritorio.



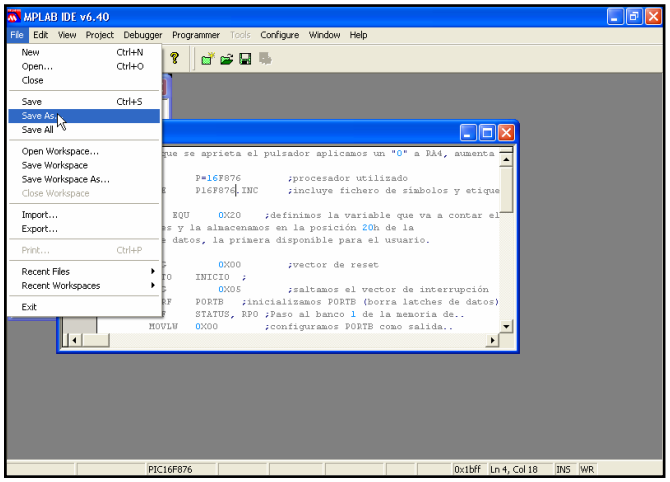

**2.** ***Crear un código fuente.***

Escribe el código fuente del programa usando el editor de textos del MPLAB, para ello se deben de seguir los siguientes pasos: seleccionar ***File – New,*** se abre una ventana en blanco en la zona de trabajo

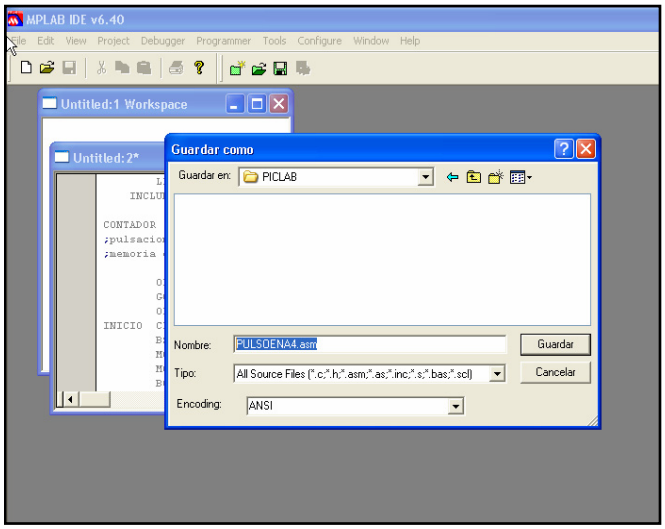
(***Workspace***). Se puede escribir en ella el programa a simular o directamente “pegar” el programa si proviene de otro documento.



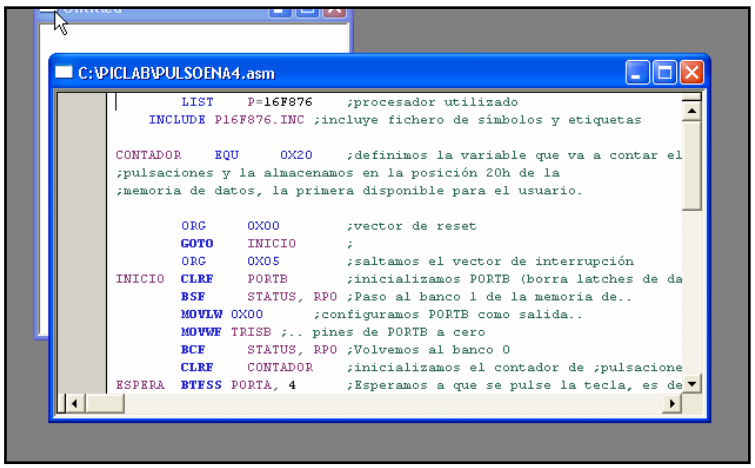

**3.** ***Guardar*** el documento en blanco utilizando ***File – Save as***




En el ejemplo lo guardamos en C:\PICLAB con el nombre PULSOENA4.asm



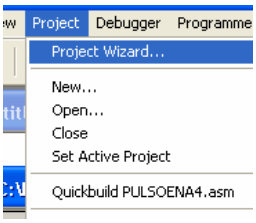

Se debe de tener en cuenta que sólo admite un path con un número menor de 64 caracteres. Una vez guardado, vemos en la pantalla que cambia el color del texto editado (etiquetas, operandos, comentarios, etc.).



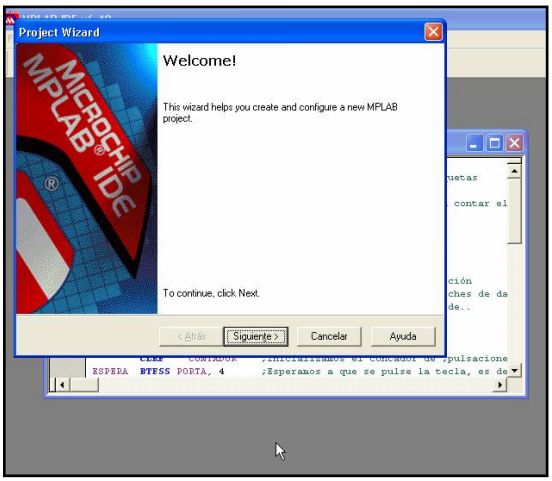
## 4. *Crear un proyecto*

Se debe preparar el proyecto, para no olvidar pasos intermedios es conveniente utilizar el ***MPLAB Wizard***, que llevará paso a paso la ejecución del mismo.

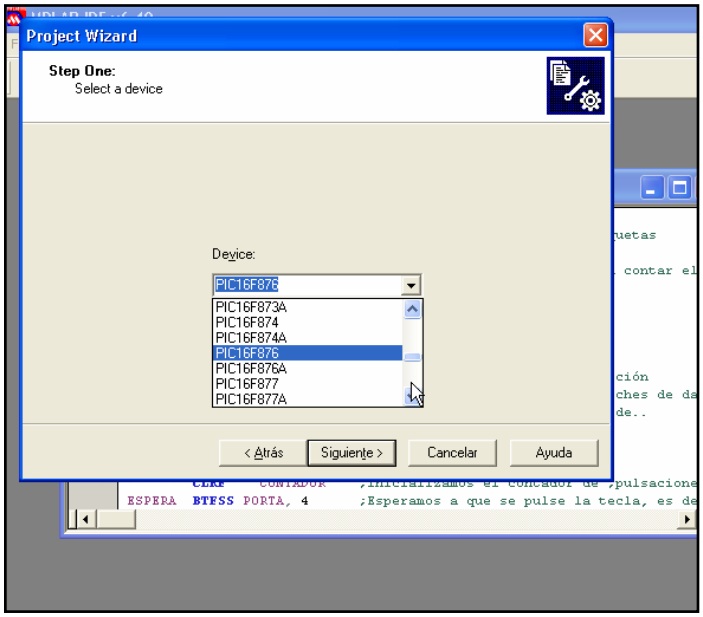
Abrir el Wizard. Para ello seleccionar ***Project – Project Wizard***.




Seleccionar ***Siguiente*** para continuar.

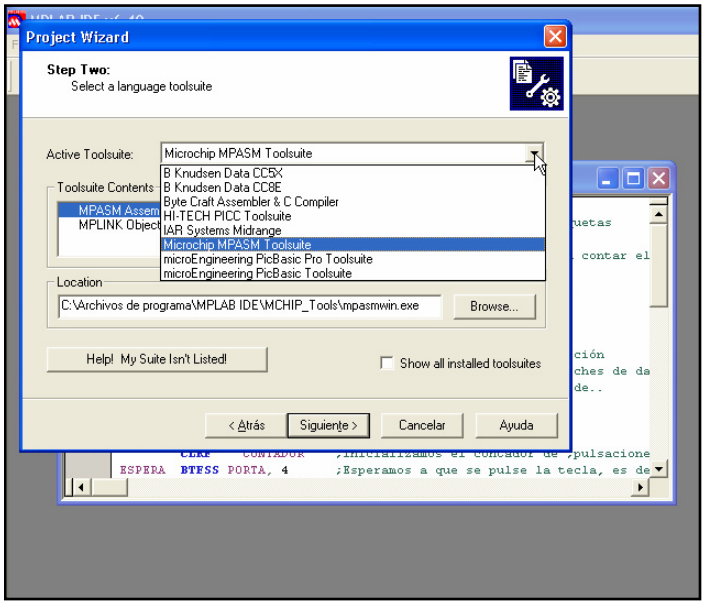


Seleccionar el dispositivo a utilizar. Abriendo la ***pestaña*** se elige el micro controlador, en nuestro caso, el ***PIC16F84***, luego seleccionamos ***Siguiente*** para continuar.

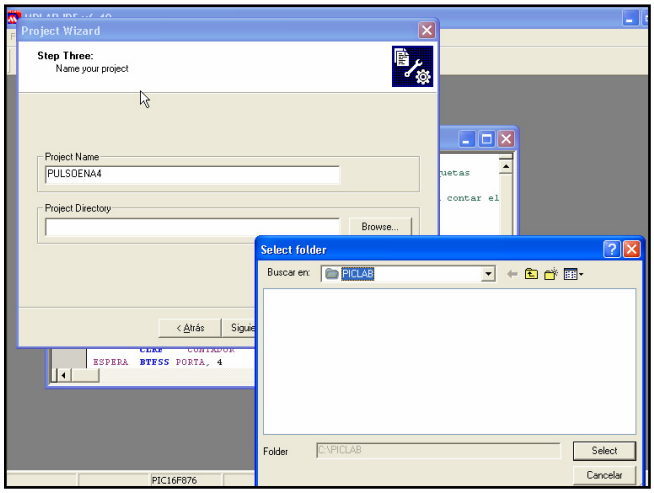



Seleccionar la ***Herramienta***. Abrir la pestaña en ***Active Toolsuit*** y elegir ***MPASM*** que es el ejecutable ensamblador. Si está instalado el programa con los valores por defecto, aparece la pantalla como en la vista, sino, habrá que buscarlo utilizando el ***Browse***.

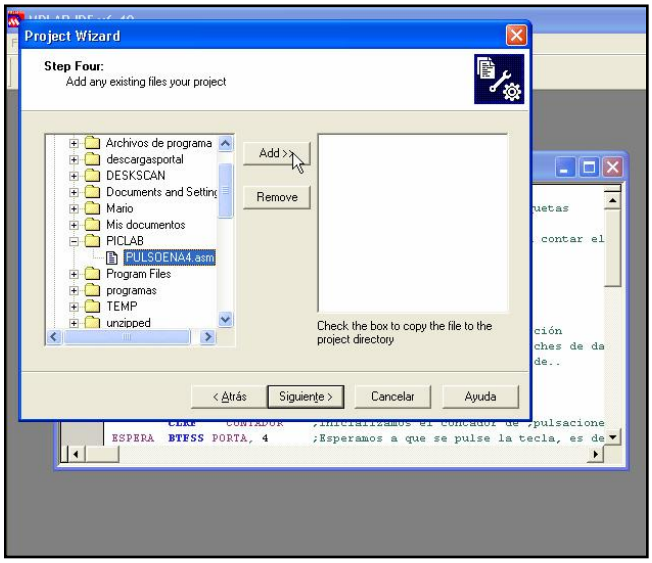
Seleccionar ***Siguiente*** para continuar.



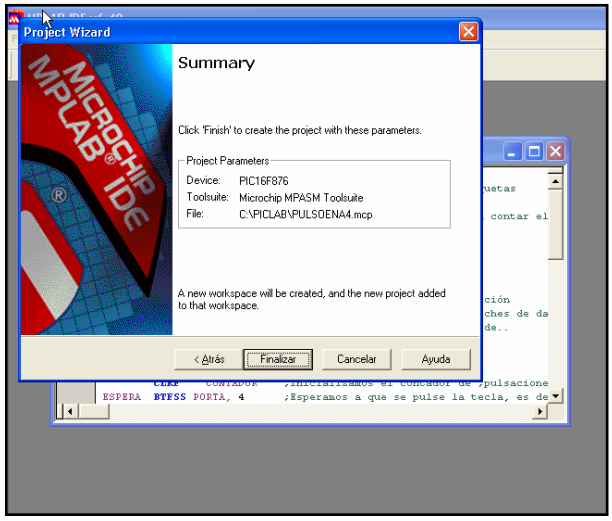

Poner nombre al proyecto. Le ponemos el mismo nombre que el ***asm*** creado, será entonces ***PULSOENA4*** y lo colocamos en C:\PICLAB. Seleccionamos ***Siguiente*** para continuar.



Agregar archivos al proyecto. Buscar en C:\PICLAB el fichero ***PULSOENA4.asm*** grabado anteriormente y agregarlo (***Add***).



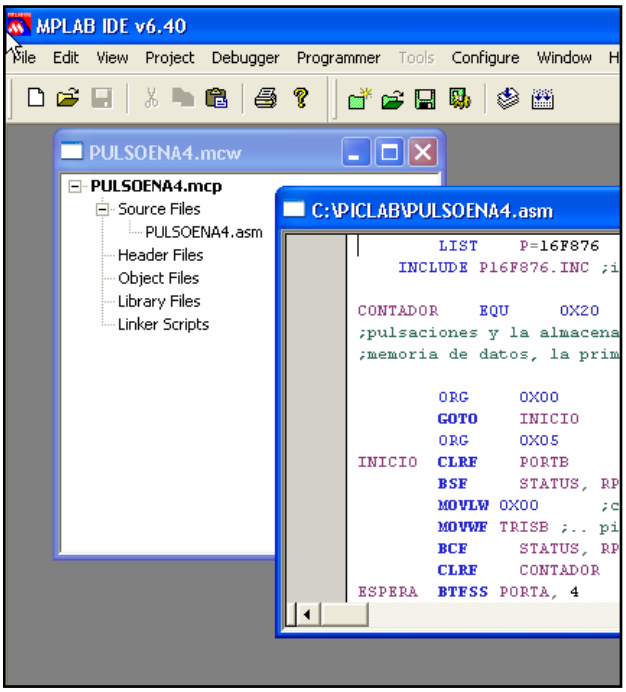

Se completa así la creación del proyecto dando la información sobre el mismo. Seleccionar ***Finalizar*** para salir del Wizard



***ENSAMBLAJE Y SIMULACIÓN.***

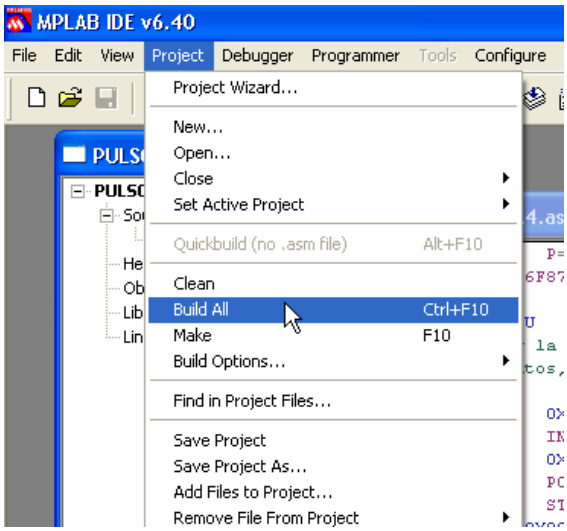
Una vez creado el proyecto se deben de realizar los siguientes pasos:

**5.** ***Ventana del proyecto***. Los archivos pueden agregarse, guardarse o borrarse en esta ventana. Para ello se hace clic con el botón derecho del ratón luego de seleccionarlo.

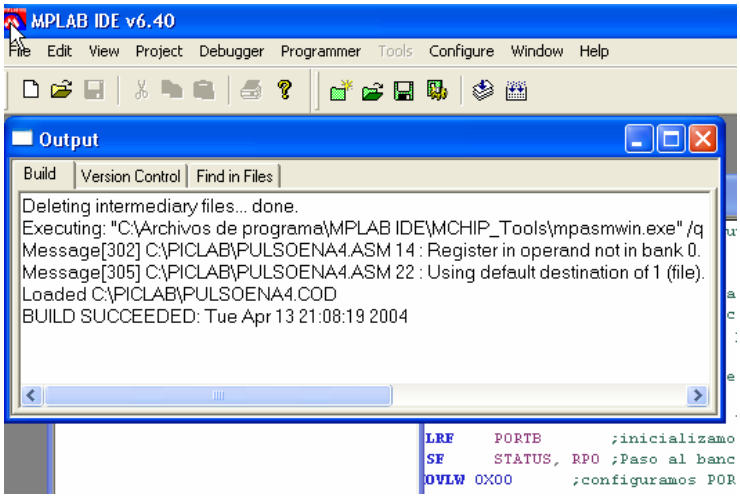



**6.** ***Construcción del proyecto.***

Una vez creado el proyecto, se debe de construir. Es decir, se debe ensamblar. Seleccionar ***Project – Build All*** para construir el mismo.




Si se efectuó de forma correcta, aparecerá la siguiente pantalla (***Ventana de salida***).



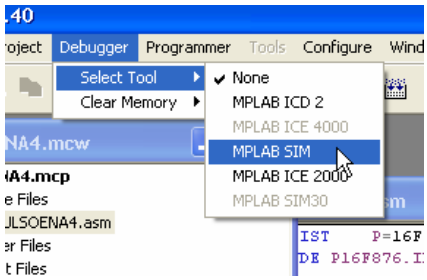
Se muestran en esta ventana los ***errores de sintaxis encontrados, los warnings (atenciones) y mensajes.***

Si hubiera errores habría que arreglarlos antes de proseguir. Haciendo doble clic sobre la línea que señala (y explica) el error, cambia la ventana al editor de texto y ubica una flecha verde en la línea donde se encuentra el error a corregir. Una vez corregido se debe volver a construir hasta que desaparezcan todos los errores y warnings.

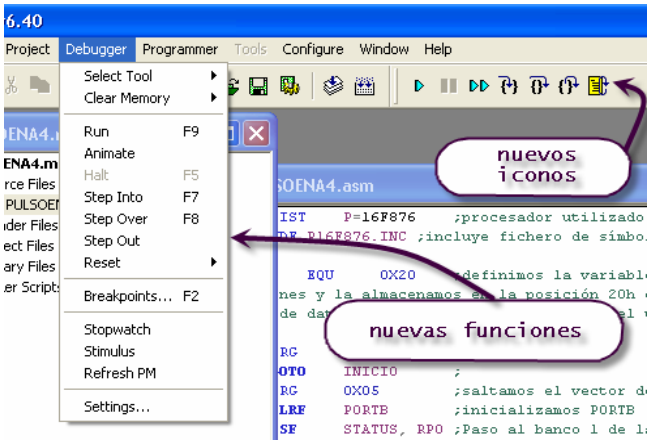
## *UTILIZACIÓN DEL MPLAB SIM*

Una vez llegado a este último paso, se puede comenzar a verificar el funcionamiento del programa. Para ello hay que cambiar la herramienta que utilizamos en el MPLAB, de ***MPASM (ENSAMBLADOR)*** al ***MPLAB SIM (SIMULADOR***).

Seleccionar ***Debugger – Select Tool – MPLAB SIM***.

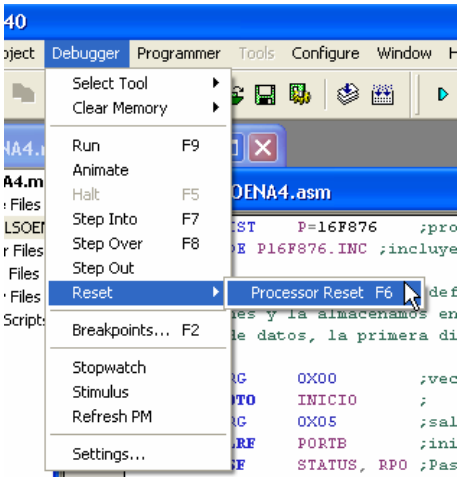


Una vez seleccionado el ***Simulador***, se habilitan otras funciones e iconos.

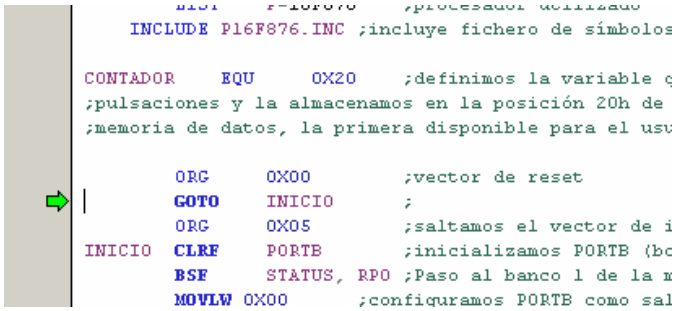



***SIMULACIÓN DEL PROGRAMA.***

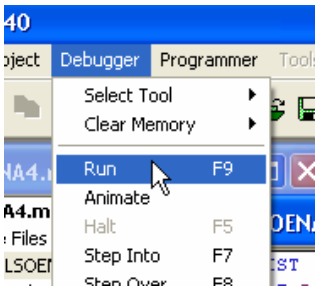
Antes de hacer correr el programa se debe reiniciar para indicar cual es la primera línea del mismo. Se puede hacer utilizando el menú o con el icono correspondiente. Con el menú se selecciona ***Debugger – Reset – F6***.



En la pantalla donde tenemos el texto del programa aparece una flecha verde indicando la posición de comienzo.



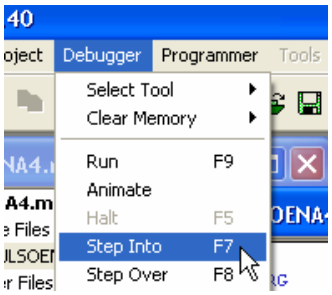

En estos momentos se puede hacer correr el programa, para ello seleccionamos ***Debugger – Run***.




Aparece en el ***status bar*** (barra de estados) ***running***, indicando el estado actual.

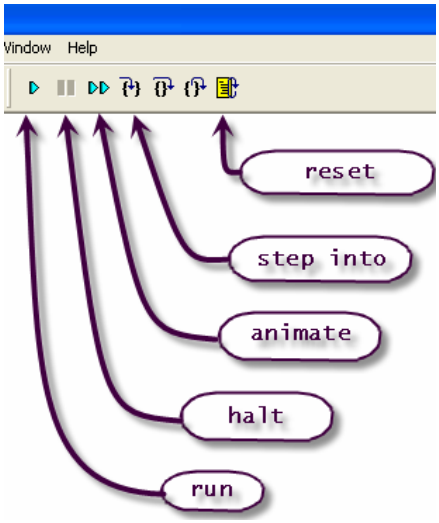

Para parar el programa se realiza con ***Debugger – Halt***. Quedará señalizado el punto donde para con la flecha verde. Para realizar la simulación paso a paso se selecciona ***Debugger – Step Intro***. Es muy útil para la

depuración del programa. Otra opción, es la utilización de la animación, para ello se selecciona ***Debugger – Animate***. Se observa en forma continua como se va ejecutando el programa a través del movimiento de la flecha verde en el lado izquierdo del programa.



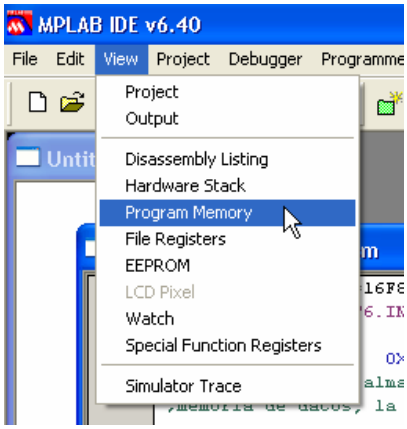

Para todos los casos anteriores se puede utilizar los iconos o las techas abreviadas.

| Debugger Menu | Toolbar Buttons | Hot Key |
|---|---|---|
| Run | | F9 |
| Halt | | F5 |
| Step Into | | F7 |
| Reset | | F6 |

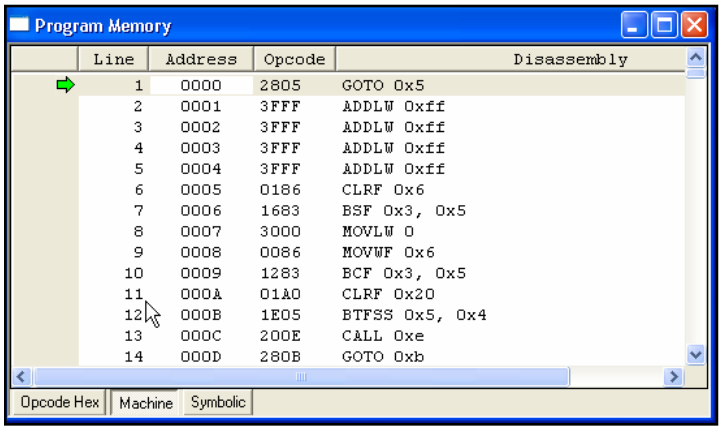


***OBSERVACIÓN DE MEMORIAS (EEPROM Y RAM).***

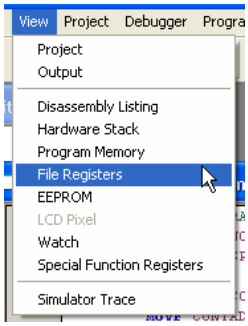
Si deseamos ver el contenido de la memoria de ***programa*** se puede seleccionar ***View – Program Memory***.



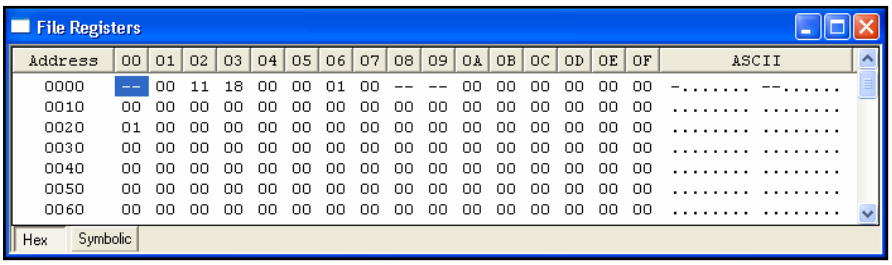

Aparece la ventana



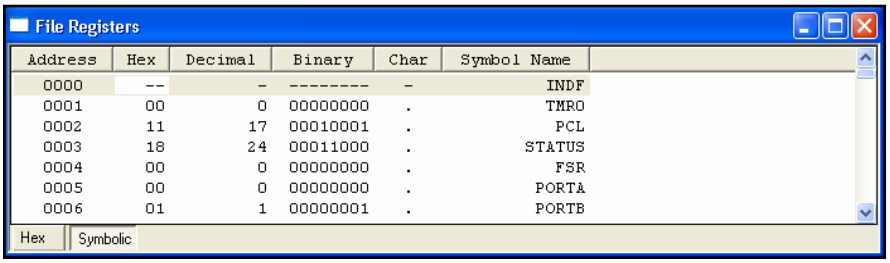
Si en cambio deseamos ver el contenido de la memoria ***RAM***, el procedimiento es similar, seleccionamos ***View – File Registers***.



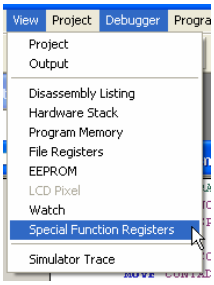

El resultado, será entonces, el mostrado por defecto con la opción ***Hex*** (hexadecimal).



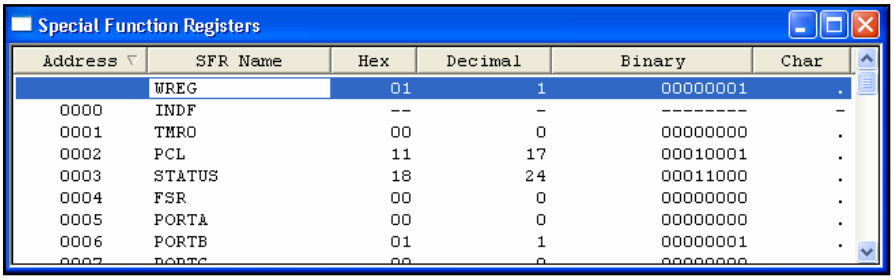

Si ponemos la opción ***Symbolic***, la información mostrada puede llegar a ser más útil, ya que es más fácil reconocer los registros al traer el nombre además de su dirección. Por supuesto, el número de registros mostrados es menor.




Podemos llegar a una ventana similar utilizando la opción ***View – Special Function Registers***. (Prácticamente es la misma tabla, pero tiene el registro de trabajo ***Work Register***)




La ventana que aparece en este caso, es muy similar a la anterior

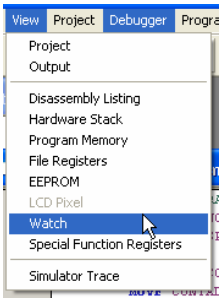


***VENTANAS A MEDIDA***

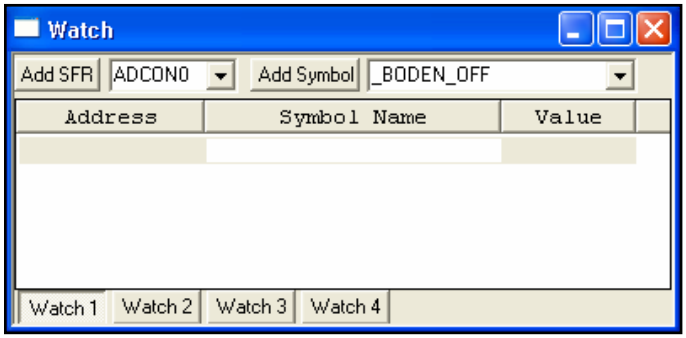
Normalmente, no es práctico tener en una ventana demasiada información, o por lo menos información que no es útil. Si será aconsejable tener una ventana donde podamos visualizar solo los Registros y / o Variables que se modifiquen en el programa y que nos sean de utilidad.

Entonces crearemos una única ventana de visualización a medida para nuestro proyecto. En ésta podremos visualizar Registros, Variables, Contadores, Puertos, etc.

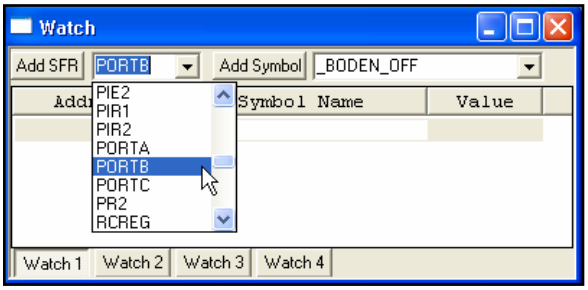
Para abrir esta ventana "***a medida***", seleccionamos ***View- Watch.***



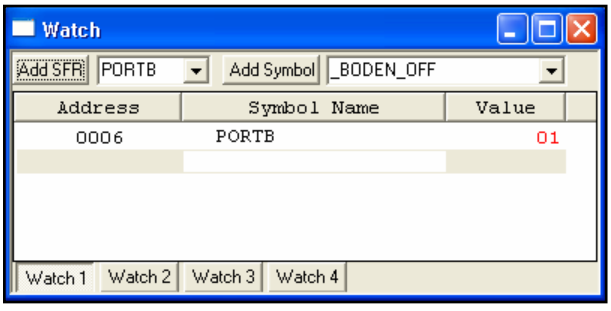

Aparece la ventana siguiente



Si lo que deseamos visualizar son ***Variables***, la forma de hacerlo será abriendo la pestaña en ***Add SFR*** y seleccionando cada uno de los Registros que nos interesen

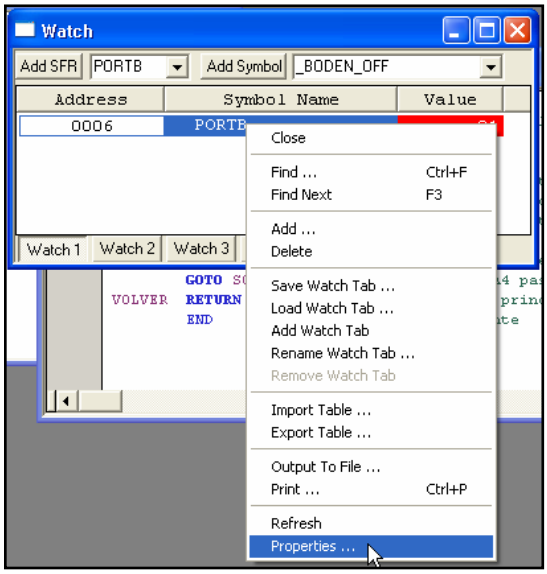



Al teclear ***Add SFR***, se añade el valor seleccionado a la ventana tal como vemos a continuación.

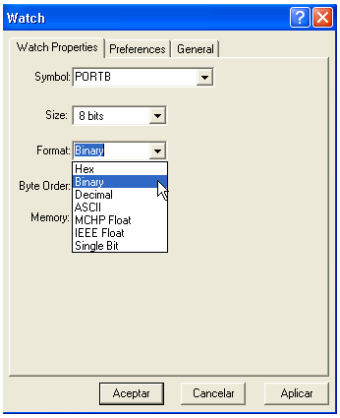



El valor que tiene el registro (señalado en color rojo) está dado en ***hexadecimal***. Si deseamos cambiarlo, por ejemplo en nuestro caso nos interesa tenerlo en binario para saber si los leds se encenderán o no, debemos

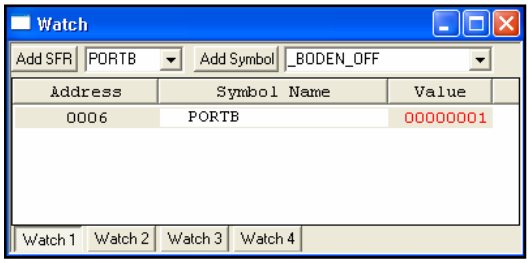
seleccionar ***PORTB*** y luego el botón derecho del ratón. De la nueva lista abierta seleccionaremos ***Properties*** (Propiedades).



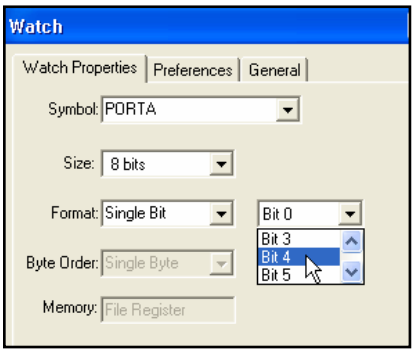

Ahora, cambiamos el valor que viene por defecto (Hex) a ***Binary*** (binario) y aceptamos.



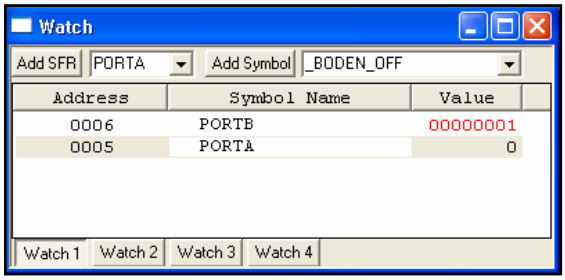
Al ***Aceptar***, aparece nuevamente la ventana de Visualización con el valor del ***Puerto B*** en binario.



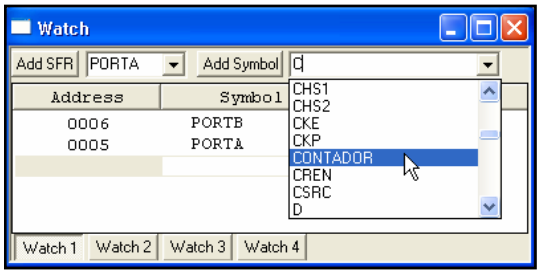

Haremos lo mismo pero ahora con el ***PORTA***, pero en este caso, sólo nos interesa el valor del bit 4. Elegimos la opción de ***Single Bit*** (único bit)



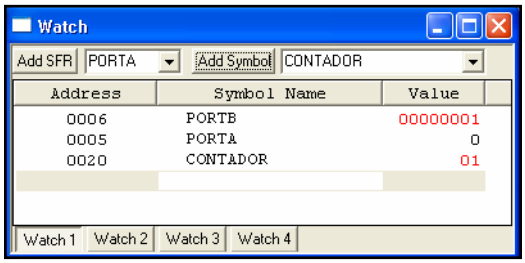

La pantalla de visualización quedará como sigue



Para agregar otras ***Variables***, como por ejemplo ***CONTADOR***, abrimos la pestaña en ***Add Symbol*** hasta encontrarlo, o podemos comenzar a escribir el nombre




Nuevamente lo seleccionamos y hacemos clic en ***Add Symbol***. El valor por defecto nos lo agrega en Hexadecimal, y lo que haremos es cambiarlo a Decimal.

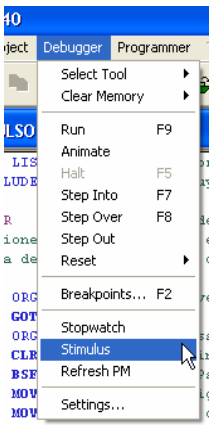



Siguiendo los mismos pasos, creamos la ventana con los valores que nos interesa. Si se mantiene esta ventana abierta mientras que se ejecuta el programa, los valores de los datos mostrados van actualizándose. Si en el paso anterior se modificó algún dato, éste se pone en rojo, en caso contrario permanece en negro.

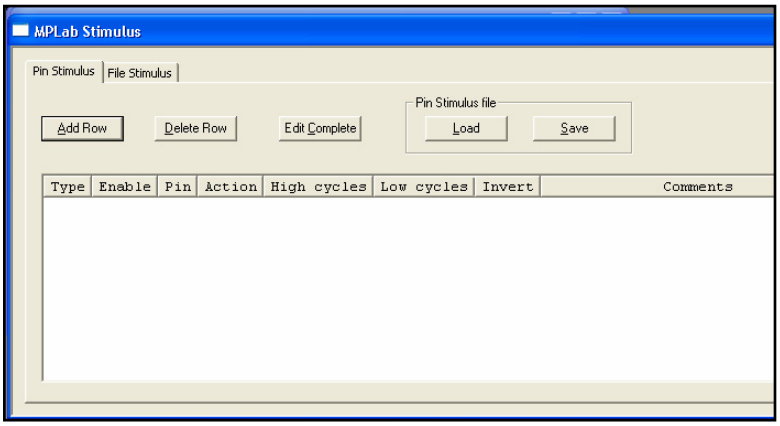
***ESTÍMULOS.***

Para simular mejor las condiciones reales, tales como entradas a nivel alto o bajo de un pin, el simulador provee mecanismos conocidos como estímulos.

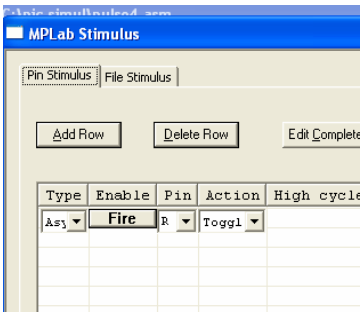
El primer tipo de estímulo se llama "***estímulo de pin***" el cual provee estímulos síncronos o asíncronos a las E/S de los pines. Los ***síncronos*** están sincronizados con los ciclos de instrucción del dispositivo que se está simulando, y los estímulos ***asíncronos*** se aplican por el usuario en tiempo real mientras que se ejecuta la simulación. Solamente veremos los ***estímulos de pin asíncronos***. Seleccionar ***Debugger - Stimulus*** y hacer clic el ***Pin Stimulus tab***.




Hacer clic ***Add Row*** para agregar una fila para colocar el dato de entrada.




Si hacemos clic en la fila agregada (debajo de cada columna), seleccionamos las características del pulso, en nuestro caso deseamos poner un pulso asíncrono en la pata RA4 y que cada vez que se pulse cambie de estado. Entonces seleccionamos ***Async*** para ***Type***, es decir, se elige el tipo de pulso (síncrono o asíncrono) ***Pin*** se abre la pestaña para elegir el RA4. ***Action*** con sus cuatro opciones (poner en uno, en cero, cambio y pulso)

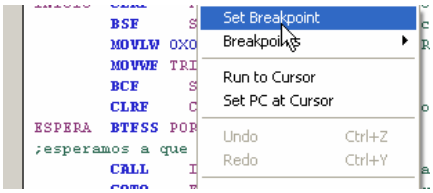


Cada vez que se haga clic sobre la posición ***Fire*** el valor del pulso cambiará de 0 a 1 y de 1 a 0. Si se tiene abierta la ventana de visualización se podrá comprobar los cambios en RA4.

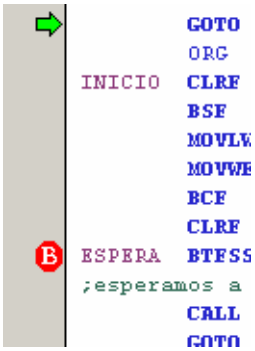
***PUNTOS DE RUPTURA.***

Se pueden colocar puntos de ruptura para que el programa se detenga en otro momento que no sea el final del mismo. Para ello, seleccionar ***Debugger – Reset*** para iniciar la aplicación.

Luego buscar la línea donde se desea agregar el punto de ruptura, hacer clic con el botón derecho y seleccionar ***Set Breakpoint***.

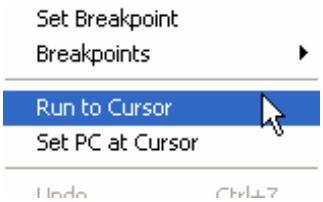



Aparecerá una señal de ***Break (rotura)*** al margen izquierdo cerca de la línea.



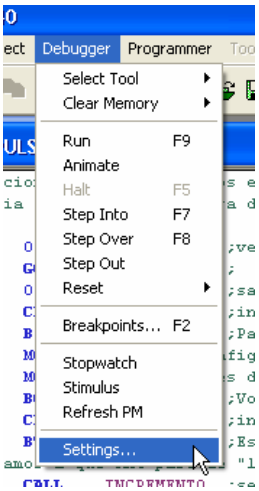
Si seleccionamos ***Debugger – Run***, el programa correrá hasta el punto de ruptura creado.

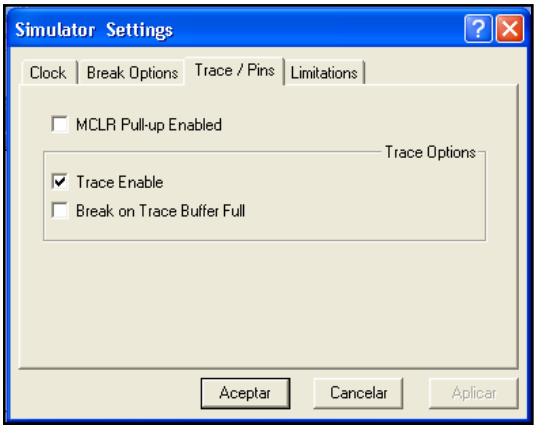

Para salir de esta situación, se coloca el cursor del ratón en alguna línea posterior y con el botón de la derecha del ratón se selecciona ***Run to Cursor***.




***TRACING CODE***

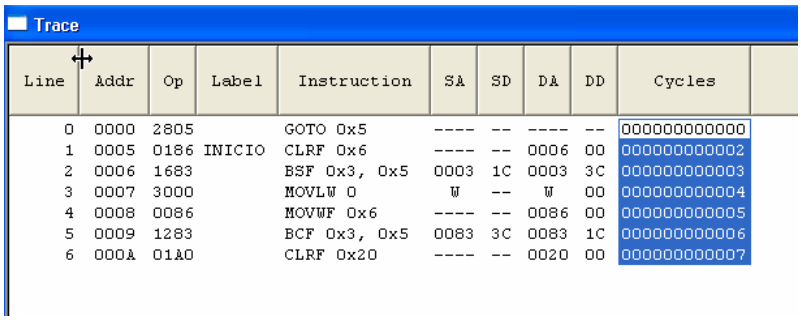
Se utiliza para grabar los pasos de la ejecución del programa. Para habilitarlo se debe seleccionar ***Debugger – Settings*** y luego elegir ***Trace / Pins***.




Hay dos opciones en el área ***Trace Options*** para controlar la simulación. Con la primera opción, ***Trace Enable***, el simulador coge los datos cuando funciona en modo ***Run***. Recoge datos hasta que se detiene en un punto de ruptura o manualmente. Muestra como máximos 8192 ciclos. Se utiliza para ver las instrucciones hasta un punto de ruptura.

Si se selecciona también la segunda opción, la memoria guardará otros 8192 ciclos más de datos, luego para de recogerlos y se detiene en un punto de ruptura. Este modo se utiliza para ver las instrucciones ejecutadas y grabadas luego de seleccionar un ***Run***. Para visualizar estas instrucciones se debe seleccionar ***View – Simulator Trace***.




Como se observa en la ventana, se pueden verificar otros datos:

***Line.*** Número de línea

***Addr.*** Dirección del contador de programa.

***Op.*** Código de la instrucción

***Label***. Etiqueta

***Instruction.*** Instrucción sin ensamblar

Valores leídos o escritos en los registros:

***SA Sourcce Address.*** Dirección del registro de operaciones de lectura

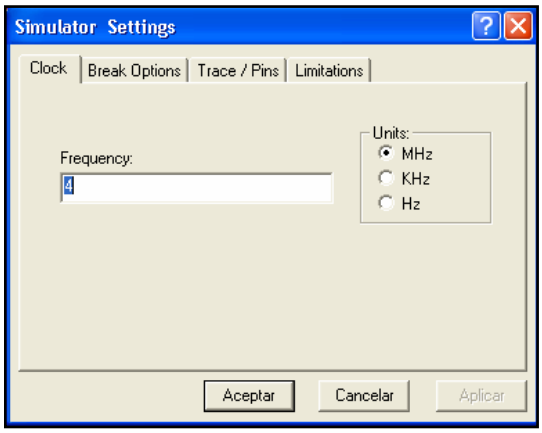
***SD Source Data.*** Dato leído del registro

***DA Destination Address.*** *D*estino del registro en operación de escritura

***DD Destination Data.*** *D*ato escrito en el registro

**--** No hubo acceso a ningún registro en la instrucción

***Cycles.*** *N*úmero de ciclos que ha llevado la ejecución del programa. Se utiliza para medir el tiempo de ejecución de las rutinas. El tiempo se calcula basado en la frecuencia de reloj que se colocó en ***Debugger – Settings – Clock Tab***.

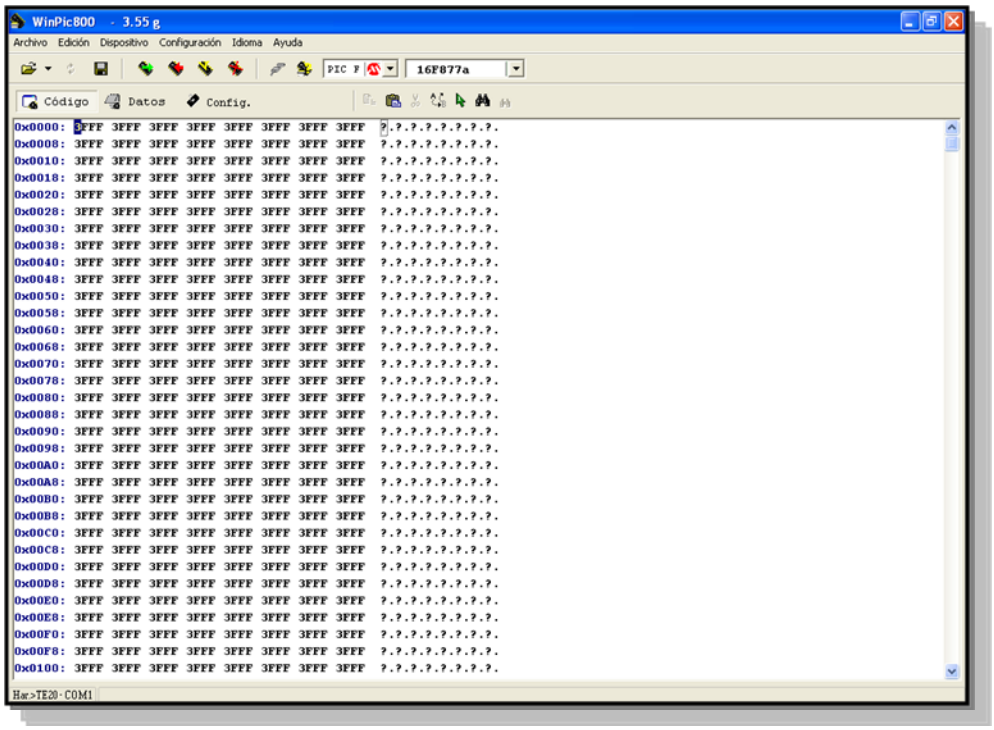



Para guardar y cerrar el proyecto y campo de trabajo se deben seguir los siguientes pasos:

***Select Project - Save Project Select File – Workspace - Save Select File – Workspace - Close***

***WINPIC800***

WinPic800 es un software diseñado para programar Micro controladores, usando un dispositivo grabador de chips el cual puede ser conectado por medio de un puerto serial, paralelo o USB. WinPic800 soporta una gran variedad de Micro controladores PIC, así como también muchos equipos grabadores. WinPic800 posee una sencilla interfaz gráfica la cual nos permite visualizar el programa (en Hex) que deseamos introducir en el Micro controlador, de la misma forma nos permite ver la información contenida dentro de un Micro controlador grabado previamente. WinPic800 es freeware, por lo tanto es gratuito y es posible distribuirlo sin ningún inconveniente.



A continuación se presentan los diferentes pasos a tomar para la grabación de un Micro controlador utilizando WinPIC800.

Lo primero es asegurarnos de que el WinPIC800 este correctamente configurado. Para ello dispone en el menú principal de la opción “Configuración”. En “Hardware” nos aseguraremos que el programador elegido sea el nuestro (JDM, Figura 1). En “Software” hay una serie de solapas y opciones (figura 2) que básicamente configuran los mensajes que recibiremos (o no) al utilizar el programa. En general, las opciones por defecto funcionarán correctamente para todos.

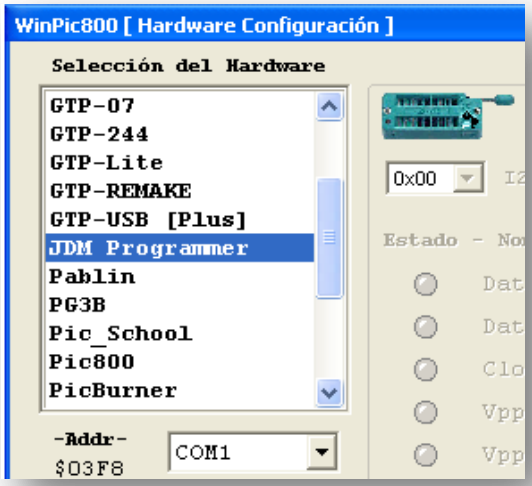



Figura 1


Figura 2

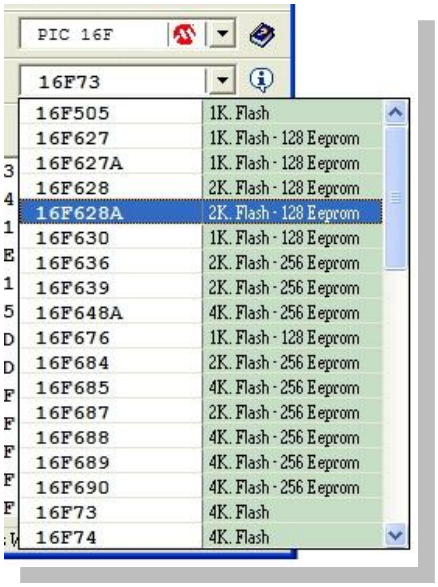



Figura 3

La figura 3 ilustra el paso siguiente: desde las listas que están a la derecha de la ventana principal del WinPIC800 seleccionamos la familia y modelo del Micro controlador que vamos a utilizar. Este debe coincidir con el modelo de PIC que se este utilizando, ya que

el programa que se genera está especialmente concebido para un modelo en particular. Como familia seleccionamos “PIC 16F” y como modelo “16F84”.

Una vez que hemos hecho esto, WinPIC800 “sabe” como deberá enviar los datos al programador. Otro punto a tener en cuenta en esta etapa del proceso es la posición que debe ocupar el PIC en el zócalo del programador. Debemos prestar atención a la posición del PIC en el quemador, éste presenta instrucciones grabadas en la placa acerca de la posición del PIC. Luego, debemos ir al menú “Archivo” --> “Abrir” y cargar el fichero HEX que generamos con el PIC SIMULATOR IDE.

El led del quemador estará encendido si todo esta correctamente instalado, por lo que podemos proceder a enviar el fichero. Para ello, presionamos el icono “Grabar Todo” que se ve en la figura 4, y en un par de segundos tenemos nuestro PIC grabado. El mensaje que veremos será el de la figura 5.

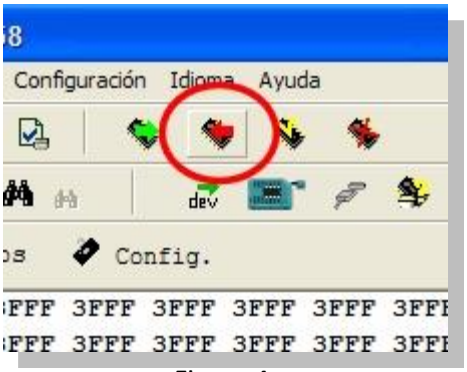


Figura 4

Figura 4

Para probar que todo funciona, tenemos que armar el circuito y alimentarlo con 5V de corriente continua.

**PROTEUS VSM**

## A.- Introducción.

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

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

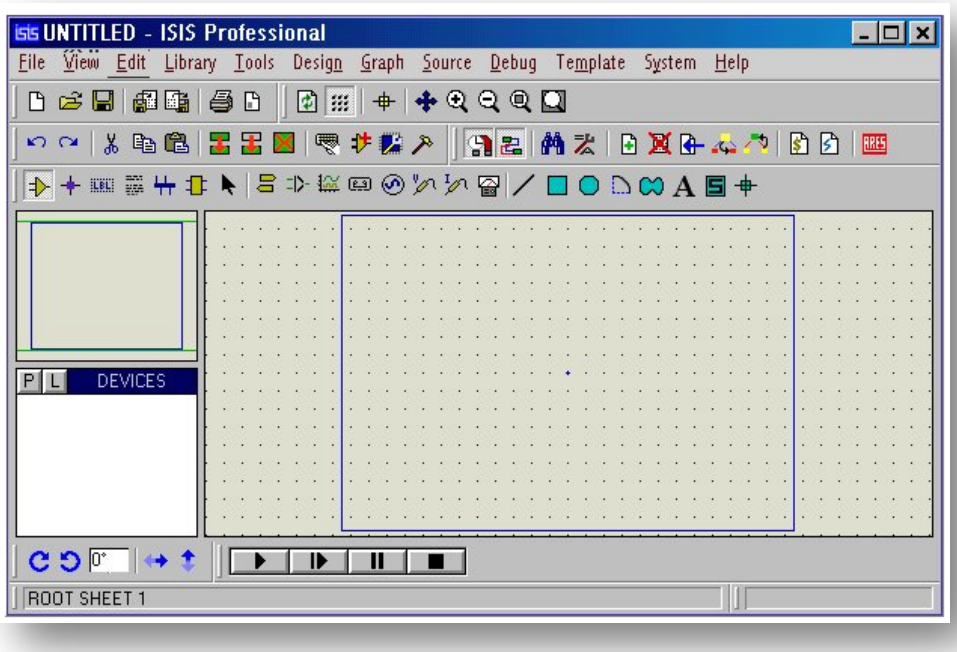
La simulación puede incluir instrumentos de medición y la inclusión de gráficas que representan las señales obtenidas en la simulación.

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

También tiene la capacidad de pasar el diseño a un programa integrado llamado ARES en el cual se puede llevar a cabo el desarrollo de placas de circuitos impresos.

### Procedimiento de Arranque del programa:

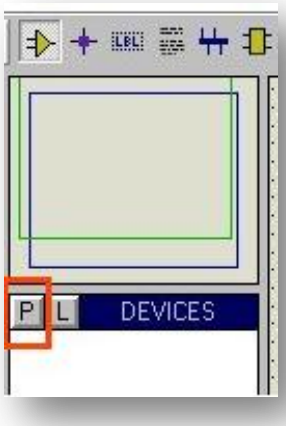
1.- Inicio -> Programas -> Proteus 6 Professional -> ISIS 6 Professional.

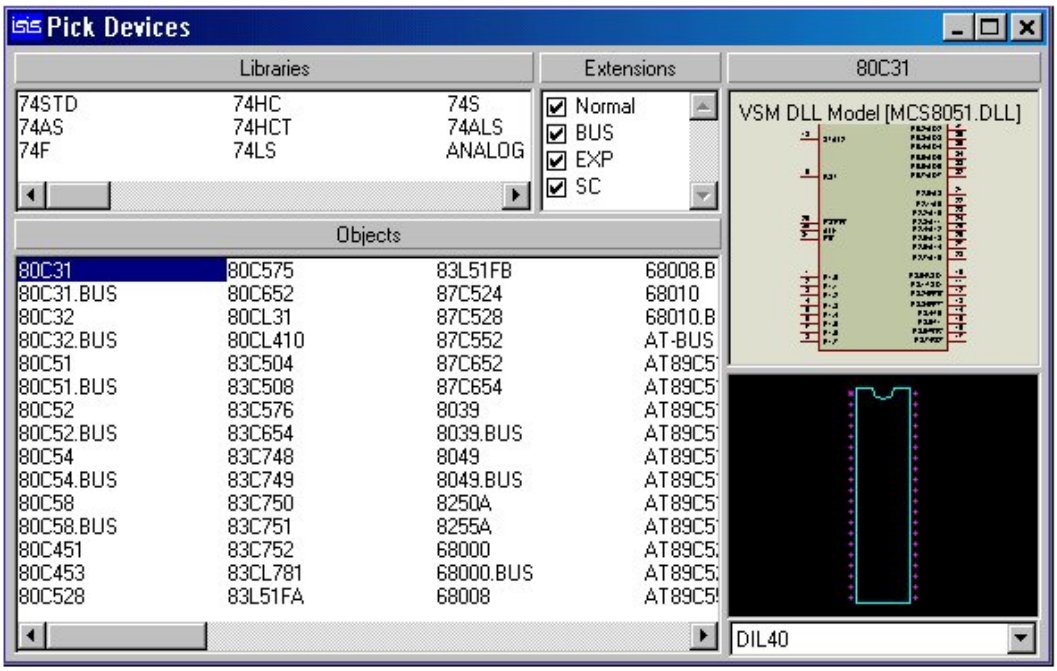



2.- La forma corta es dar doble clic en el icono del programa ubicado en el escritorio.

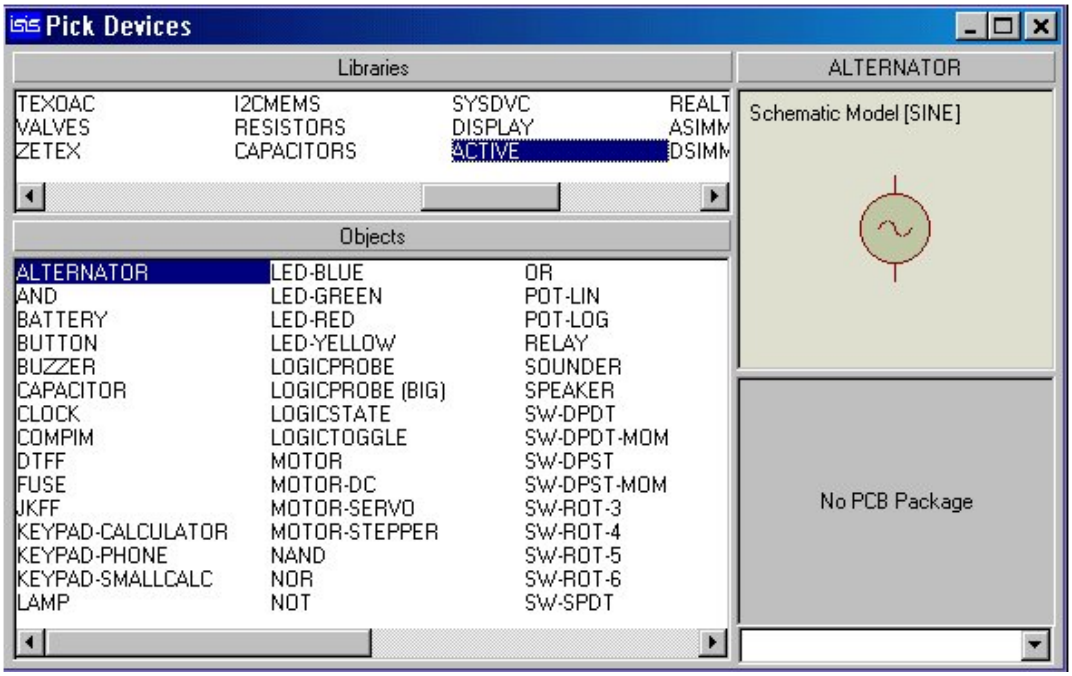

**B.- Circuito Básico # 1 (Desarrollo) - Alimentación de un foco de corriente alterna.**

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

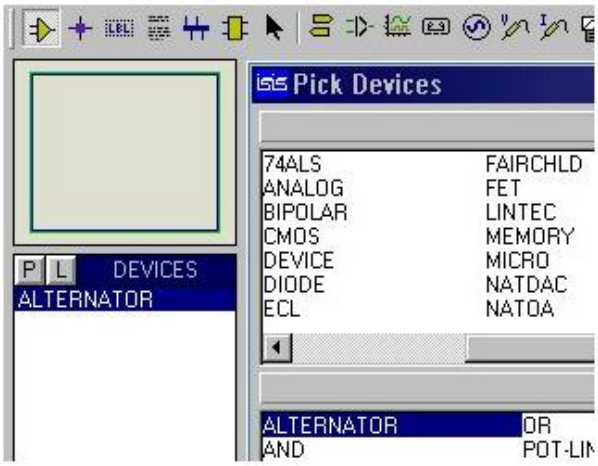

P
L
DEVICES


Pick Devices
Libraries
Extensions
80C31
74STD
74HC
74S
74AS
74HCT
74ALS
74F
74LS
ANALOG
Normal
BUS
EXP
SC
VSM DLL Model [MCS8051.DLL]
Objects
80C31
80C575
83L51FB
68008.B
80C31.BUS
80C652
87C524
68010
80C32
80CL31
87C528
68010.B
80C32.BUS
80CL410
87C552
AT-BUS
80C51
83C504
87C652
AT89C5
80C51.BUS
83C508
87C654
AT89C5
80C52
83C576
8039
AT89C5
80C52.BUS
83C654
8039.BUS
AT89C5
80C54
83C748
8049
AT89C5
80C54.BUS
83C749
8049.BUS
AT89C5
80C58
83C750
8250A
AT89C5
80C58.BUS
83C751
8255A
AT89C5
80C451
83C752
68000
AT89C5
80C453
83CL781
68000.BUS
AT89C5
80C528
83L51FA
68008
AT89C5
DIL40

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




3.- En la ventana **Objects** elegir el componente **ALTERNADOR** dando doble clic sobre el nombre.

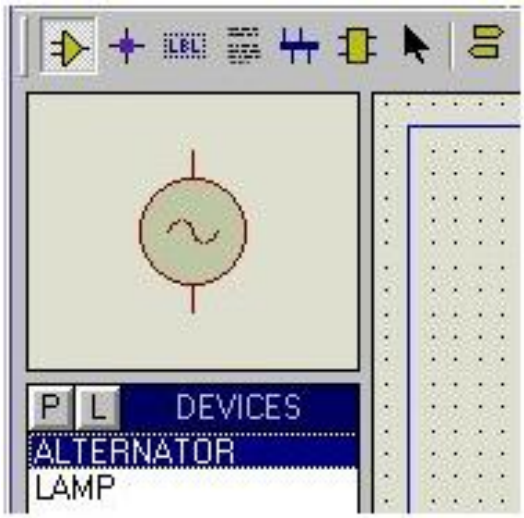



Se puede observar que en la ventana **DEVICES** aparece el nombre del componente elegido. Si es el único componente que se va a elegir se puede cerrar la forma Pick Devices, pero si es necesario más de uno, se puede continuar eligiendo los componentes necesarios para nuestro diseño.

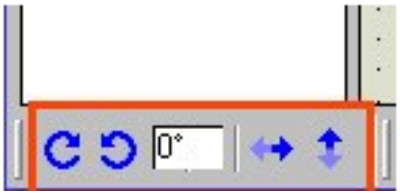
4.- En la misma librería **ACTIVE** dar doble clic sobre el componente **LAMP**.


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

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



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

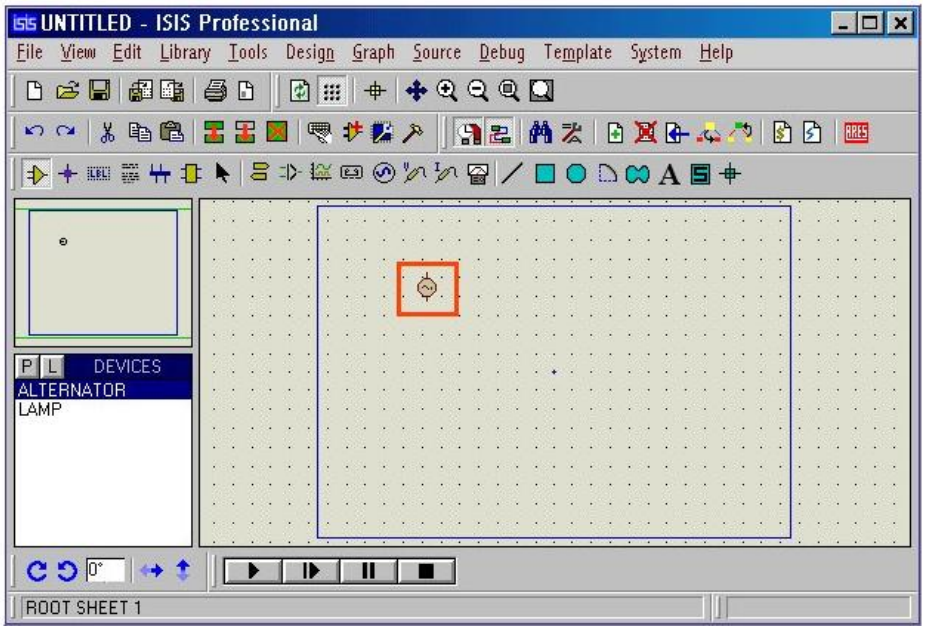



8.- Comenzando por la izquierda presionar cada uno de los botones de orientación.

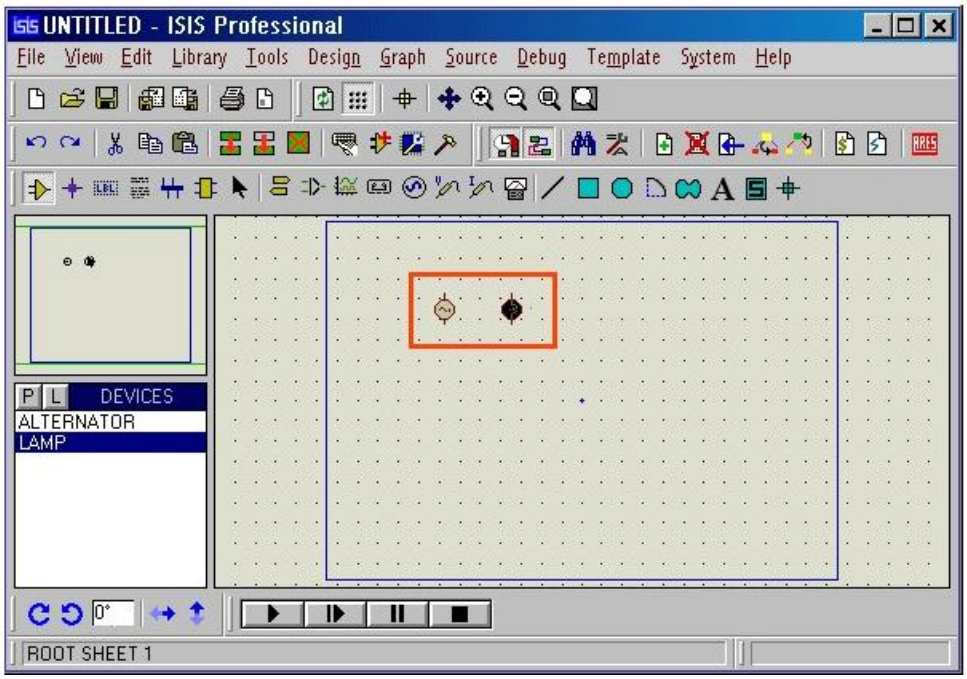
9.- En el cuadro de texto se puede introducir un ángulo pero sólo acepta valores de (0º, +/-90º, +/-180º, +/-270º), por lo que es mejor manejar la orientación por medio de los botones. Este mismo cuadro de texto muestra el ángulo actual obtenido al presionar los botones.

10.- Dejar el componente en la posición inicial.

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



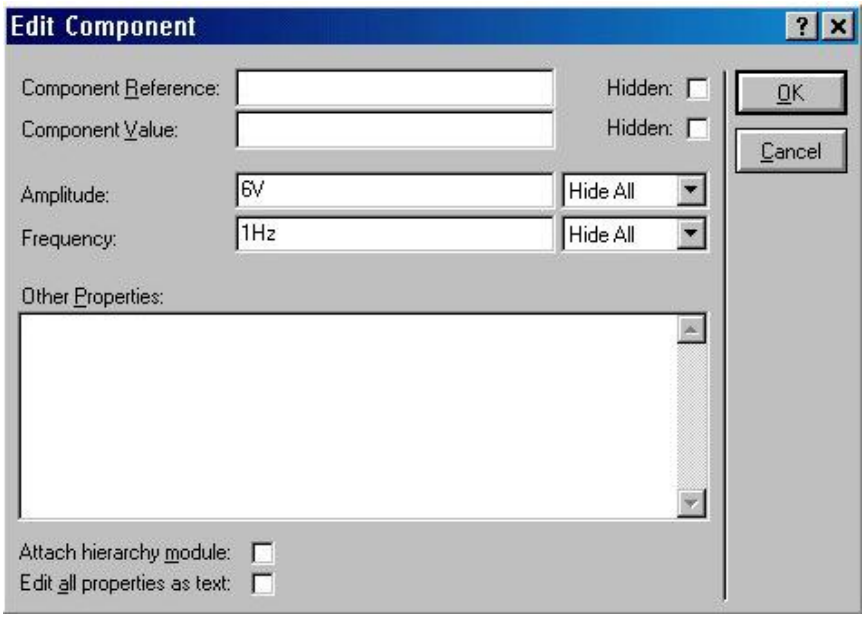

12.- Repetir el procedimiento anterior con el componente LAMP.



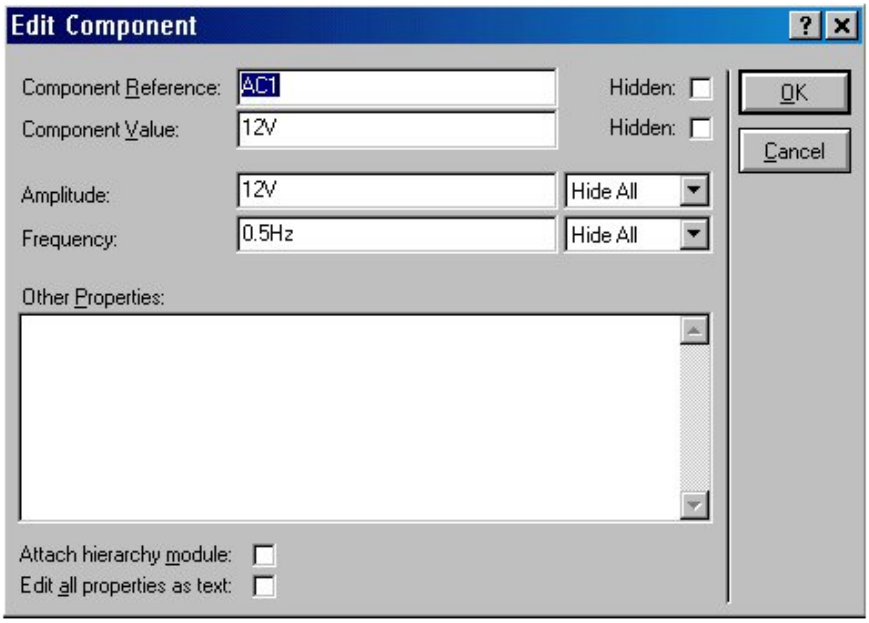
13.- Configurar los componentes de la siguiente manera.

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

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

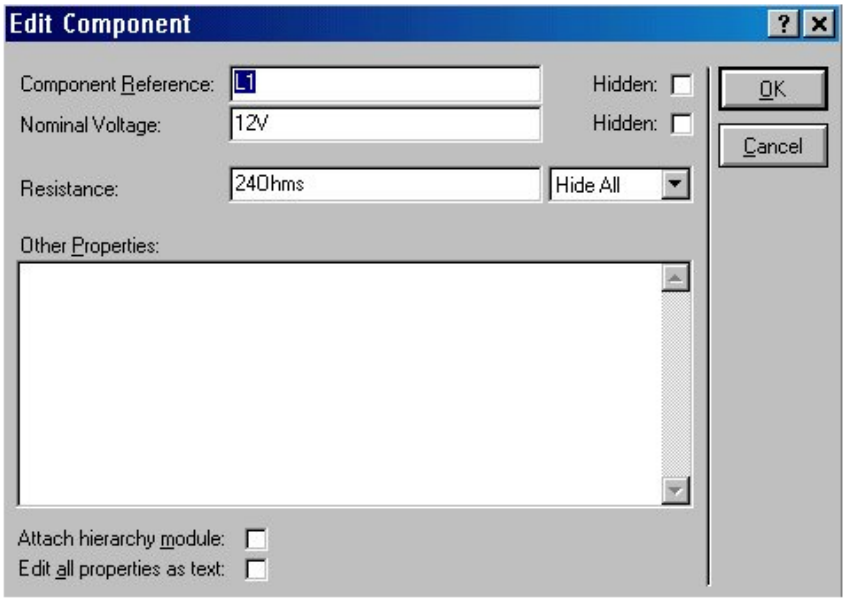



c.- Dar un nombre al componente en el campo Component Reference (AC1), Poner el valor del componente en el Component Value (12V), Modificar el valor de la amplitud a (12V) y la frecuencia a 0.5Hz.




d.- Presionar el botón OK.

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




14.- Realizar la conexión de los componentes de la siguiente forma:

a.- Colocar el puntero del mouse en el extremo superior del ALTERNADOR. Aparece una cruz en el extremo de la flecha.

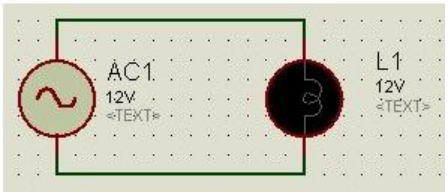
b.- Dar un clic para habilitar la conexión por medio de cable.

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

d.- Dar otro clic para realizar la conexión.

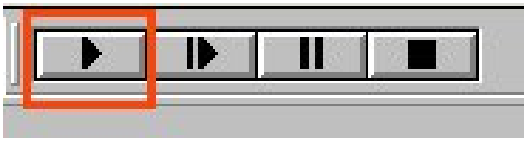
e.- Repetir los pasos anteriores para la pare inferior de los componentes.

**Resultado:**

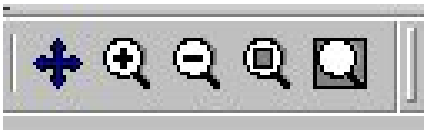



Este es el procedimiento estándar para conectar cualquier componente con el que se trabaje en el programa.

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



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



Comenzando de izquierda a derecha tenemos:

a.- Re-centrar la pantalla.

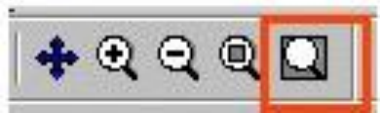
b.- Incrementar el acercamiento.

c.- Decrementar el acercamiento.

d.- Ver la hoja completa.

e.- Ver un área seleccionada

Usar la herramienta para Ver una área seleccionada dando un clic

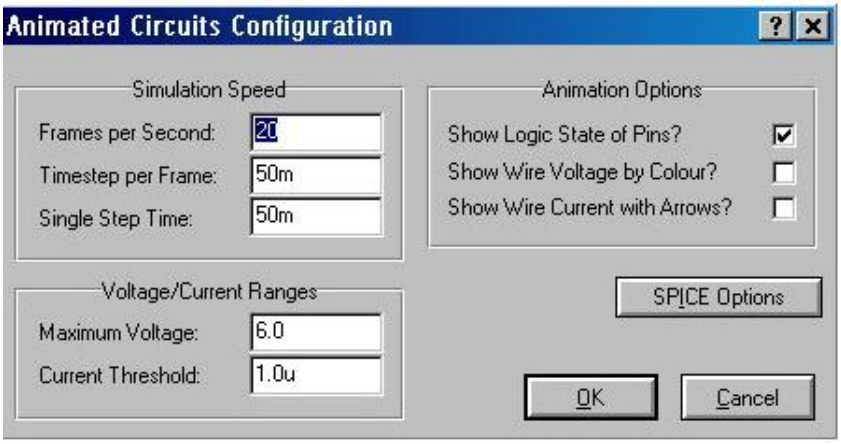


Usando el cursor modificado presionar el botón izquierdo en el extremo superior izquierdo del circuito armado y sin soltar el botón formar un rectángulo que contenga todo el circuito, por último soltar el botón.

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

17.- Habilitar los colores de voltaje y las flechas de corriente del circuito para completar la simulación.

a.- Ingresar al menú System y seleccionar **Set Animation Options...** para abrir la forma **Animated Circuits Configuration**.



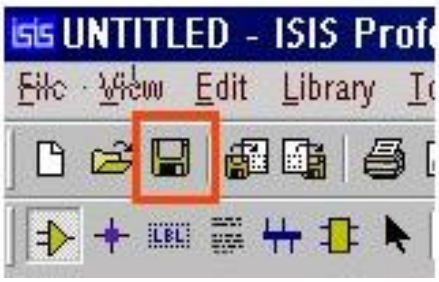

b.- Habilitar las casillas Show Wire Voltaje by Colour? y Show Wire Current with Arrows?.

c.- Presionar OK.

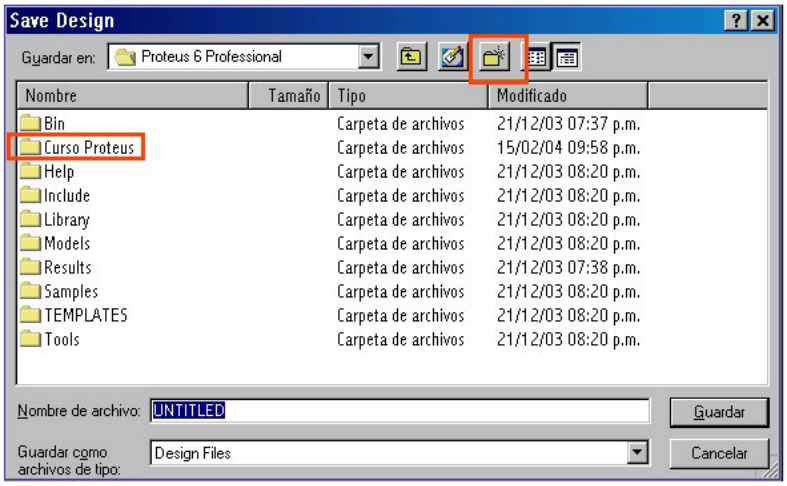
18.- Volver a simular el circuito y observar lo que ocurre.

19.- Guardar el circuito.

a.- Seleccionar la herramienta Save current design.

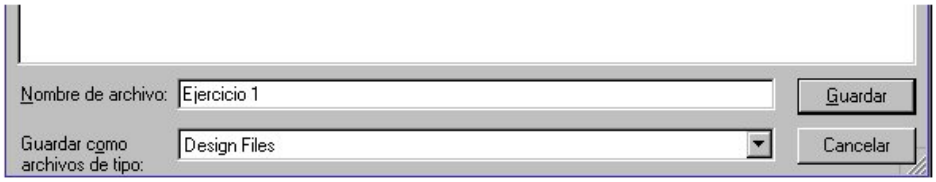



b.- Dar un clic en el botón crear una nueva carpeta y darle el nombre de Curso Proteus.




c.- Ingresar a la carpeta Proteus y crear dentro de ésta, una carpeta con el nombre Ejercicio 1.


d.- En el campo **Nombre de archivo** nombrar al archivo como Ejercicio 1.

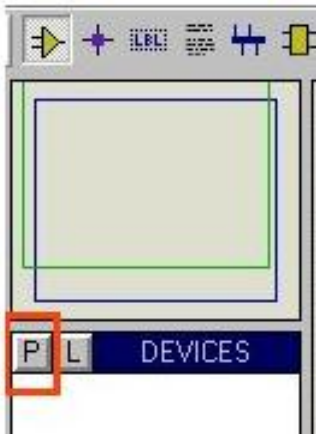



dar un clic en el botón guardar o presionar la tecla **ENTER.**

**C.- Circuito Básico #2 (Desarrollo) - Batería - Interruptor - Foco.**

1.- Dar un clic en **Create a New Design.**


2. Presionar el botón **Pick Devices.**

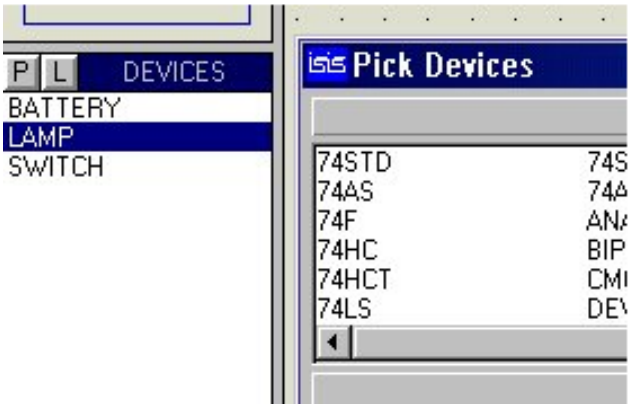



3.- De la ventana **Libraries** seleccionar ACTIVE y en la ventana Objects elegir los componentes:

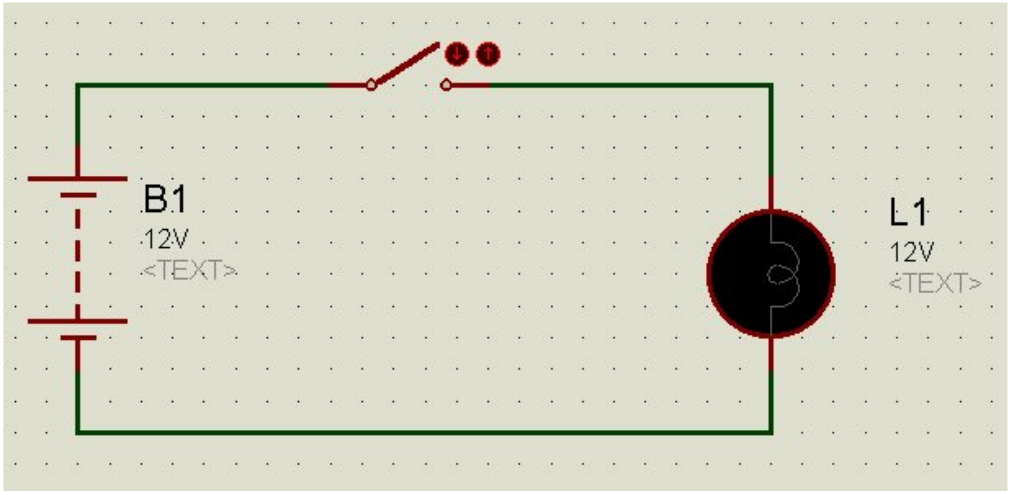
- BATTERY

- LAMP

- SWITCH




4.- Armar el siguiente circuito:




Modificar los valores de los componentes si es necesario.

5.- Ejecutar la simulación del circuito y probar el funcionamiento del interruptor. Dando clics con el botón izquierdo en las flechas arriba - abajo del interruptor.

6.- Guardar el archivo.

a.- Presionar Save current design.

b.- Salir de la carpeta del Ejercicio 1. (Subir un nivel)

c.- Dentro de la carpeta de Curso Proteus, crear una nueva carpeta con el nombre Ejercicio 2.

d.- Ingresar a la carpeta Ejercicio 2 y dentro de ella guardar el archivo con el nombre Ejercicio 2.

**D.- Circuito Básico #3 (Desarrollo) - Batería - Resistencia Variable - Foco.**

1.- Dar un clic en **Create a New Design.**

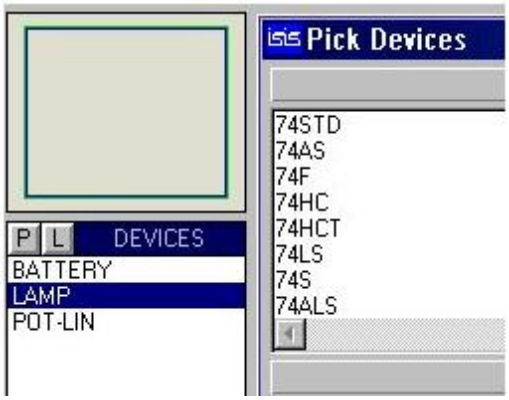
2. Presionar el botón **Pick Devices.**

3.- De la ventana **Libraries** seleccionar ACTIVE y en la ventana Objects elegir los componentes:

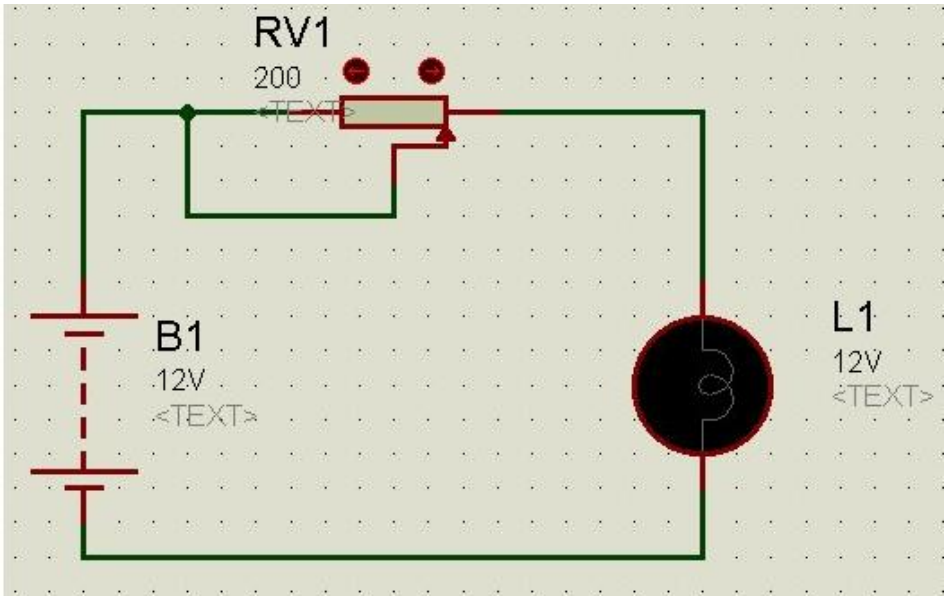
- BATTERY

- LAMP

- POT-LIN




4.- Armar el siguiente circuito:




5.- Ejecutar la simulación del circuito y probar el funcionamiento de la resistencia variable. Con el puntero del mouse dar clic en las flechas para aumentar o disminuir la resistencia.

6.- Guardar el archivo.

a.- Presionar Save current design.

b.- Salir de la carpeta del Ejercicio 2. (Subir un nivel)

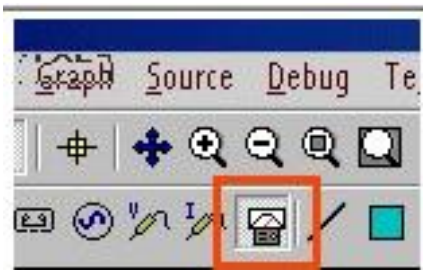
c.- Dentro de la carpeta de Curso Proteus, crear una nueva carpeta con el nombre Ejercicio 3.

d.- Ingresar a la carpeta Ejercicio 3 y dentro de ella guardar el archivo con el nombre Ejercicio 3.

**E.- Agregar instrumentos de medida a un circuito.**

Usando el circuito anterior hacer lo siguiente:

a.- De la barra superior de herramientas seleccionar Virtual Instruments.

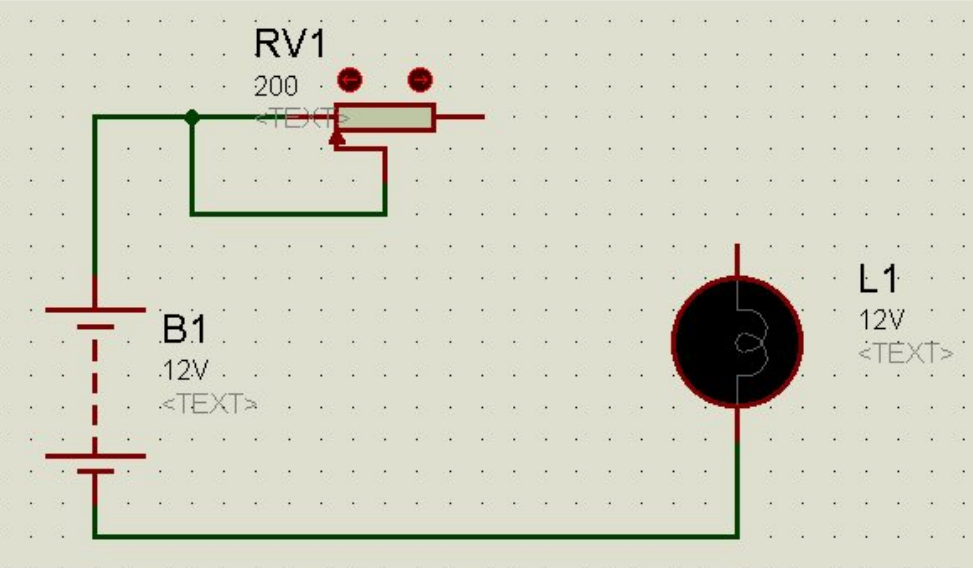


b.- Dar doble clic con el botón en el cable que une la resistencia variable y la lámpara para dejar espacio a un amperímetro. Si es necesario mover un poco la lámpara hacia la derecha.

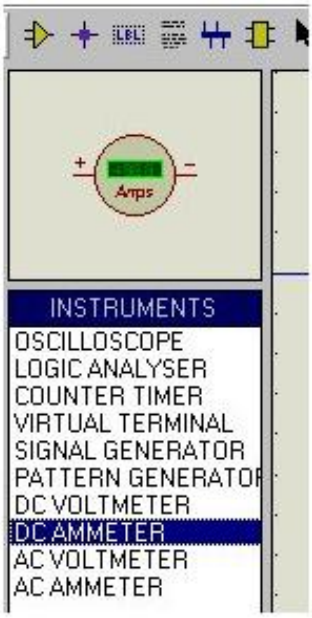
NOTA: Para mover un componente en el área de trabajo se realizan los siguientes pasos:

i.- Dar un clic con el botón derecho sobre el componente que se desea mover.

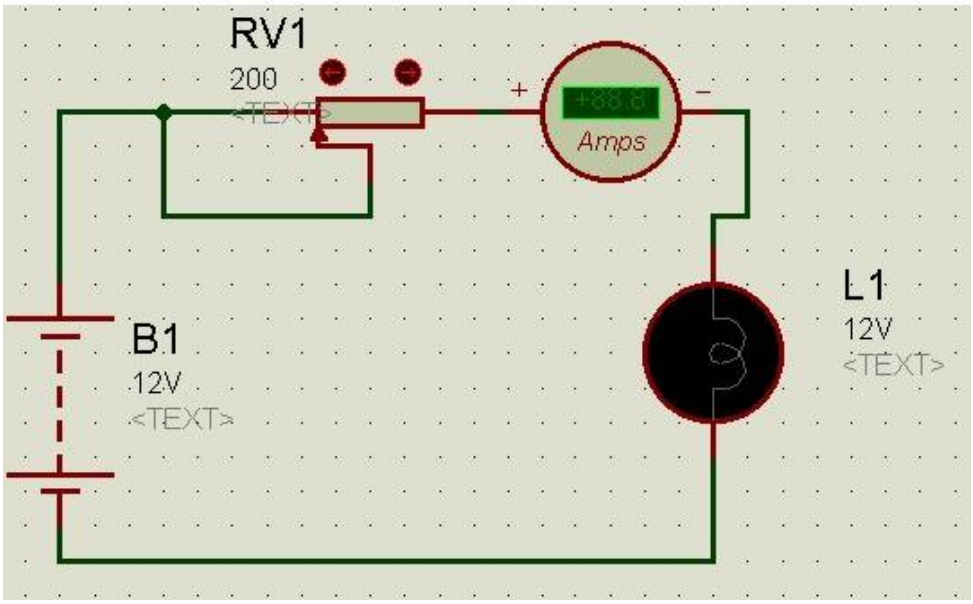
ii.- Presionar el botón izquierdo sobre el componente y sin soltar arrastrar el componente a la posición deseada.



c.- De la ventana **INSTRUMENTS** al lado izquierdo de la pantalla seleccionar con un clic izquierdo el instrumento **DC AMMETER**.

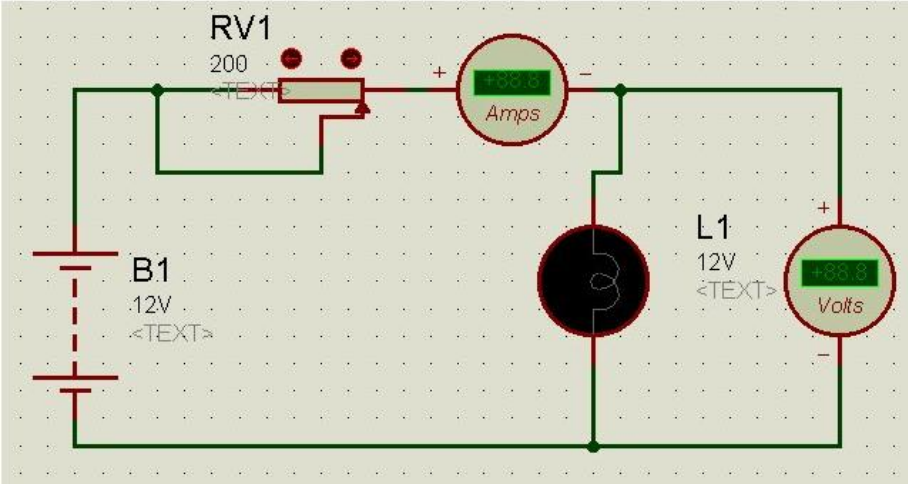



d.- Dar un clic en el área de trabajo entre la resistencia variable y la lámpara, y conectar los componentes para obtener el siguiente circuito.




e.- Seleccionar el instrumento **DC VOLTMETER** de la ventana **INSTRUMENTS**.

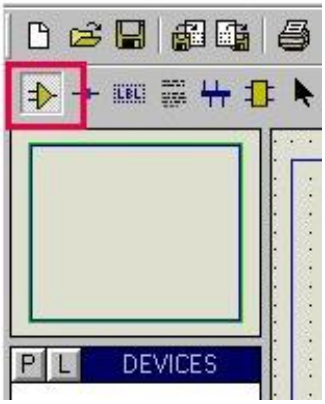
f.- Colocar en el área de trabajo de la misma forma que el instrumento anterior para obtener el siguiente circuito.



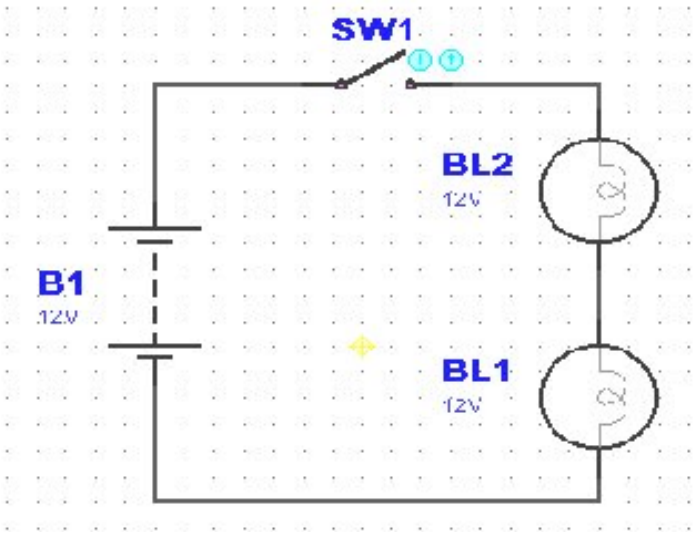

g.- Ejecutar la simulación del circuito.

h.- Guardar el circuito. Como ya había sido guardado el archivo ya no es necesario cambiar de directorio o nombrar el archivo.

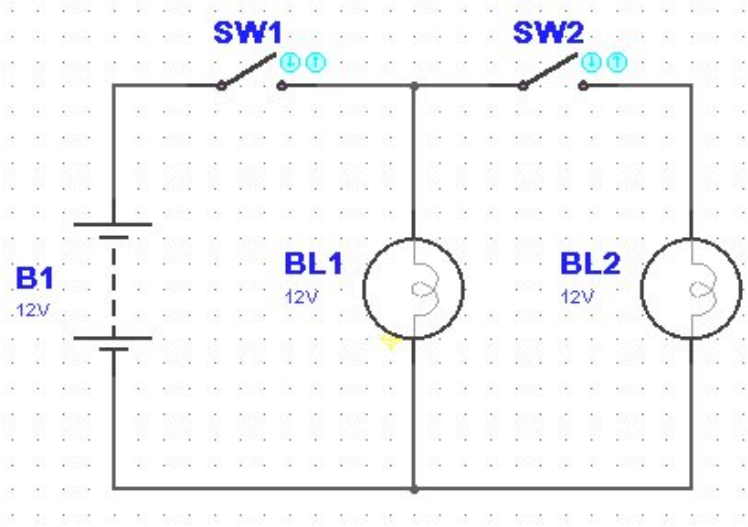
NOTA: para volver ha habilitar la ventana **DEVICES** presionar, en la barra de herramientas, **Component.**



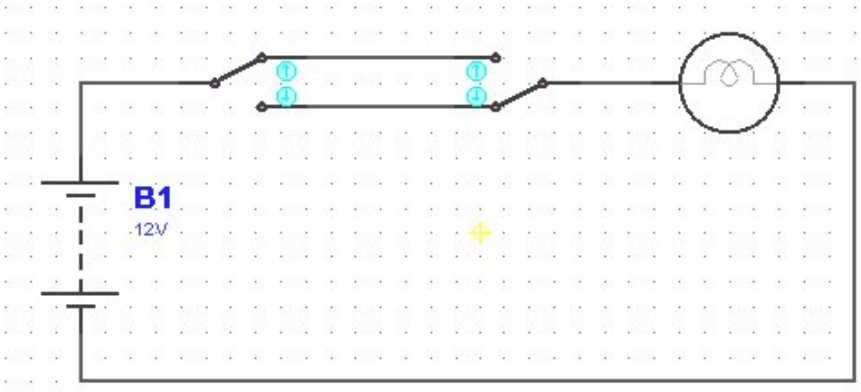

**F.- Circuito Básico #4 - Circuito Serie.**




**G.- Circuito Básico #5 - Circuito Paralelo.**




**H.- Circuito Básico #6 - Circuito con dos interruptores par control en dos direcciones.**




NOTA: Usar el componente **SW-SPDT** de la librería **ACTIVE**.