# UNIVERSIDAD POLITÉCNICA SALESIANA
## FACULTAD DE CIENCIAS TÉCNICAS

**CARRERA DE INGENIERÍA ELÉCTRICA**

**TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO**

**TEMA:**

# DISEÑO Y CONSTRUCCIÓN DE DOS MÓDULOS DIDÁCTICOS PARA EL LABORATORIO DE ELECTRÓNICA DE LA UNIVERSIDAD POLITÉCNICA SALESIANA

**Autores:**
**FAUSTO DANIEL PABÓN PLAZA**
**ÁNGEL OMAR TAPIA NOLIVOS**

**Director:**
**ING. ESTEBAN INGA ORTEGA**

**QUITO, Julio de 2007**

**DECLARATORIA DE AUTORÍA**

Nosotros, Fausto Daniel Pabón Plaza y Ángel Omar Tapia Nolivos autorizo a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de grado y su reproducción sin fines de lucro.

Además declaro que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

Quito, Julio 2007

| | |
|---|---|
| _____ | _____ |
| **Fausto Daniel Pabón Plaza** | **Ángel Omar Tapia Nolivos** |
| **CC: 1714000237** | **CC: 1716287378** |

**AUTORES**

**CERTIFICA**

Haber dirigido y revisado prolijamente cada uno de los capítulos técnicos y financieros del informe de la monografía, así como el funcionamiento del "Diseño y construcción de dos módulos didácticos para el laboratorio de electrónica de la Universidad Politécnica Salesiana" realizada por los Srs. Fausto Daniel Pabón Plaza y Ángel Omar Tapia Nolivos, previa a la obtención del título de Ingeniero Eléctrico en la Carrera de Ingeniería Eléctrica.

Por cumplir los requisitos autoriza su presentación.

Quito, Julio del 2007

_____

**Ing. Esteban Mauricio Inga Ortega**

**DIRECTOR**

## DEDICATORIA

**Daniel:**
Este trabajo está dedicado a mis queridos padres Fausto y Suly, a mi amada esposa Rocío, mis hermanos Cristina y Francisco, mi amigo de esfuerzo y trabajo Ángel, y aquellos que con su trabajo y consejo me apoyaron, guiaron y colaboraron para que esta meta sea posible realizarla, que Dios los bendiga, de corazón GRACIAS!

**Ángel:**
Todo el esfuerzo y la dedicación de este trabajo solamente fue posible gracias al apoyo de mi familia: Mis queridos padres y hermanos, mi amigo Daniel y mis amigos más allegados que siempre estuvieron junto a mí en cualquier circunstancia hasta llegar a culminar esta meta…..GRACIAS DE TODO CORAZON!

# INDICE GENERAL

## INDICE DE FIGURAS

## INDICE DE TABLAS

# RESUMEN

*Resumen*—"Este trabajo pretende entregar las pautas necesarias para desarrollar sistemas microncontrolados, comenzando por una introducción de los elementos electrónicos, sensores y transductores, posterior a esta etapa se desarrollaran los programas de aplicación basados en microntroladores PIC y el uso de su herramienta de programación, a esto se suma la etapa de simulación y finalmente la creación de los prototipos utilizando normas de diseño electrónico. Este completo trabajo además de obtener una guía didáctica, proveerá de diferentes prácticas requeridas para que estudiantes y docentes incursionen en el desarrollo de sistemas de adquisición de datos microcontrolados, usando diferentes medios o interfaz de comunicación inalámbrica y con aplicaciones reales de control sobre diferentes escenarios."

*Índice de Términos— Microncontroladores, Sistema de adquisición de Datos, Microelectrónica, Sensores, Transductores, Comunicaciones Inalámbricas, Microbótica*

# CAPITULO I

# MICROCONTROLADORES

## 1.1 Generalidades

**El Microcontrolador.**[1]

Es un circuito integrado programable que contiene todos los componentes de un computador.

Se emplea para controlar el funcionamiento de una tarea determinada y debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que le confiere la denominación de «controlador incrustado» (embedded controller).

El microcontrolador es un computador dedicado. En su memoria sólo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar, y todos los recursos complementarios disponibles tienen como única finalidad atender sus requerimientos. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada.

Un microcontrolador es un computador completo, aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado y se destina a gobernar una sola tarea.

El número de productos que funcionan en base a uno o varios microcontroladores aumenta de forma exponencial. No es aventurado pronosticar que en el siglo XXI habrá pocos elementos que carezcan de microcontrolador. En esta línea de prospección del futuro, la empresa Dataquest calcula que en cada hogar americano existirán varios centenares de microcontroladores en los comienzos del tercer milenio.

La industria Informática acapara gran parte de los microcontroladores que se fabrican. Casi todos los periféricos del computador, desde el ratón o el teclado hasta la impresora, son regulados por el programa de un microcontrolador.

Los electrodomésticos de línea blanca (lavadoras, hornos, lavavajillas, etc.) y de línea marrón (televisores, vídeos, aparatos musicales, etc.) incorporan numerosos microcontroladores. (Inga Ortega, 2002)

[1] ANGULO Usategui, Microcontroladores PIC, Mac Graw Hill, Edición 3, Pag 1-26

Igualmente, los sistemas de supervisión, vigilancia y alarma en los edificios utilizan estos chips. También se emplean para optimizar el rendimiento de ascensores, calefacción, aire acondicionado, alarmas de incendio, robo, etc.

Las comunicaciones y sus sistemas de transferencia de información utilizan profundamente estos pequeños computadores incorporándolos en los grandes automatismos y en los modernos teléfonos.

La instrumentación y la electro-medicina son dos campos idóneos para la implantación de estos circuitos integrados. Una importante industria consumidora de microcontroladores es la de automoción, que los aplica en el control de aspectos tan populares como la climatización, la seguridad y los frenos ABS.

Las comunicaciones y los productos de consumo general absorben más de la mitad de la producción de microcontroladores. El resto se distribuye entre el sector de la automoción, los computadores y la industria.

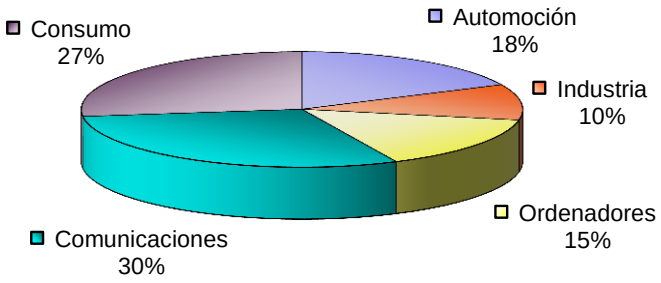



**Figura 1.1** *Distribución por sectores de aplicación.*

**Diferencia entre un Microprocesador y un Microcontrolador.**

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

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

Un microprocesador es un sistema abierto con el que puede construirse un computador con las características que se desee, acoplándole los módulos necesarios.

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

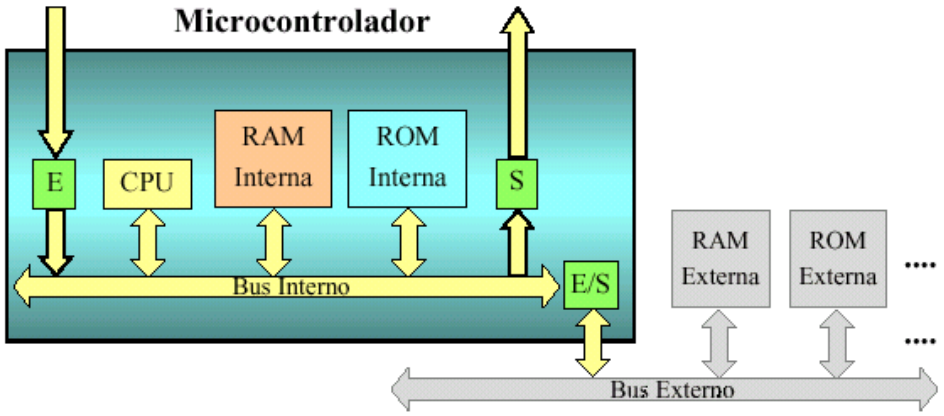



**Figura 1.2.** *Estructura de un sistema abierto basado en un microprocesador. La disponibilidad de los buses en el exterior- permite que se configure a la medida de la aplicación*.

## 1.2 Arquitectura básica.

Según la arquitectura interna de la memoria del microcontrolador se puede distinguir entre:

Microcontroladores con arquitectura Von Neumann.

Microcontroladores con arquitectura Harvard.

Inicialmente, todos los microcontroladores adoptaron la arquitectura clásica de Von Neumann. Actualmente, muchos microcontroladores utilizan esta arquitectura, pero poco a poco 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 por un sistema de buses único (direcciones, datos y control). Esta arquitectura presenta algunos problemas cuando se demanda rapidez.

La arquitectura Harvard dispone de dos memorias independientes; una, que contiene sólo instrucciones y otra, sólo datos. Ambas, disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias. Esta estructura no modifica nada desde el punto de vista del usuario y la velocidad de ejecución de los programas es impresionante.

### 1.2.1 Arquitectura del procesador o UCP.

Según la filosofía de la arquitectura del procesador se puede distinguir entre:
Microcontroladores CISC.
Microcontroladores RISC.
Microcontroladores SISC.
Un microcontrolador basado en la filosofía CISC (Computadores de Juego de Instrucciones Complejo) dispone 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 actúa como macros.
Tanto la industria de los computadores comerciales como los de los microcontroladores están decantándose hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecuta en un solo ciclo.
La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.
En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es específico, o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadores de Juego de Instrucciones Específico).
Arquitectura cerrada o abierta.
Entre los fabricantes de microcontroladores hay dos tendencias para resolver las demandas de los usuarios.
Los microcontroladores con arquitectura cerrada poseen una determinada UCP, cierta cantidad de memoria de datos, cierto tipo y capacidad de memoria de instrucciones, un número de E/S y un conjunto de recursos auxiliares muy concreto. El modelo no admite variaciones ni ampliaciones. La aplicación a la que se destina debe encontrar en su estructura todo lo que precisa, y en caso contrario, hay que desecharlo.

Los microcontroladores con arquitectura abierta se caracterizan porque, además de poseer una estructura interna determinada, emplean sus líneas de

E/S para sacar al exterior los buses de datos, direcciones y control, con lo que se posibilita la ampliación de la memoria y las E/S con circuitos integrados externos. Esta solución se asemeja a la que emplean los clásicos microprocesadores.
La línea que separa unos de otros es muy delgada, pero el concepto de microcontrolador se acerca posiblemente más a la arquitectura cerrada.

## 1.3 Las gamas de PIC.

La empresa Microchip Technology oriento su negocio a los PIC, las memorias EPROM paralelo y las EEPROM serie. Se comenzó rediseñando los PIC, que pasaron a fabricarse con tecnología CMOS, surgiendo la familia de gama baja PIC16C5X, considerada como la "clásica", la gama media PIC16CXXX, la gama alta PIC17CXXX y la gama mejorada PIC18CXXX y DsPIC150XXX.
Diferencia a sus modelos por la letra intermedia (C, F o CR:

C: Significa que la memoria de instrucciones es EEPROM.
F: Indica que la memoria de instrucciones es tipo Flash.
CR: La memoria de instrucciones es ROM y se graba en fábrica. Sólo se usa para grandes series.

### 1.3.1 Gama baja: PIC16C5X con instrucciones de 12 bits

Consiste en una serie de PIC de recursos limitados, pero con una de las mejores relaciones coste/prestaciones. Sus versiones están encapsuladas con 18 y 28 patitas y pueden alimentarse a partir de una tensión de 2,5 V lo que les hace ideales en las aplicaciones que funcionan con pilas. Tienen un repertorio de 33 instrucciones cuyo formato consta de 12 bits. No admiten ningún tipo de interrupción y la pila sólo dispone de dos niveles.

### 1.3.2 Gama media: PIC16CXXX con instrucciones de 14 bits

Es la gama más variada y completa de los PIC. Abarca modelos con encapsulado desde 18 patitas hasta 68, cubriendo varias opciones que integran abundantes periféricos. Dentro de esta gama se halla el «fabuloso PIC 16F84».
El repertorio de instrucciones es de 35 a 14 bits cada una y compatible con el de la gama baja. Sus distintos modelos contienen todos los recursos que se

precisan en las aplicaciones de los microcontroladores de 8 bits. También dispone de interrupciones y una Pila de 8 niveles que permite el anidamiento de subrutinas.

La gama media puede clasificarse en las siguientes subfamilias:

Gama media estándar (PIC16C55X);

Gama media con comparador analógico (PIC16C62X/64X/66X);

Gama media con módulo de captura (CCP), modulación de anchura de impulsos (PWM) y puerta serie (PIC16C6X);

Gama media con CAD de 8 bits (PIC16C7X);

Gama media con CAD de precisión (PIC14000);

Gama media con memoria Flash y EEPROM (PIC16F87X y PIC16X8X);

Gama media con driver LCD (PIC16C92X).

Encuadrado en la gama media también se halla la versión PIC14C000, que soporta el diseño de controladores inteligentes para cargadores de baterías, pilas pequeñas, fuentes de alimentación ininterrumpidas y cualquier sistema de adquisición y procesamiento de señales que requiera gestión de la energía de alimentación. Los PIC14C000 admiten cualquier tecnología de las baterías como Li Ion, NiMH, NiCd, Pb y Zinc.

### 1.3.3 Gama alta: PIC17CXXX con instrucciones de 16 bits

Se alcanzan las 58 instrucciones de 16 bits en el repertorio y sus modelos disponen de un sistema de gestión de interrupciones vectorizadas muy potente. También incluyen variados controladores de periféricos, puertas de comunicación serie y paralelo con elementos externos y un multiplicador hardware de gran velocidad.

Quizás la característica más destacable de los componentes de esta gama es su arquitectura abierta, que consiste en la posibilidad de ampliación del microcontrolador con elementos externos.

Para este fin, las patitas sacan al exterior las líneas de los buses de datos, direcciones y control, a las que se conectan memorias o controladores de periféricos. Esta filosofía de construcción del sistema es la que se empleaba en los microprocesadores y no suele ser una práctica habitual cuando se emplean microcontroladores.

### 1.3.4 Gama mejorada: PIC18C(F)XXX con instrucciones de 16 bits

En los inicios del tercer milenio de nuestra era Microchip presentó la gama mejorada de los microcontroladores PIC con la finalidad de soportar las aplicaciones avanzadas en las áreas de automoción, comunicaciones, ofimática y control industrial. Sus modelos destacaron por su alta velocidad (40 Mhz) y su gran rendimiento (10 MIPS a 10 Mhz).

Entre las aportaciones más representativas de esta serie de modelos que crece cada año, destacan.

a) Un espacio de direccionamiento para la memoria de programa que permite alcanzar los 2 MB, y 4 KB para la memoria de datos.

b) Inclusión de la tecnología FLASH para la memoria de código.

c) Potente juego de 77 instrucciones de 16 bits cada una. Permiten realizar una multiplicación 8 x 8 en un ciclo de instrucción, mover información entre las memorias y modificar el valor de un bit en un registro o en una línea de E/S.

d) Orientación a la programación en lenguaje C con la incorporación de compiladores muy eficientes para este lenguaje.

e) Nuevas herramientas para la emulación.

Inicialmente aparecieron cuatro modelos (PIC18C242/252/442/452) con 28 y 40 patitas que tenían hasta 16 KB de memoria de programa y hasta 1.536 bytes de RAM, ambas ampliables.

Podían funcionar a 40 MHz, con 16 causas de interrupción, 4 temporizadores, 2 módulos

CCP, Conversor A/D de 5 u 8 canales, y comunicación serie y paralelo. Luego aparecieron los PIC18FXXX que incorporaron la memoria FLASH para contener el código. Entre ellos destaca el modelo PIC18F720 con 128 KB de memoria FLASH y 3.840 bytes de RAM, estando encapsulado con 80 patitas. En la tabla de la Figura 2.7 se ofrecen las principales características de los primeros modelos de la gama mejorada

La memoria EEPROM, de igual forma que la FLASH, puede grabarse y borrarse eléctricamente, sin someterla a rayos ultravioleta como sucede con el borrado de las EPROM con ventana. Además, se puede realizar la grabación y el borrado en serie, lo cual posibilita la grabación de un programa, su depuración y su borrado tantas veces como se desee y manteniendo insertado el PIC en el zócalo de la aplicación. La memoria EEPROM admite hasta

1.000.000 de ciclos de escritura/borrado y almacena la información durante más de 40 años.
La memoria FLASH tiene un valor típico de 1.000 ciclos de escritura/borrado, pero aventaja técnicamente en varios aspectos a la EEPROM. Destacan en la gama media los PIC 16F87X con memoria FLASH de gran capacidad y numerosos recursos.
Cuando se prueban muchos programas en la fase de aprendizaje resulta muy práctico y económico.
Aplicaciones típicas de estos microcontroladores son el control de puertas de garaje, instrumentación, inmovilizadores de vehículos, tarjetas codificadas, pequeños sensores, etc. La grabación de los PIC16X8X en el propio circuito les hace recomendables para el almacenamiento de datos de calibración y para la modificación del programa al variar las condiciones

## 1.4 Prestaciones y recursos especiales.

Además de las clasificaciones anteriores, se podrían hacer otras dos clasificaciones más. Atendiendo a las prestaciones y atendiendo a los recursos especiales que pueden tener los microcontroladores.
Respecto a las prestaciones cabe destacar:

- Precio.
- Velocidad de ejecución de código.
- Eficiencia en la compactación de código.
- Inmunidad al ruido.

Indudablemente, el precio es uno de los factores decisivos a la hora de emplear uno u otro microcontrolador.
La velocidad de ejecución del código depende principalmente de la frecuencia de funcionamiento del microcontrolador, pero también influyen otras características como la arquitectura o el tipo de memoria empleada.
En lo que se refiere al número de palabras en la memoria que emplea cada microcontrolador en contener un programa, esta depende sobre todo de la arquitectura básica y de la longitud de la palabra de datos.
La inmunidad al ruido, así como otras características especiales como rangos amplios de temperaturas de funcionamiento, destacan sobre todo en microcontroladores destinados al uso militar.

Los recursos especiales más comunes que pueden poseer los microcontroladores son los siguientes:

- Temporizador y/o contador.
- Perro guardián o “Watchdog”.
- Protección ante el fallo de la alimentación.
- Estado de reposo o de bajo consumo.
- Conversor analógico-digital (CAD).
- Conversor digital-analógico (CDA).
- Comparador analógico.
- Modulador de anchura de impulsos o PWM.
- Puertas de entrada y salidas digitales.
- Puertas de comunicación (USART, USB, SCI, etc.)

Los temporizadores se emplean para controlar periodos de tiempo, actuando como temporizador, o para llevar la cuenta de acontecimientos que suceden en el exterior, actuando como contador.

El perro guardián consiste en un temporizador que cuando se desborda provoca un reset automáticamente en el microcontrolador, para así evitar que el sistema se quede “colgado”. Se debe diseñar el programa de tan modo que refresque o inicialice el perro guardián antes de que provoque el reset.

La protección ante el fallo de la alimentación consiste en un circuito que provoca un reset al microcontrolador cuando el voltaje de alimentación sea inferior a un voltaje mínimo. Mientras el voltaje de alimentación sea inferior al mínimo, el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

Son abundantes las situaciones reales en las que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, factor clave en los aparatos portátiles, los microcontrolador disponen de una instrucción especial que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos.

Los microcontrolador que incorporan un convertidor 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 los terminales del circuito integrado.

El conversor digital-analógico transforma los datos digitales obtenidos del procesamiento del microcontrolador, en su correspondiente señal analógica, que saca al exterior por unos terminales de la cápsula. Existen muchos efectores que trabajan con señales analógicas.

Algunos modelos de microcontrolador disponen internamente de un amplificador operacional que actúa como comparador analógico entre una señal de referencia fija y otra variable que se aplica por una de los terminales de la cápsula. La salida de comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.

El modulador de anchura de pulsos o PWM, es un circuito que proporciona en su salida impulsos de anchura variable, que se ofrecen al exterior a través de los terminales del encapsulado.

Todos los microcontrolador destinan algunos de sus terminales a soportar líneas de entrada y salida digitales. Por lo general, estas líneas se agrupan de ocho en ocho, formando así lo que se conoce como puertas.

Con el objeto de dotar al microcontrolador de la capacidad de comunicarse con otros dispositivos externos, otros buses de microcontrolador o microprocesadores, buses de sistemas o buses de redes y poder adaptarlos con otros elementos y con otras normas y protocolos, algunos microcontrolador disponen de puertas de comunicación. Destacan las conexiones serie UART y USART, las puertas paralelas, o el moderno bus serie USB desarrollado para los PC.

En la Tabla 1.1 se exponen a grandes rasgos las características más importantes de las gamas baja, media y alta.

| | Gama baja | Gama media | Gama alta |
|---|---|---|---|
| **Arquitectura** | Harvard / Cerrada | Harvard / Cerrada | Harvard / Abierta |
| **Procesador tipo** | 8 bit / RISC | 8 bit / RISC | 8 bit / RISC |
| **Segmentación** | Sí | Sí | Sí |
| **Máxima frecuencia** | 20 Mhz (1) | 20 Mhz (1) | 25 Mhz (1) |
| **Repertorio instrucciones** | 33 | 35 | 55 o 58 |
| **Longitud instrucciones** | 12 bits | 14 bits | 16 bits |
| **Tipo memoria ROM** | OTP, QTP, SQTP, EPROM | OTP, QTP, SQTP, EPROM, EEPROM, FLASH | OTP, QTP, SQTP, EPROM |
| **Tamaño ROM** | 512 – 2 k | 512 – 4 k | 2 k – 8 k |
| **Memoria datos SRAM** | 24 – 73 bytes | 31 – 192 bytes | 232 – 454 bytes |
| **Memoria datos EEPROM** | No | 64 bytes (1) | No |
| **Niveles de la pila** | 2 | 8 | 16 |
| **Encapsulado** | 18, 20 o 28 pines | 18, 28 o 40 pines | 40 o 44 pines |
| **Protección fallo $V_{DD}$** | No | Sí (1) | Sí (1) |
| **Modo de reposo** | Sí | Sí | Sí |
| **Interrupciones externas** | No | Sí | Sí |
| **Vectores de interrupción** | No | 1 | 4 |
| **Fuentes de interrupción** | 0 | Hasta 8 | 11 |
| **Perro guardián** | Sí | Sí | Sí |
| **Temporizadores** | 1 de 8 bits | De 1 a 3 de 8 | 4 de 8/16 |

| | | | |
|---|---|---|---|
| | | bits | bits |
| **Convertidor A/D** | No | Sí (1) | Sí (1) |
| **Módulo captura/comparación /PWM** | No | Sí (1) | Sí (1) |
| **Puerta serie** | No | Sí (1) | Sí (1) |
| **Puerta paralela esclava** | No | Sí (1) | Sí (1) |
| **Multiplicador hardware** | No | No | Sí |
| **Rango de tensión de alimentación** | 2 a 6,25 V (1) | 2 a 6 V (1) | 4,5 a 5,5 V (1) |
| **Precio aproximado** | 400 – 2.500 pts | 750 – 5.900 pts | 1.950 – 5.100 pts |

(1) Según modelo de la gama.

**Tabla 1.1** *Modelo de la gama*

## 1.5 Características de otros microcontroladores.

### 1.5.1 Altair.

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

Los microcontroladores Altair disponen de un microprocesador de 8 bits 100% compatible a nivel de código, 256 bytes de memoria interna, 128 registros especiales de función, puertos de entrada/salida de propósito general, 111 instrucciones y posibilidad de direccionar 128 Kbytes.

Existen distintos modelos dependiendo de la velocidad de ejecución, del número de E/S o de los periféricos de los que dispongan (DAC, ADC, Watchdog, PWM, etc.). La elección de un modelo u otro dependerá de las necesidades.

Como entrenador o sistema de iniciación existen varios modelos, entre los que destacan el Altair 32 Básico o bien el Altair 535A completo. Para proyectos avanzados o desarrollos profesionales, el Altair 537 A.

### 1.5.2 Intel.

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

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

Las mejoras incluyen más memoria, más puertos, convertidores analógico-digitales, más temporizadores, más fuentes de interrupción, Watchdog, y subsistemas de comunicación en red. Todos los microcontroladores de la familia usan el mismo conjunto de instrucciones, el MCS-51. Las características mejoradas son programadas y controladas por SFR adicionales.

En el Futuro está bien documentado y posee cientos de variantes e incontables herramientas de desarrollo.

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

### 1.5.3 Siemens.

El Siemens SAB80C515 es un miembro mejorado de la familia 8051 de microcontroladores. El 80C515 es de tecnología CMOS que típicamente reduce los requerimientos de energía. Las características que tiene frente al 8051 son más puertos, un versátil convertidor analógico-digital, un segundo temporizador optimizado, un Watchdog, y modos de ahorro de energía sofisticados.

El 80C515 es completamente compatible con el 8051. Esto es, usa el mismo conjunto de instrucciones del lenguaje ensamblador MCS-51. Las nuevas facilidades del chip son controladas y monitoreadas a través de SFR adicionales.

### 1.5.4 Motorola.

El 68hc11 de Motorola, es un potente microcontrolador de 8 bits en su bus de datos, 16 bits en su bus de direcciones, con un conjunto de instrucciones que es similar a los más antiguos miembros de la familia 68xx (6801, 6805, 6809). El 68hc11 dispone internamente de memoria de programa EEPROM u OTP, memoria de datos RAM, temporizadores, convertidor A/D de 8 bits y 8 canales, generador PWM, y canales de comunicación síncrona (SPI) y asíncrona (SCI). La corriente típica que maneja es menor que 10 mA.

La CPU tiene dos acumuladores de 8 bits (A y B) que pueden ser concatenados para suministrar un acumulador doble de 16 bits (D). Dos registros índices de 16 bits están presentes (X, Y) para suministrar indexamiento para cualquier lugar dentro del mapa de memoria. El tener dos registros índices significa que el 68hc11 es muy bueno para el procesamiento de datos. Aunque es un microcontrolador de 8 bits, el 68hc11 tiene algunas instrucciones de 16 bits. También dispone de un puntero de pila de 16 bits y de instrucciones para la manipulación de la pila.

En el futuro los modelos 683xx (Motorola), surgidos a partir de la popular familia 68k, a la que se incorporan algunos periféricos. Son microcontroladores de altísimas prestaciones.

## 1.6 Futuro de los microcontroladores Microchip

Los nuevos microcontroladores **PIC** Flash **PIC16F684**, **PIC16F688** y **PIC12F683**. Estos nuevos microcontroladores de 8 y 14 pines ofrecen al diseñador hasta 7.168 bytes de memoria de programa Flash PEEC propietaria de Microchip con un millón de ciclos de lectura escritura, memoria **EEPROM** interna y distintas opciones con respecto a los periféricos, todo ello con tecnología **nanoWatt®** integrada. Estos nuevos dispositivos son adecuados para un gran rango de aplicaciones incluyendo electrodomésticos, sensores, equipos portátiles alimentados por baterías, funciones de control de propósito general, etc.

Con los microcontroladores **PIC16F684**, **PIC16F688** y **PIC12F683** los clientes de Microchip tienen muy fácil la migración desde los clásicos dispositivos de 8 y 14 pines a estos microcontroladores con nuevas tecnologías. Además estos nuevos dispositivos ofrecen la familiar arquitectura media de 14bits con características estándares que incluyen un rango de tensión desde 2 hasta 5,5 voltios, memoria **EEPROM**, tecnología **nanoWatt** con consumo en reposo de nanoamperios, oscilador interno calibrado, modos de bajo consumo sin pérdida de prestaciones del sistema, etc. Los periféricos analógicos de estos tres nuevos **PIC** incluyen 8 canales de conversión A/D de 10 bit de resolución para medir fácilmente señales analógicas así como 2 comparadores de propósito general para el control de entradas/salidas analógicas.

El **PIC12F683** incluye un módulo **PWM/captura/comparador**, el **PIC16F688** se caracterizan por tener un módulo **EUSART** que soportará comunicaciones estándar **RS232/485** así como el protocolo **LIN**, tan popular en automoción. El **PIC16F684** tiene un módulo **PWM/captura/comparador** mejorado con retraso programable de banda muerta, más de cuatro salidas y modo bajo consumo, todo ello para mejorar la capacidad de control en aplicaciones como control de pequeños motores y fuentes de alimentación.

Los tres nuevos microcontroladores están disponibles en pequeños encapsulados **SOIC** de 8 pines, **TSSOP** de 14 pines y están soportados por distintas herramientas de desarrollo como son el entorno de desarrollo **MPLAB® IDE** con su versión más actual, el depurador en circuito **MPLAB ICD2** o el **PICkit 1 Flash Starter Kit.**

## 1.7 Análisis comparativo de prestaciones.

La arquitectura Harvard y la técnica de segmentación, son los principales recursos en los que se apoya el elevado rendimiento de los microcontroladores PIC, mejorando dos características que son esenciales, como la velocidad de ejecución y la eficiencia en la compactación del código.

A continuación se proporciona una comparación entre los modelos PIC16C5X a 20 Mhz, frente a los de otros importantes fabricantes.

National COP800 a 20 MHz.

SGS-Thomson ST62 a 8 MHz.

Motorola MC68HC05 a 4,2 Mhz.

Zilog Z86CXX a 12 Mhz.
Intel 8048/8049 a 11 Mhz.

Resultados de la comparativa respecto al tamaño promedio que ocupa el código de los programas usados en la prueba para los diversos modelos de microcontroladores usados.

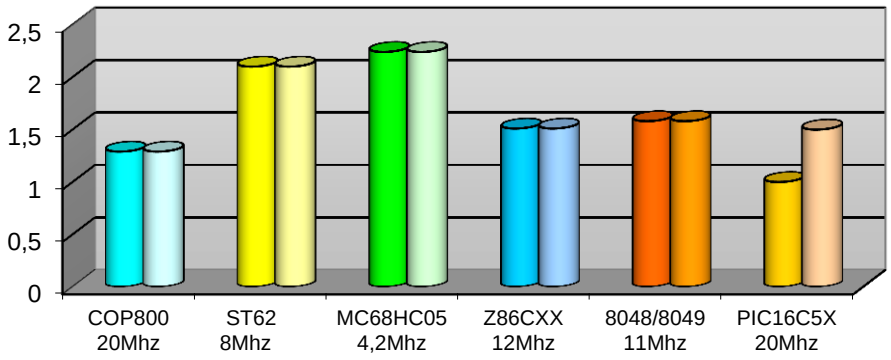



**Figura 1.3.** *Tamaño relativo del código.*

En lo que se refiere al número de palabras en la memoria de instrucciones que emplea cada microcontrolador en contener cada programa de prueba, hay que precisar que la longitud de las palabras que contienen código en los PIC16C5X es de 12 bits por tener una memoria de instrucciones independiente, frente a 8 bits del resto de modelos. La gráfica muestra esta situación representando en color más claro, el código equivalente en el caso de considerar la memoria de programa de 8 bits en todos los modelos. Resultados de la comparativa respecto a la velocidad promedio con la que ejecutan los diversos microcontroladores el conjunto de programas de prueba.

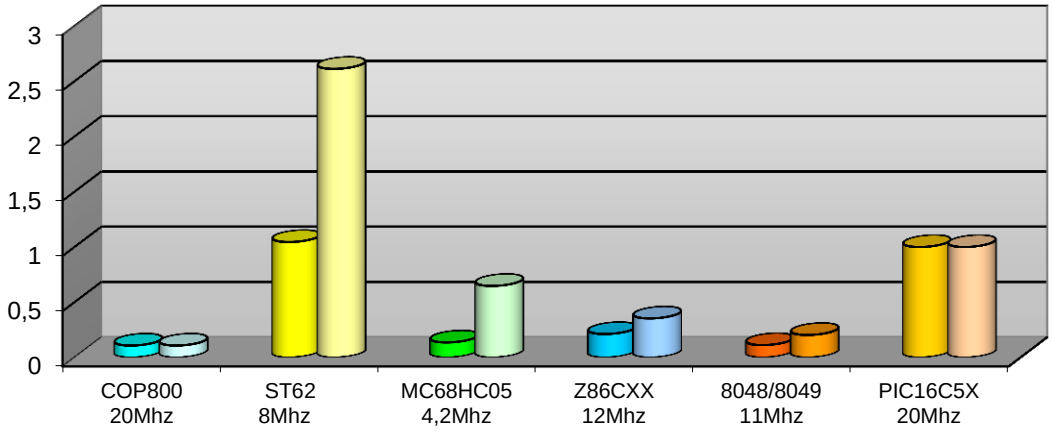



**Figura. 1.4.** *Velocidad promedio de ejecución.*

Para ser más justos en la comparativa, también se han representado los resultados, en color más claro, teniendo en cuenta la velocidad de reloj, es decir, representando la velocidad equivalente en el caso de que todos los modelos funcionen a 20 Mhz.

El estudio se ha realizado tomando como base un conjunto de programas de prueba y midiendo el tiempo promedio que tardan en ser ejecutados por los diversos microcontroladores comparados, así como el espacio de código que ocupan en la memoria de instrucciones. Los programas seleccionados para la prueba son muy sencillos pero muy representativos de las acciones típicas que llevan a cabo las aplicaciones que utilizan microcontroladores. Son los siguientes:

Empaquetamiento de dos dígitos BCD.

Control de un bucle que decrementa un contador hasta cero.

Transmisión síncrona por desplazamiento serie.

Temporizador software de 10 ms.

Exploración de un bit y salto si vale 1.

El Pic De 16 Bits De La Familia Microchip Dspic 30f2xxx, 30f3xxx Y 30f6xxx[2]

Microchip lanzó recientemente una nueva familia de microcontroladores llamada dsPIC. Esta línea viene para atender un mercado donde el procesamiento de un sistema se torne algo fundamental para la realización de

[2] Texto tomado y traducido al español de **http://www.cerne-tec.com.br**

un proyecto. De entre diversas ventajas de esta familia frente a las familias PIC16 y PIC18 podríamos citar las siguientes:

- MCU de 16 bits
- Poder de procesamiento de un DSP, utilizando una arquitectura de instrucciones dedicadas;
- La idea de Microchip es desarrollar herramientas de apoyo de mucha base matemática. Un ejemplo es el software para el proyecto de filtros que ya generan al código C o ensamblador.
- El precio debe ser equivalente al de los microcontroladores del familiar 18;
- La tecnología de 0,4u;
- No posee la tecnología NanoWatt;
- 84 instrucciones;
- Se ejecutan 86% de las instrucciones en 1 ciclo de la máquina (algunas de las instrucciones como la división especial en 18 ciclos);
- Velocidad de procesamiento: 30MIPS máximo, nominal,: 20 MIPS;
- Alimentación de 2,5 a 5,0V;
- Flexibilidad del reloj que posee PLL, divisor de frecuencia, RC interno y posibilidad de la oscilación para cristal del timer1;
- Más grande velocidad de reacción ante el modo sleep (Wake-up), cuando el oscilador es RC. Ya con cristal tarda algunos ciclos de la máquina para despertarse, lo que puede tardar algunos ms.
- A/D de 10 bits a 500ks o 12 bits a 100ks. Sampling simultáneo de 4 canales y la conversión individual a 500ks lo que reduce la frecuencia por 125ks;
- Conversión automático que agrupa dos canales y produce un buffer de 16 palabras;
- A/D puede ponerse en el sincronismo con el PWM;
- Memoria RAM de 32kx16bits;
- EEPROM 1k a 4k de 16 bits;
- Memoria de programa de 64kb hasta 4Mb de memoria externa con un barrido de programa de 24 bits;

- 16 registradores W de 16 bits. Algunos se operan para el funcionamiento de DSP. W15 = el indicador de la pila;
- Contador de Programa = 23 bits;
- La pila esta en la RAM, en otros términos, que la limitación de la pila depende de la RAM;
- Multiplicación de 17 x 17 bits;
- 2 acumuladores de 40 bits;
- 2 Registros de Estado (STATUS) (DSP STATUS y MCU STATUS);
- Buffer Circular (Filtros digitales)
- Bit de reversa (FFT). Para el TMS320LF240 tarda 74% más que el dsPIC para calcular un FFT;
- Instrucciones DO y REPEAT;

- WDT de 2ms a 16s con 1% de precisión;
- Protección contra falla en el cristal, (el fracaso del reloj Principal, el dsPIC para el
- el oscilador interno);
- Cada interrupción tiene su vector de interrupción, no siendo necesario verificar los bits. Las interrupciones de Trampa (Trap), (fallas en el oscilador, ejecución de programa en área inválida para ICD).
- 50 niveles de interrupciones, con 7 niveles de prioridad;
- La instrucción DISI apaga las interrupciones para N ciclos de máquina para escribir en la EEPROM interna, por ejemplo;
- Protección de la memoria. El componente puede auto-programarse, si en el dsPIC se intentar grabar en una área de memoria protegido, una interrupción,
- de Trampa (Trap) se genera;
- Bootloader;
- La grabación del componente es hecho en los bloques de 16 bytes y tarda 2ms. El tiempo total de grabar está alrededor de 5 a 15 segundos;
- Timers de 16 bits, pero ellos pueden ponerse en forma de cascada para generar cronómetro de 32 bits;
- Los nuevos recursos en el módulo capturan para generar la interrupción a cada 4 la captura y otro;
- La entrada para lectura de 3 Encoders de quadratura, directo de motores, para el mando de la posición y velocidad.
- 8 PWM simple y 4 complementa con banda muerta. Pueden ser seleccionados PWM tipo Edge (todos los pwm pueden funcionar al mismo tiempo), evento del chamusco (usó para la corrección de factor de potencia) o centro (ningún pwm se levanta al mismo tiempo, ideal por codificar puentes);
- Hasta 2 UART con 4 bytes de pila cada uno;
- I2C Multi-maestro;
- Lectura de CODEC por hardware;
- 2 CAN
- Dividido en 3 familias: para motores: (dsPIC30F2010/3011/6010) para sensores de uso general (dsPIC30F6012)
- Compilador C30 de Microchip (US895).
- No uso de MPLAB, posee recurso “Virtual Initializer”, una opción para configuración del componente en diagrama de bloques.
- Software de Momentum Data System para proyecto de filtros digitales.
- Instrucción PWRSAV, en modo IDLE el cpu detiene pero no cronometra. En el modo SLEEP es posible despertarse con el oscilador interno que es mucho más rápido que el externo.
- Interruptor de “Detección de bajo voltaje”.

Los dsPICs están disponibles en tres familias, mientras siendo estos las familias de mando de motores, sensor y uso general.

Con este Microcontrolador nosotros podemos probar prácticamente todos los recursos del dsPIC30F3012 que tiene el Microchip presentados debajo:

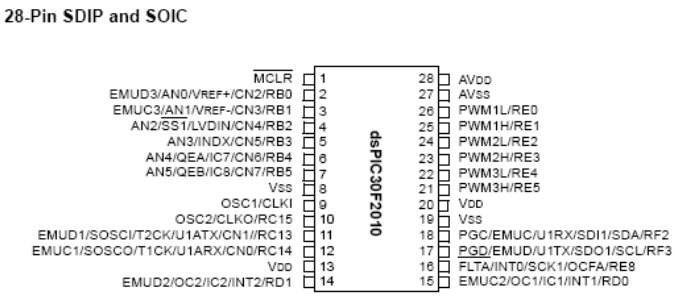



**Figura 1.5** *Diagrama de Pines de un Microcontrolador dsPIC*

# CAPITULO II

# SENSORES

**Definición.-**[3]

Un elemento eléctrico/mecánico/químico capaz de convertir una característica del entorno en una medida cuantitativa.

Cada sensor se basa en un principio de transducción: conversión de la energía de una forma a otra.

Sentidos y Órganos Humanos:

Visión: ojos (óptico, luz)

Oído: oídos (acústico, sonido)

Tacto: piel (mecánico, calor, textura, ...)

Olor: nariz (química, vapores)

Gusto: lengua (química, líquidos)

Extensión del rango de percepción humano:

Visión fuera del espectro visible

Cámara de Infrarrojos, visión nocturna

Visión activa

Medición con radar, sonar, láser, ...

Sonidos fuera del rango de los 20 Hz – 20 kHz

[3] Texto de Ramón Pallas Areny, SENSORES Y ACONDICIONADORES, Pag.17-49

Medición con ultrasonidos

Análisis químicos que sustituyan el gusto o el olfato

Nariz electrónica

Radiación: α, β, γ-rays, neutrones, etc.

## 2.1 Terminología

Los siguientes términos se emplean para definir el funcionamiento de los transductores y, con frecuencia, el de los sistemas de medición como un todo.

Rango y margen. El rango de un transductor define los límites entre los cuales puede variar la entrada. El margen es el valor máximo de la entrada menos el valor mínimo. Por ejemplo, una celda de carga utilizada para medir fuerzas, podría tener un rango de 0 a 50kN y un margen de 50 kN.

Error. El error es la diferencia entre el resultado de una medición y el valor verdadero de la cantidad que se mide.

Error = valor medido - valor real

Por ejemplo, si un sistema de medición marca un valor de temperatura de 25°C, cuando el valor real de la temperatura es 24 °C, el error es +1°C. Si la temperatura real fuera 26°C, entonces el error sería -1 °C. El sensor puede producir un cambio en la resistencia de 10.2 Ω, cuando el cambio verdadero debió ser de 10.5 Ω. El error es de -0.3 Ω.

Exactitud. La exactitud es el grado hasta el cual un valor producido por un sistema de medición podría estar equivocado. Es por lo tanto, igual a la suma de todos los errores posibles más el error en la exactitud de la calibración del transductor. Por ejemplo, si la exactitud de un instrumento para medir temperatura se especifica como un valor de ±2 °C, la lectura en el instrumento estará entre +2 y -2°C del valor real. Es común expresar la exactitud como un porcentaje de la salida a rango total, o como una desviación a escala total. El término desviación a escala total se originó cuando las salidas de los sistemas de medición se presentaban casi siempre en una escala circular o lineal. Por ejemplo, la especificación de exactitud de un sensor sería de ±5%

de la salida a escala total; y si el rango del sensor fuera de O a 200°C, la lectura sería entre + 10 Y-10°C de la lectura real.

Sensibilidad. La sensibilidad es la relación que indica qué tanta salida se obtiene por unidad de entrada, es decir, salida/entrada. Por ejemplo, un termómetro de resistencia puede tener una sensibilidad de 0.5 Ω/ºC. Es frecuente que este término también se utilice para indicar la sensibilidad de otras entradas, además de la que se mide, por ejemplo, a cambios ambientales. Entonces, puede haber sensibilidad del transductor a los cambios en la temperatura ambiente, o quizás a las fluctuaciones en el suministro de voltaje de la línea comercial. Puede decirse que un transductor para medir tiene sensibilidad de ±0.1 % °C de la lectura por °C de cambio en la temperatura.

Error por histéresis. Los transductores pueden producir distintas salidas de la misma magnitud que se mide, si dicha magnitud se obtuvo mediante un incremento o una reducción continuos. A este efecto se le conoce como histéresis. La figura 2.1 muestra una salida de este tipo, donde el error por histéresis es la diferencia máxima en la salida obtenida a partir de valores de incremento y de decremento.

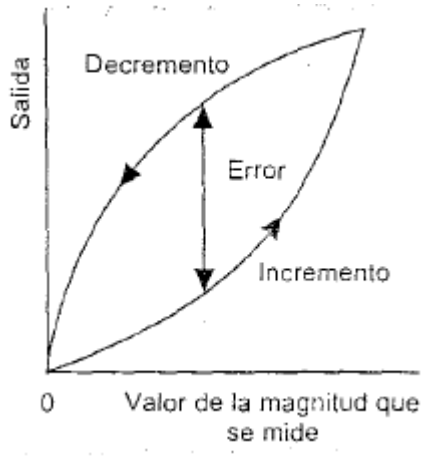



**Figura 2.1** *Histéresis*

Error por no linealidad. Para muchos transductores, se supone que en un rango de funcionamiento la relación entre la entrada y la salida es lineal, es decir, la gráfica de la salida respecto a la entrada produce una línea recta. Sin embargo, son pocos los transductores en los que la relación anterior es realmente una línea recta; por ello, al suponer la existencia de esta linealidad

se producen errores. Este error se define como la desviación máxima respecto a la línea recta correspondiente. Para expresar numéricamente el error por no linealidad se utilizan varios métodos. Las diferencias ocurren al determinar la relación de la línea recta que especifica el error. Uno de estos métodos consiste en dibujar la recta que une los valores de la salida con los puntos extremos del rango; otro es determinar la recta con el método de mínimos cuadrados, a fin de calcular qué línea se adapta mejor considerando que todos los valores tienen la misma probabilidad de error; otro más es encontrar la línea recta con el método de mínimos cuadrados para determinar el mejor ajuste que también pase por el punto cero. En la **figura 2.2** se ilustran los tres métodos y cómo cada uno afecta el respectivo error por no linealidad. En general este error se expresa como un porcentaje de la salida a rango total. Por ejemplo, un transductor para medir presión tendría un error por no linealidad de ±0.5% del rango total.

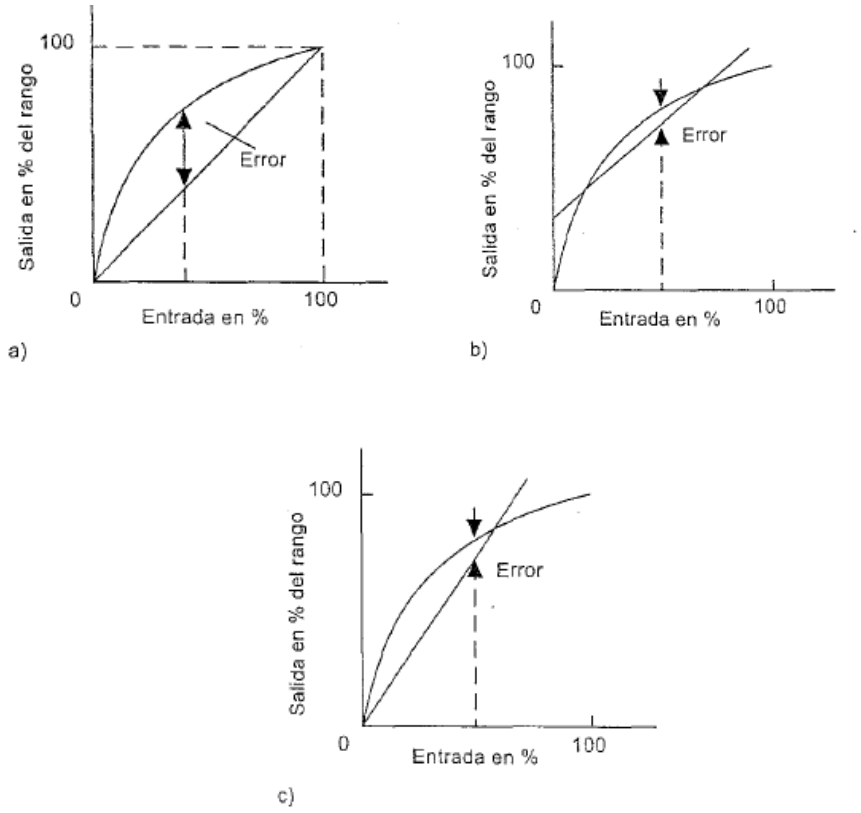



**Figura 2.2** *Error por no linealidad utilizando: a) Valores extremos del rango, b) la mejor línea recta que incluya todos los valores, c) la mejor línea recta que pase por el punto cero*

Repetibilidad/Reproducibilidad. Los términos repetibilidad y reproducibilidad se utilizan para describir la capacidad del transductor para producir la misma salida después de aplicar varias veces el mismo valor de entrada. Cuando ya no se logra obtener la misma salida después de aplicar el valor de entrada, el error se expresa como un porcentaje de la salida a rango total.

$$\text{Repetibilidad} = \frac{\text{val.max} - \text{vsl.mínimo obtenidos}}{\text{rango total}} \text{X}100$$

Se dice que un transductor para medir la velocidad angular tiene una repetitividad de ±0.01% del rango total a una velocidad angular determinada.

Estabilidad. La estabilidad de un transductor es su capacidad para producir la misma salida cuando se emplea para medir una entrada constante en un periodo. Para describir el cambio en la salida que ocurre en ese tiempo, se utiliza el término deriva. Ésta se puede expresar como un porcentaje del rango de salida total. El término deriva del cero se refiere a los cambios que se producen en la salida cuando la entrada es cero.

Banda/tiempo muerto. La banda muerta o espacio muerto de un transductor es el rango de valores de entrada durante los cuales no hay salida. Por ejemplo, en la fricción de rodamiento de un medidor de flujo con rotar significaría que no se produce salida hasta que la entrada alcanza cierto umbral de velocidad. El tiempo muerto es el lapso que transcurre desde la aplicación de una entrada hasta que la salida empieza a responder y cambiar.

Resolución. Cuando la entrada varía continuamente en todo el rango, las señales de salida de algunos sensores pueden cambiar a pequeños intervalos. Un ejemplo es el potenciómetro con devanado de alambre: la salida aumenta escalonada conforme el deslizador del potenciómetro pasa de una vuelta del devanado a otra. La resolución es el cambio mínimo del valor de la entrada capaz de producir un cambio observable en la salida. Por ejemplo, la resolución de un potenciómetro con devanado de alambre podría ser 0.5°, o quizás un porcentaje de la desviación a escala total. Para sensores con salida digital, el cambio mínimo de señal de salida sería de 1 bit. Por lo tanto, un sensor que produzca una palabra de datos de Nbits, es decir, un total de 2N bits, la resolución se expresaría como l/2N.

Impedancia de salida, Cuando un sensor que produce una salida eléctrica se vincula con un circuito electrónico, es necesario conocer la impedancia de salida dado que ésta se va a conectar en serie o en paralelo con dicho circuito. Al incluir el sensor, el comportamiento del sistema con el que se

conecta podría modificarse de manera considerable, En la sección 4.1.1 se aborda el tema de la carga.

Para ejemplificar lo anterior considere el significado de las siguientes especificaciones de un transductor expresión de galgos extensométricos:

Rangos: 70 a 1000 kPa, 2000 a 70 000 kPa
Voltaje de alimentación: 10 V cd oca, r,m,s,
Salida a rango total: 40 mV
Alinealidad e histéresis: ±0.5% de la salida a rango total
Rango de temperatura: - 54 DC a +120 DCen funcionamiento
Deriva del cero térmica: 0.030% de la salida a rango total/DC

El rango anterior indica que el transductor sirve para medir presiones entre 70 y 1000 kPa, o 2000 y 70 000 kPa. Para funcionar requiere una fuente de alimentación de 10 Vcd o Vca r.m.s., produce una salida de 40 mV cuando la presión en el rango inferior es 1000 kPa y cuando es 70 000 kPa en el rango superior. La no linealidad y la histéresis pueden producir errores de ±0.5% de 1000, es decir, ±5 kPa en el rango inferior y de ±O,5% de 70 000, es decir, ±350 kPa en el rango superior. Este transductor se puede utilizar entre -54 Y+120 DC de temperatura. Cuando la temperatura cambia en 1 ºC, la salida del transductor correspondiente a una entrada cero cambia 0.030% de 1000 = 0.3 kPa en el rango inferior y 0.030% de 70 000 = 21 kPa en el rango superior.

## 2.2 Características Estáticas y Dinámicas.

Las características estáticas son los valores obtenidos cuando se presentan condiciones de estado estable, es decir, valores obtenidos una vez que el transductor se estabiliza después de recibir cierta entrada. La terminología anterior se refiere a este tipo de estado, Las características dinámicas se refieren al comportamiento entre el momento en que cambia el valor de entrada y cuando el valor que produce el transductor logra su valor de estado estable. Las características dinámicas se expresan en función de la respuesta del transductor a entradas con determinadas formas. Por ejemplo, en una entrada tipo escalón, la entrada cambia bruscamente de O a un valor constante; en una entrada tipo rampa, la entrada se modifica a velocidad constante; o en una entrada senoidal con una frecuencia determinada.

## 2.3 Términos que se encuentran en los sistemas dinámicos:

Tiempo de respuesta: Es el tiempo que transcurre después de aplicar una entrada constante, una entrada escalón, hasta que el transductor produce una salida correspondiente a determinado porcentaje, como 95% del valor de la entrada (figura 2.3). Por ejemplo, si un termómetro de mercurio en tubo de vidrio se pone en un líquido caliente transcurrirá un lapso apreciable, quizás 100 s o más, antes de que el termómetro indique 95% de la temperatura real del líquido.

Constante de tiempo. Es el 63.2% del tiempo de respuesta. La constante de tiempo de un termopar en el aire podría ser de 40 a 100 s. La constante de tiempo es una medida de la inercia del sensor y de qué tan pronto reaccionará a los cambios en su entrada; cuanto mayor sea la constante de tiempo más lenta será su reacción ante una señal de entrada variable.

Tiempo de subida. Es el tiempo que requiere la salida para llegar a un porcentaje especificado de la salida en estado estable. Es común que el tiempo de subida se refiera al tiempo que tarda la salida en subir de 10% a 90% o 95% del valor en estado estable.

Tiempo de estabilización. Es el tiempo que tarda la salida en estabilizarse a un porcentaje de un valor determinado, por ejemplo, 2% del valor en estado estable.

Para ilustrar lo anterior, considere los siguientes datos sobre cómo cambiaron con el tiempo las lecturas de un instrumento, obtenidas en un termómetro hundido en un líquido en el tiempo t = O. Se requiere el tiempo necesario para el 95% de la respuesta.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Tiempo (s) | 0 | 30 | 60 | 90 | 120 | 150 | 180 |
| Temp. (ºC) | 20 | 28 | 34 | 39 | 43 | 46 | 49 |
| Tiempo (s) | 210 | 240 | 270 | 300 | 330 | 360 | |
| Temp. (ºC) | 51 | 53 | 54 | 55 | 55 | 55 | |

**Tabla 2.1** *Cambio de lecturas de un termómetro en función del tiempo*

La **figura 2.3** muestra una gráfica de la variación de la temperatura en el tiempo que indica el termómetro. El valor del estado estable es de 55.oC y dado que 95% de 55es 52.25 °C, el tiempo de respuesta para 95% es casi 228 s.

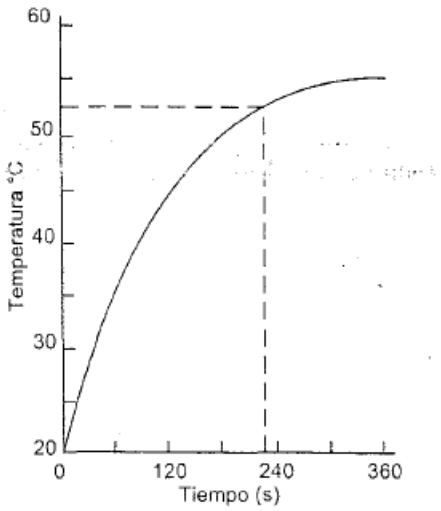



**Figura 2.3** *Termómetro en un líquido*

## 2.4 Tipos de Sensores

### 2.4.1 Detectores de ultrasonidos

Los detectores de ultrasonidos resuelven los problemas de detección de objetos de prácticamente cualquier material. Trabajan en ambientes secos y pulverulentos. Normalmente se usan para control de presencia/ausencia, distancia o rastreo.

### 2.4.2 Interruptores básicos

Se incluyen interruptores de tamaño estándar, miniatura, subminiatura, herméticamente sellados y de alta temperatura. Los mecanismos de precisión se ofrecen con una amplia variedad de actuadores y características operativas. Los interruptores de Sensores de Control son idóneos para aplicaciones que requieran tamaño reducido, poco peso, repetitividad y larga vida.

### 2.4.3 Interruptores final de carrera

Sensores de Control ofrece la línea de interruptores de precisión de acción rápida más avanzada del mundo para una amplia gama de aplicaciones. Las versiones selladas son estancas a la humedad y otros contaminantes. Los modelos antideflagrantes están diseñados para uso en lugares peligrosos.

### 2.4.4 Interruptores manuales

La amplia selección de productos incluye pulsadores, indicadores, manipulados, balancines, selectores rotativos y conmutadores de enclavamiento **Figura 2.4**. Estos productos ayudan al ingeniero con ilimitadas opciones en técnicas de indicación visual, actuación y disposición de componentes. Muchas versiones satisfacen especificaciones militares.

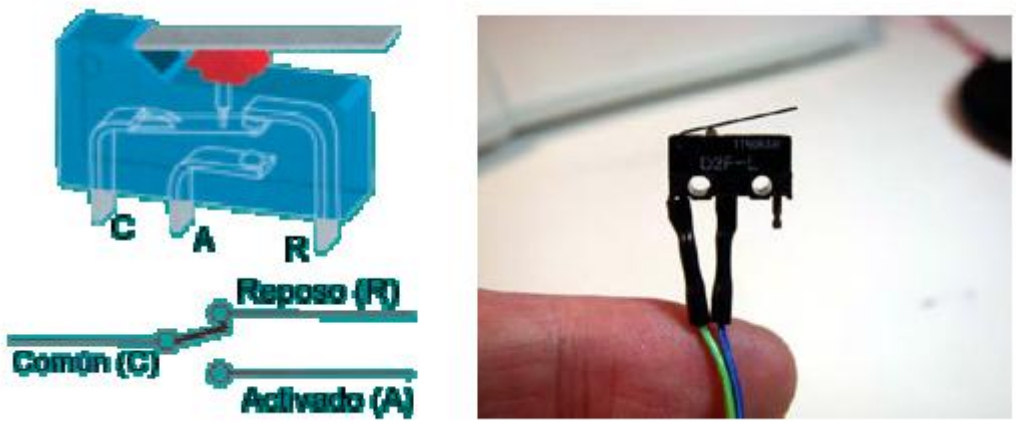



**Figura 2.4** *Interruptor – Sensor de contacto tipo pulsador*

**Productos encapsulados**

Diseños robustos, de altas prestaciones y resistentes al entorno o herméticamente sellados. Esta selección incluye finales de carrera miniatura, interruptores básicos estándar y miniatura, interruptores de palanca y pulsadores luminosos.

**Productos para fibra óptica**

El grupo de fibra óptica está especializado en el diseño, desarrollo y fabricación de componentes optoelectrónicos activos y submontajes para el mercado de la fibra óptica. Los productos para fibra óptica son compatibles con la mayoría de los conectores y cables de fibra óptica multimodo estándar disponibles actualmente en la industria. También se pueden ofrecer productos bajo especificación del cliente; son productos estándar con pequeñas variaciones para cumplir requisitos especiales. Se desarrollan continuamente nuevos productos.

### 2.4.5 Sensores potenciométricos

Un potenciómetro es un elemento resistivo que tiene un contacto deslizante que puede desplazarse a lo largo de dicho elemento. Éste se puede utilizar

tanto -en desplazamientos lineales como rotacionales; dicho desplazamiento se convierte en una diferencia de potencial. El potenciómetro rotacional está formado por una pista o canal circular con devanado de alambre o por una capa de plástico conductor; sobre la pista rota un contacto deslizante giratorio (**figura 2.5**) y ésta puede ser una sola circunferencia o helicoidal. Con un voltaje de entrada constante Vs entre las terminales 1 y 3, el voltaje de salida Va entre las terminales 2 y 3 es una fracción del voltaje de entrada, la fracción que depende de la relación de resistenciaR23 entre las terminales 2 y 3 comparada con la resistencia total R13 entre las terminales

1 y 3, es decir: Va/Vs = R23/R13 Si la resistencia de la pista por unidad de longitud (por ángulo unitario) es constante, entonces la salida es proporcional al ángulo a lo largo del cual gira el deslizador. En este caso un desplazamiento angular se puede convertir en una diferencia de potencial.

En una pista con devanado de alambre, al pasar de una vuelta a la otra, la parte deslizante cambia la salida de voltaje en pasos, cada uno de los cuales corresponde al avance de una vuelta. Si el potenciómetro tiene N vueltas, la resolución expresada en porcentaje es de 100/N. Por lo tanto, la resolución de una pista de alambre está limitada por el diámetro del alambre utilizado y su valor suele variar entre 1.5 mm en pistas con devanado burdo, y hasta 0.5 mm para pistas con devanado fino. Los errores por la no linealidad de la pista varían de menos de 0.1% hasta casi 1%. La resistencia de la pista varía entre 20Ω y 200 kΩ. El plástico conductor idealmente tiene una resolución infinita, y los errores por la no linealidad de la pista son del orden de 0.05% y valores de resistencia entre 500 Ω y 80 kΩ. El coeficiente por temperatura de la resistencia del plástico conductor es mayor que el del alambre, por lo que los cambios de temperatura tienen mayor influencia en la exactitud.

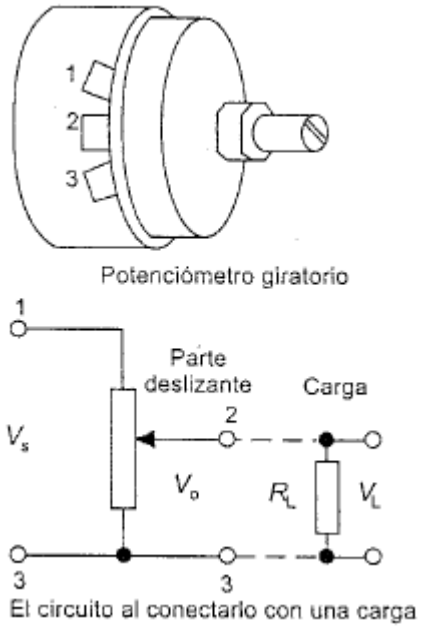



**Figura 2.5** *Potenciómetro giratorio*

### 2.4.6 Sensores para automoción

Se incluyen sensores de efecto Hall, de presión y de caudal de aire. Estos sensores son de alta tecnología y constituyen soluciones flexibles a un bajo coste. Su flexibilidad y durabilidad hace que sean idóneos para una amplia gama de aplicaciones de automoción.

### 2.4.7 Sensores de caudal de aire

Los sensores de caudal de aire contienen una estructura de película fina aislada térmicamente, que contiene elementos sensibles de temperatura y calor. La estructura de puente suministra una respuesta rápida al caudal de aire u otro gas que pase sobre el chip.

### 2.4.8 Sensores de corriente

Los sensores de corriente monitorizan corriente continua o alterna. Se incluyen sensores de corriente lineales ajustables, de balance nulo, digitales y lineales. Los sensores de corriente digitales pueden hacer sonar una alarma, arrancar un motor, abrir una válvula o desconectar una bomba. La señal lineal duplica la forma de la onda de la corriente captada, y puede ser utilizada

como un elemento de respuesta para controlar un motor o regular la cantidad de trabajo que realiza una máquina.

### 2.4.9 Sensores de efecto Hall

Cuando un haz de partículas cargadas atraviesa un campo magnético existen fuerzas que actúan sobre las partículas, y la trayectoria lineal del haz se deforma. Cuando una corriente fluye a través de un conductor se comporta como un haz de partículas en movimiento, por lo que al pasar por un campo magnético esta corriente se puede desviar. Este efecto fue descubierto por E.R. Hall en 1879 y se conoce como Efecto Hall.

Estos sensores sirven como sensores de posición, desplazamiento y proximidad cuando se dota al objeto que se desea detectar con un pequeño imán permanente. Un ejemplo es el sensor que se utiliza para determinar el nivel de combustible del depósito de gasolina de un auto. En el flotador se coloca un imán y conforme el nivel del combustible cambia, también se modifica la distancia que separa al flotador del sensor Hall (**figura 2.6**). El resultado es una salida con voltaje Hall que corresponde a una medida de la distancia entre el flotador y el sensor y, por lo tanto, del nivel de combustible en el tanque.

Otra aplicación de los sensores de efecto Hall es en motores de cd sin escobillas. En éstos es necesario determinar si el rotar de imán permanente está alineado de manera correcto con los devanados del estator a fin de que la corriente que circula a través de ellos pueda pasar en el instante correcto y mantener girando el rotar. Los sensores de efecto Hall sirven para detectar si la alineación es correcta.

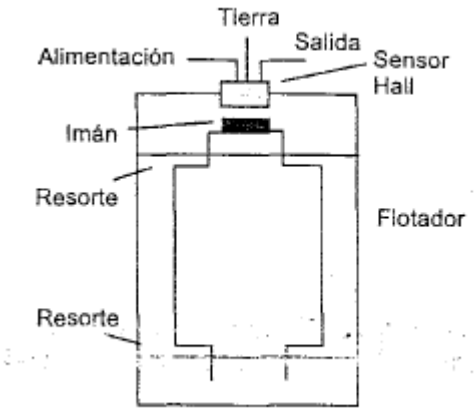



**Figura 2.6** *Detector de Nivel de Fluido*

### 2.4.10 Sensores de humedad

Los sensores de humedad relativa/temperatura y humedad relativa están configurados con circuitos integrados que proporcionan una señal acondicionada. Estos sensores contienen un elemento sensible capacitivo en base de polímeros que interacciona con electrodos de platino. Están calibrados por láser y tienen una intercambiabilidad de +5% HR, con un rendimiento estable y baja desviación.

### 2.4.11 Sensores de posición de estado sólido

Los sensores de posición de estado sólido, detectores de proximidad de metales y de corriente, están disponibles en varios tamaños y terminaciones. Estos sensores combinan fiabilidad, velocidad, durabilidad y compatibilidad con diversos circuitos electrónicos para aportar soluciones a las necesidades de aplicación.

Estos dispositivos detectan señales para actuar en un determinado proceso u operación, teniendo las siguientes características:

- Son dispositivos que actúan por inducción al acercarles un objeto.
- No requieren contacto directo con el material a sensar.
- Son los más comunes y utilizados en la industria
    - Se encuentran encapsulados en plástico para proveer una mayor facilidad de montaje y protección ante posibles golpees

**Aplicaciones:**

Control de cintas transportadoras,
Control de alta velocidad
Detección de movimiento
Conteo de piezas,
Censado de aberturas en sistemas de seguridad y alarma
Sistemas de control como finales de carrera. (PLC´s)
Sensor óptico.

**Características:**

Son de confección pequeña, pero robustos

Mayor distancia de operación.
Detectan cualquier material.
Larga vida útil.

### 2.4.12 Sensores Inductivos

Consiste en un dispositivo conformado por:

- Una bobina y un núcleo de ferrita.
- Un oscilador.
- Un circuito detector (etapa de conmutación)
- Una salida de estado sólido.

El oscilador crea un campo de alta frecuencia de oscilación por el efecto electromagnético producido por la bobina en la parte frontal del sensor centrado con respecto al eje de la bobina. Así, el oscilador consume una corriente conocida. El núcleo de ferrita concentra y dirige el campo electromagnético en la parte frontal, convirtiéndose en la superficie activa del sensor.
Cuando un objeto metálico interactúa con el campo de alta frecuencia, se inducen corrientes EDDY en la superficie activa. Esto genera una disminución de las líneas de fuerza en el circuito oscilador y, en consecuencia, desciende la amplitud de oscilación. El circuito detector reconoce un cambio específico en la amplitud y genera una señal, la cual cambia (pilotea) la salida de estado sólido a “ON” u “OFF”. Cuando se retira el objeto metálico del área de senado, el oscilador genera el campo, permitiendo al sensor regresar a su estado normal.

### 2.4.13 Sensores Capacitivos

Un sensor capacitivo es adecuado para el caso de querer detectar un objeto no metálico. Para objetos metálicos es más adecuado escoger un sensor inductivo.
Para distancias superiores a los 40 mm es totalmente inadecuado el uso de este tipo de sensores, siendo preferible una detección con sensores ópticos o de barrera.

**Figura 2.7** *Sensor de Proximidad Tipo Capacitivo*

Los sensores capacitivos funcionan de manera similar a un capacitor simple.
La lámina de metal [1] en el extremo del sensor esta conectado eléctricamente a un oscilador [2].

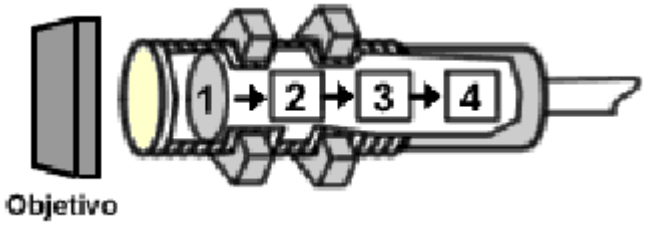



**Figura 2.8** *Etapas de constituyen un Sensor Capacitivo*

El objeto que se detecta funciona como una segunda lámina. Cuando se aplica energía al sensor el oscilador percibe la capacitancia externa entre el objetivo y la lámina interna.
Los sensores capacitivos funcionan de manera opuesta a los inductivos, a medida que el objetivo se acerca al sensor capacitivo las oscilaciones aumentan hasta llegar a un nivel limite lo que activa el circuito disparador [3] que a su vez cambia el estado del switch [4].

### 2.4.14 Sensores de presión y fuerza

Los sensores de presión son pequeños, fiables y de bajo coste. Ofrecen una excelente repetitividad y una alta precisión y fiabilidad bajo condiciones ambientales variables. Además, presentan unas características operativas constantes en todas las unidades y una intercambiabilidad sin recalibración. Sensores de Control le ofrece cuatro tipos de sensores de medición de presión: absoluta, diferencial, relativa y de vacío y rangos de presión desde ±1,25 kPa a 17 bar.

### 2.4.15 Sensores de temperatura

Los sensores de temperatura se catalogan en dos series diferentes: TD y HEL/HRTS. Estos sensores consisten en una fina película de resistencia variable con la temperatura (RTD) y están calibrados por láser para una mayor precisión e intercambiabilidad. Las salidas lineales son estables y rápidas.

### 2.4.16 Detectores de Temperatura por Resistencia

La resistencia de la mayoría de los metales aumenta, en un rango limitado de temperatura, de manera razonablemente lineal con la temperatura (**figura 2.9** ). Para una relación lineal, se tiene que:

$$R_t = R_0(1 + at)$$

Donde R es la resistencia a una temperatura de t °C, Ro la resistencia a 0°C y a una constante del metal denominada coeficiente de temperatura de la resistencia. Los detectores de temperatura por resistencia

(DTR) son elementos resistivos sencillos que adoptan la forma de bobinas de alambre hechas de platino, níquel o aleaciones níquel-cobre; el platino es el que más se utiliza. Los elementos hechos de delgadas películas de platino en general se obtienen depositando el metal en un sustrato adecuado; los elementos de bobina por lo general consisten en un alambre de platino sujeto con un adhesivo de vidrio para altas temperaturas en el interior de un tubo de cerámica.

Estos detectores son muy estables y sus respuestas son reproducibles durante largos periodos. Sus tiempos de respuesta tienden a ser del orden de 0.5 a 5 s, o mayores.

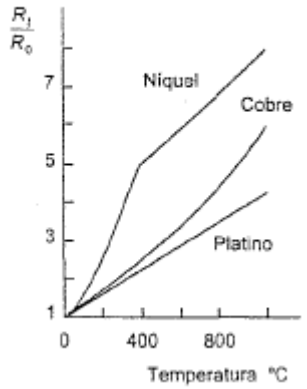



**Figura 2.9** *Variación de la resistencia en los metales en Función a la temperatura*

### 2.4.17 Termistores

Los termistores son pequeñas piezas de materiales hechos con la mezcla de óxidos metálicos, por ejemplo, de cromo, cobalto, hierro, manganeso y níquel. Todos estos óxidos son semiconductores. El material puede tener formas diversas como cuentas, discos y varillas (**figura 2.10**). La resistencia de los termistores convencionales de óxido metálico disminuye de una manera no lineal con el aumento en la temperatura, como ilustra la figura 2.10. Estos termistores tienen coeficientes de temperatura negativos (CTN), aunque también los hay con coeficientes de temperatura positivos (CTP). El cambio de la resistencia por cada grado de temperatura que cambie es mucho mayor que el que ocurre con los metales. La relación resistencia-temperatura de un termistor

Se puede expresar mediante la siguiente ecuación:

$$R_t = Ke^{\beta/t}$$

Donde Rt es la resistencia de la temperatura t, y K y β son constantes. Si se comparan con otros sensores de temperatura, los termistores ofrecen muchas ventajas. Son fuertes y pueden ser muy pequeños, por lo cual permiten el monitoreo de temperaturas casi en cualquier punto. Gracias a su reducido tamaño, responden muy rápido a los cambios de temperatura, Producen cambios de resistencia muy grandes por cada grado de cambio en la temperatura, pero su principal desventaja es su no linealidad.

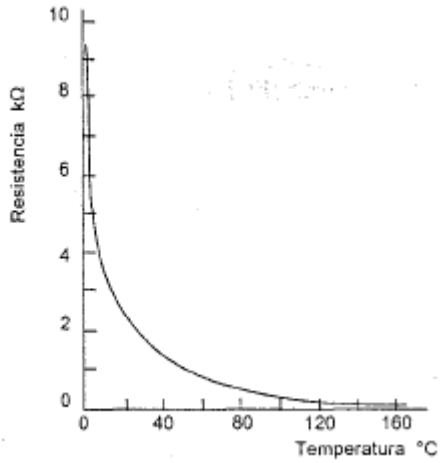



**Figura 2.10** Variación de la resistencia en función de la temperatura de un termistor típico

### 2.4.18 Termodiodos y transistores

El diodo semiconductor de unión con frecuencia se utiliza como sensor de temperatura. Cuando cambia la temperatura de semiconductores con impurezas, también se modifica la movilidad de sus portadores de carga, lo cual afecta la velocidad de difusión de electrones y huecos a través de pna unión p-n. Por lo tanto, si una unión p-n tiene una diferencia de potencial V, la corriente 1 que circula por la unión será función de la temperatura, la cual está dada por:

$$I = I_0(e^{eV/kT} - 1)$$

Donde T es la temperatura en la escala Kelvin, e la carga de un electrón y k e lo son constantes. Utilizando logaritmos, la ecuación anterior se puede expresar, en función del voltaje, de la siguiente manera:

$$V = \left(\frac{kT}{e}\right)\ln\left(\frac{I}{I_0} + 1\right)$$

Así, si la corriente es constante, V es proporcional a la temperatura en la escala Kelvin, por lo que la medida de la diferencia de potencial en un diodo de corriente constante puede servir como medida de la temperatura. Este tipo de sensores son tan compactos como los termistores, pero tienen además la gran ventaja de que su respuesta es una función lineal de la temperatura. Circuitos integrados como el LM3911 o el LM35 tienen este tipo de diodos que se utilizan como sensores de temperatura, y proporcionan el

acondicionamiento de señal respectivo. El voltaje de salida del LM3911 es proporcional a la temperatura a razón de 10 mV/ºC.

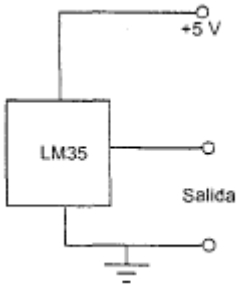



**Figura 2.11** *LM35*

### 2.4.19 Sensores de turbidez

Los sensores de turbidez aportan una información rápida y práctica de la cantidad relativa de sólidos suspendidos en el agua u otros líquidos. La medición de la conductividad da una medición relativa de la concentración iónica de un líquido dado.

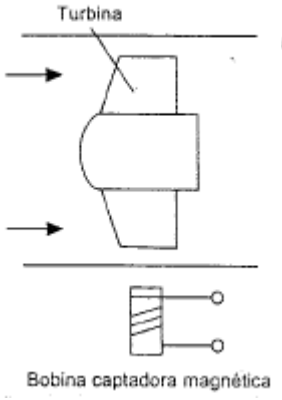



**Figura 2.12** *Medidor de Flujo de Turbina*

### 2.4.20 Sensores magnéticos

Los sensores magnéticos se basan en la tecnología magnetoresisitiva SSEC. Ofrecen una alta sensibilidad. Entre las aplicaciones se incluyen brújulas, control remoto de vehículos, detección de vehículos, realidad virtual, sensores de posición, sistemas de seguridad e instrumentación médica.

### 2.4.21 Sensores de presión

Los sensores de presión están basados en tecnología piezoresistiva, combinada con microcontroladores que proporcionan una alta precisión, independiente de la temperatura, y capacidad de comunicación digital directa con PC. Las aplicaciones afines a estos productos incluyen instrumentos para aviación, laboratorios, controles de quemadores y calderas, comprobación de motores, tratamiento de aguas residuales y sistemas de frenado.

La balanza de resorte es un ejemplo de sensor de fuerza; en ella se aplica una fuerza, un peso, al platillo y ésta provoca un desplazamiento, es decir, el resorte se estira. El desplazamiento es entonces, una medida de la fuerza. Las fuerzas por lo general se miden con base en un desplazamiento. El siguiente método ilustra lo anterior.

## 2.5 Indicador de Presiones con Deformímetro

Una modalidad muy común de transductor para medir fuerza se basa en el empleo de deformímetros de resistencia eléctrica para monitorear la deformación de cierto elemento cuando éste se estira, comprime o dobla por la aplicación de una fuerza. A este transductor se le conoce como indicador depresiones; en la **figura 2.13** se muestra un ejemplo. El indicador de presiones es un tubo cilíndrico en el que se colocan deformímetros. Al aplicar fuerzas para comprimir el cilindro, los deformímetros producen un cambio de resistencia, el cual es la medida de la deformación y, por lo tanto, de las fuerzas aplicadas. Dado que la temperatura también produce cambios en la resistencia, el circuito acondicionador de señal que se utilice deberá eliminar los efectos debidos a la temperatura. Por lo general, estos indicadores de presión se utilizan para fuerzas de hasta 10MN, su error aproximado por no linealidad es de ±0.03% del rango total, el error por histéresis de ±0.02% del rango total y el error de repetibilidad de ±O.02% del rango total. Los indicadores de presión con deformímetros que se basan en el doblamiento de un elemento metálico se deben usar para fuerzas menores, por ejemplo, para rangos de 0 a 5N y hasta 0 a 50 kN. Los errores más comunes se deben a un

error por no linealidad de casi ±0.03% del rango total, el error por histéresis de ±0.02% del rango total y el error de repetibilidad de ±0.02% del rango total.

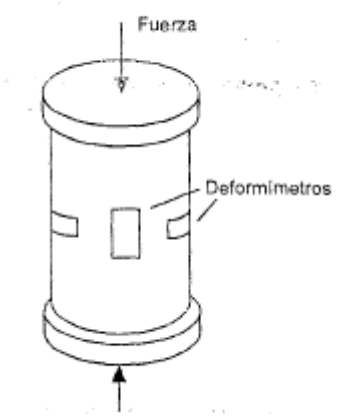



**Figura 2.13** *Indicador de Presiones con deformímetro*

## 2.6 Presión de fluidos

En muchos de los dispositivos utilizados para monitorear la presión de fluidos de procesos industriales se monitorea la deformación elástica de diafragmas, cápsulas, fuelles y tubos. Los tipos de medición que se necesitan son: presión absoluta, en cuyo caso la presión que se mide es relativa a una presión cero, es decir, al vacío; presión diferencial, con la cual se mide una diferencia de presiones, y presión manométrica, en la que la presión se mide en relación con la presión barométrica.

En un diafragma (**figura 2.14**) hay una diferencia de presión entre ambas caras, por lo que el centro del diafragma se desplaza. Un diafragma corrugado ofrece mayor sensibilidad. El movimiento del diafragma se monitorea mediante un sensor de desplazamiento que puede ser un deformímetro, como se muestra en la **figura 2.15**. Es frecuente utilizar deformímetros de diseño especial, los cuales constan de cuatro deformímetros, dos para medir el esfuerzo en la dirección de la circunferencia y dos en dirección radial. Los cuatro deformímetros se conectan de manera que formen los brazos de un puente de Wheatstone. Es posible adherir los deformímetros al diafragma, pero también existe la opción de hacer un diafragma de silicio en el que los deformímetros son áreas especiales del diafragma con impurezas.

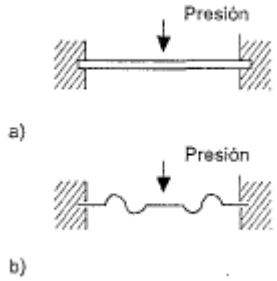



**Figura 2.14** *Diafragmas a) Plano y; b) Corrugado*

Otra forma de sensor de presión con diafragma de silicio es el que utilizan en los sensores de presión Motorola MPX. El deformímetro se integra, junto con un circuito resistivo, en un solo chip de diafragma de silicio. Cuando una corriente pasa a través del deformímetro se le aplica una presión en ángulo recto, se produce un voltaje en dirección transversal. El sensor MPX cuenta con todo lo anterior, así como con circuitos para acondicionar la señal y para compensar la temperatura. El voltaje de salida es directamente proporcional a la presión. Existen sensores como el anterior para medir presión absoluta (las terminaciones del sistema de numeración MX son A, AP, AS o ASX), presión diferencial (terminaciones D o DP) y presión manométrica (terminaciones GP, GVP, GS, GVS, GSV o GVSX). Por ejemplo, la serie MPX2l00 tiene un rango de presión de 100 kPa con un voltaje de 16V; cd, produce para las modalidades de presión absoluta y presión diferencial, una salida de voltaje para un rango total de 40 mV.

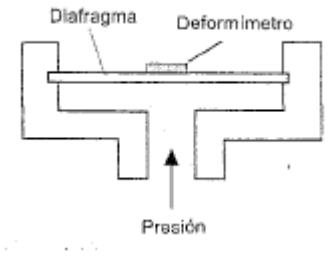



**Figura 2.15** *Deformímetro de Diafragma*

El tiempo de respuesta, l0 a 90%, para un escalón de O a 100 kPa es alrededor de 1.0 ms y la impedancia de salida del orden de 1.4 a 3.0 kΩ. Los sensores de presión absoluta tienen diversas aplicaciones como altímetros y barómetros; los sensores de presión diferencial para medir el flujo de aire, y

los sensores de presión manométrica para medir la presión en motores y llantas.

### 2.7 Sensores neumáticos

Los sensores neumáticos utilizan aire comprimido, y el desplazamiento, o la proximidad de un objeto se transforma en un cambio en la presión del aire. La **figura 2.16** muestra la configuración básica de estos sensores. Un puerto en el frente del sensor deja salir aire a baja presión. Este aire, en ausencia de un objeto cercano, escapa y al hacerlo reduce la presión en el puerto de salida del sensor más próximo. Sin embargo, si hay un objeto cerca, el aire no escapa con facilidad y la presión aumenta en el puerto de salida del sensor. La presión de salida del sensor dependerá, por lo tanto, de la cercanía de los objetos. Estos sensores se usan para medir desplazamientos de fracciones de milímetros, para rangos característicos de 3 a 12 mm.

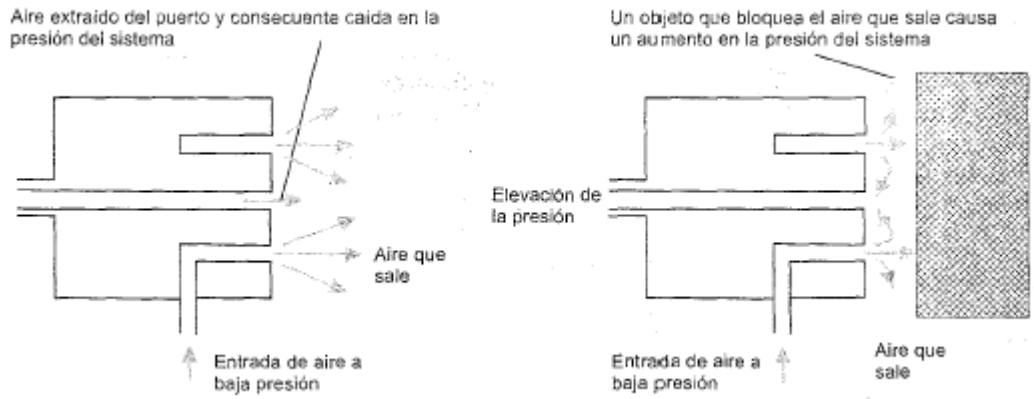



**Figura 2.16** *Sensor de Proximidad Neumático*

### 2.8 Sensor Táctil

El sensor táctil o de tacto es una forma particular de sensor de presión. Se utiliza en 'las yemas de los dedos' de las 'manos' de los robots para determinar en qué momento la 'mano' tiene contacto con un objeto. También se utiliza en las pantallas 'sensibles al tacto', donde se requiere detectar contactos físicos. En una modalidad de sensor táctil se utiliza una capa de fluoruro de polivinilideno piezoeléctrico (PVDP, por sus siglas en inglés). Se usan dos capas de la película separadas con una capa suave, la cual transmite las vibraciones (**figura 2.17**). A la capa inferior de PVDP se le aplica

un voltaje alterno que produce oscilaciones mecánicas en la película (es el caso inverso del efecto piezoeléctrico descrito antes). La película intermedia transmite esta vibración, en la capa de PVDP de la parte superior. Debido al efecto piezoeléctrico, estas vibraciones producen un voltaje alterno a través de la película superior. Cuando se aplica presión a la película superior de PVDP se afectan sus vibraciones y se modifica el voltaje alterno de salida.

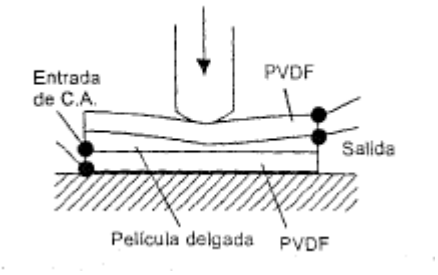



**Figura 2.17** *Sensor de tacto PVDF*

### 2.9 Selección de los Sensores

Al seleccionar un sensor para una aplicación en particular hay que considerar varios factores:

- El tipo de medición que se requiere, por ejemplo, la variable que se va a medir, su valor nominal, el rango de valores, la exactitud, velocidad de medición y confiabilidad requeridas, las condiciones ambientales en las que se realizará la medición.
- El tipo de salida que se requiere del sensor, lo cual determinará las condiciones de acondicionamiento de la señal, a fin de contar con señales de salida idóneas para la medición.
- Con base en lo anterior se pueden identificar algunos posibles sensores, teniendo en cuenta rango, exactitud, linealidad, velocidad de respuesta, confiabilidad, facilidad de mantenimiento, duración, requisitos de alimentación eléctrica, solidez, disponibilidad y costo.

La elección de un sensor no se puede hacer sin considerar el tipo de salida que el sistema debe producir después de acondicionar la señal; por ello, es necesaria una integración idónea entre sensor y acondicionador de señal.
Como ejemplo de lo anterior, considere la selección de un sensor para medir el nivel de ácido corrosivo de un recipiente. Dicho nivel varía entre 0 y 2 m y el recipiente es de forma circular con diámetro de 1 m. El recipiente vacío pesa 100 kg. La mínima variación en nivel que se desea detectar es 10 cm.

La densidad del ácido es 1050kg/$m^3$. El sensor debe producir una salida eléctrica.

Debido a lo corrosivo del ácido, lo adecuado es emplear un método indirecto para determinar el nivel. Por lo tanto, se utilizaría uno o varios indicadores de presión, para monitorear el peso del recipiente. Estos indicadores producen una salida eléctrica. El peso del líquido cambia de 0, cuando el recipiente está vacío, a 1050 X 2 X π($l^2$/4) X 9.8 = 16.2 kN, cuando está lleno. Si el peso anterior se añade al del recipiente cuando está vacío, se obtiene un peso que varía de 1 a 17 kN. La resolución requerida es de cambios de nivel de 10 cm, es decir, cambios de peso de 0.10 X 1050 X π ($l^2$/4 X 9.8 = 0.8kN. Si para sostener el tanque se utilizan tres indicadores de presión, cada uno de ellos necesitará un rango de aproximadamente 0 a 6 kN y una resolución de 0.27 kN. A continuación se consultarán catálogos de fabricantes para verificar si dichos indicadores de presión están a la venta.

## 2.10 Módulos de Envío y Recepción de Datos (Tx – Rx) para comunicación inalámbrica4

### 2.10.1 Comunicación por infrarrojos:

Al hablar de comunicación inalámbrica lo primero que se piensa es en señales de radio. Sin embargo, olvidamos que nos comunicamos habitualmente con equipos electrónicos utilizando una tecnología que se ha vuelto muy común, extremadamente sofisticada y eficaz: las comunicaciones mediante infrarrojos. Cuando se opera un control remoto, lo que uno hace es comunicarse por medio de luz en la gama de los infrarrojos.

### 2.10.2 Comunicación por Radiofrecuencia (RF):

Una de las maneras más prácticas de trabajar con un robot o un dispositivo de salida móvil, es programar sus funciones en una computadora y comandar el dispositivo de manera remota por medio de algún enlace. Los datos de los sensores del dispositivo llegan a la computadora central por un enlace y los comandos para los movimientos llegan al dispositivo por otro (otra frecuencia, si hablamos de RF, porque la interconexión se puede hacer también por un

[4] Texto tomado de **http://www.robots-argentina..com.ar**

enlace de infrarrojos). Para comunicarse por medio de un enlace de RF con un dispositivo es posible usar receptores integrados muy pequeños y de bajo costo, como el **RWS-433**, o el **RXLC-434**, y otros similares, que trabajan en frecuencias de entre 303 y 433 Mhz. La elección de los transmisores dependerá de la distancia entre receptor y emisor.

### 2.10.3 Módulos Tx-Rx por Infrarrojos

La optoelectrónica es la integración de los principios ópticos y la electrónica de semiconductores. Los componentes optoelectrónicos son sensores fiables y económicos. Se incluyen diodos emisores de infrarrojos (IREDs), sensores y montajes.

Se pueden usar infrarrojos para la detección de obstáculos, los cuales se denominan Sensores reflectivos. Prácticamente con los mismos elementos que se usan para la comunicación, aunque con una distribución física distinta, es posible hacer rebotar sobre un obstáculo la emisión de una señal infrarroja codificada tal como en los controles remotos y detectarla cuando llega de regreso al robot, con lo cual, en lugar de comunicación tenemos un sistema para la detección de obstáculos. Esta tarea se ha eficientizado al máximo en dispositivos como los GP2D02, GP2D05, GP2D12, etc., de Sharp, que utilizan infrarrojos para medir la distancia a la que se encuentran los objetos dentro de un determinado rango.

**Figura 2.18** *Medidor de Distancias por infrarrojos*

El enlace se divide, obviamente, en una sección de emisión y otra de recepción. Los elementos utilizados en los emisores son LEDs especializados, y en ese caso lo más importante es elegirlos bien en base a su potencia de emisión, tipo de lentilla, consumo de energía y frecuencia de operación. Una opción práctica es utilizar un control remoto "universal" como los que se venden en la actualidad a un precio bastante bajo.

La parte de la recepción de la señal infrarroja es la más crítica.

Lo delicado del sistema está en el receptor. Debe ser capaz de separar la señal real de otras radiaciones de infrarrojo, como la de la luz del sol, e incluso la de algunos equipos de iluminación incandescente, y de fuentes de calor en general. Como la señal la emitimos desde el robot, y a pesar de que utilicemos los elementos más sofisticados y de mayor potencia la energía emitida no será muy grande, los receptores deben ser bien sensibles. Y entonces el problema que se presenta es que pueden ser "cegados" por radiaciones naturales como la de la luz del día, por dar un ejemplo.

Los dispositivos disponibles en el comercio, que son los que se utilizan en los equipos con control remoto, están diseñados especialmente para este uso. Estos sensores tienen señales de salida fácilmente adaptables a los microcontroladores, así que son fáciles de conectar. La parte óptica —lentilla y filtro— también está solucionada.

### 2.10.4 Receptores de Infrarrojos:

Los receptores de infrarrojos codificados **Figura 2.19** integran en un chip el elemento sensible al infrarrojo, una lente, un filtro de espectro y toda la lógica necesaria para distinguir señales moduladas a una determinada frecuencia.


**Figura 2.19** *Receptor de infrarrojo integrado*

Para facilitar a los lectores la posibilidad de implementar un enlace así en sus sistemas, voy a tomar como ejemplo el receptor que encontré disponible en más lugares en el mercado de electrónica de Argentina.

### 2.10.5 Receptor de infrarrojos IRM8601S

El diagrama interno de un encapsulado infrarrojo IRM8601S es como se lo muestra en la **Figura 2.20**

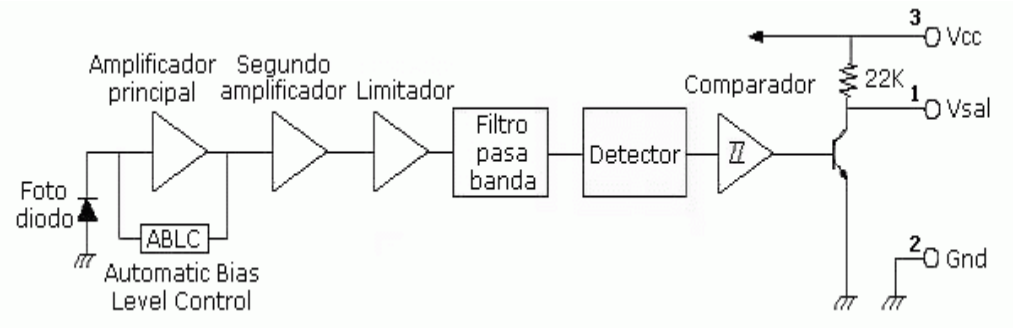



**Figura 2.20** *Diagrama lógico del IRM8601S*

El receptor está disponible en una cápsula similar a los transistores TIP **Figura 2.21** y, como los transistores, también tiene tres patas. Existe también una cápsula con cobertura metálica.

**Figura 2.21** *Versión metálica del IRM8601S*

La conexión es muy simple **Figura 2.22,** una de las patas es la alimentación de 5V, la otra la señal de salida y la tercera es el común o tierra.

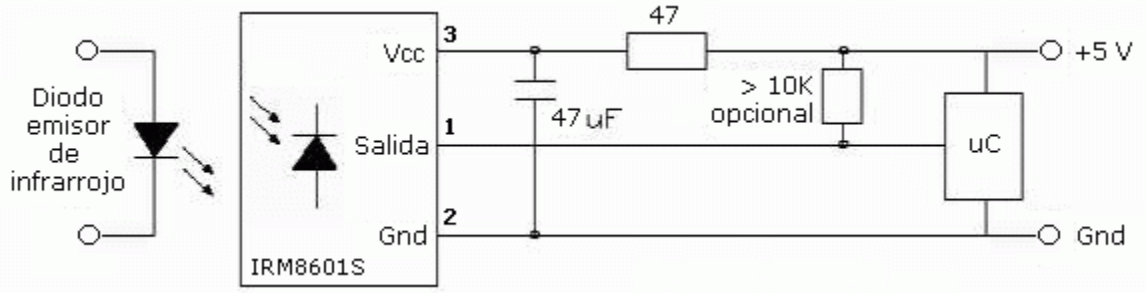



**Figura 2.22** *Circuito de aplicación del IRM8601S*

La hoja de datos lista las siguientes características:

- Inmunidad contra interferencias electromagnéticas.
- Disponible en cápsula metálica.
- Lente elíptico que mejora la recepción
- Bajo voltaje y bajo consumo
- Alta inmunidad a la luz ambiente
- Fotodiodo con circuito integrado
- Compatible con TTL y CMOS
- Recepción a larga distancia
- Elevada sensibilidad

Otros receptores de tipo similar:

- Vishay TSOP 1738
- Vishay TSOP 1838
- Vishay TSOP 11.. series
- Siemens SFH 506 (discontinuado)
- Siemens SFH 5110 (sucesor del SFH 506)
- Radio Shack 276-0137
- Everlight IRM 8100-3-M (Radio Shack part no. 276-0137B)
- Mitsumi IR Preamp KEY-COOSV (0924G)
- TOSHIBA TK19 444 TFMS 5360
- TEMIC TFMS 5380 Por Telefunken Semiconductors
- Sharp IS1U60 (Disponible como RS)
- Sony SBX 1620-12
- Sharp GP1U271R
- Kodenshi PIC-12043S
- Daewoo DHR-38 C 28

### 2.10.6 Emisores de infrarrojo

La otra parte del sistema, la emisión, se puede solucionar con un control remoto universal. Los receptores como el que se describe están ajustados para estos emisores de infrarrojos para electrodomésticos, así que quizás no sea práctico complicarse con otros circuitos. Una de las maneras más directas será utilizar el mando para enviar órdenes al dispositivo. La otra sería

"hackear" el control remoto y utilizar su plaqueta para nuestro emisor, conectando el dispositivo con el teclado.

Si de todos modos se desea implementar un circuito, se puede utilizar, por ejemplo, el integrado codificador HT12E, que codifica 12 entradas (8 de dirección y 4 de datos, o comandos) en una señal en serie (para decodificarlas se utilizaría su el HT12D). El circuito a utilizar es:

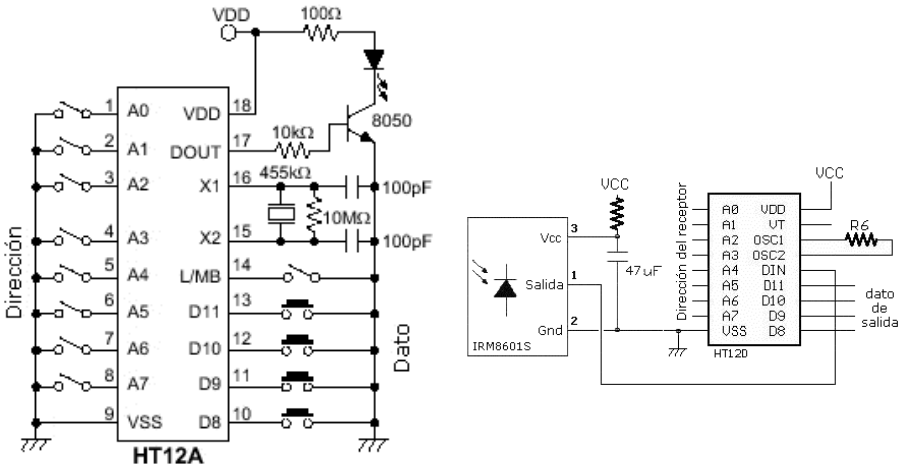



**Figura 2.23** *Circuito de un receptor con HT12D y Circuito de aplicación del HT12A*

Estudiando el formato de las señales del estándar que utilizan los controles remotos (o mando a distancia, como se les llama en algunos lugares), se puede crear la señal por programa, en un microcontrolador.

### 2.10.7 MCP2120: Codificador / Decodificador de infrarrojo

El integrado MCP2120 de Microchip es muy interesante. Permite realizar una conversión entre los datos de una UART (Universal asynchronous receiver transmitter = Receptor Transmisor Asincrónico Universal) y la codificación IrDA, que es uno de los formatos de datos de los controles remotos de infrarrojos, ver **Figura** 2.10.

Los datos de una UART estándar (dentro de un microcontrolador, por ejemplo) se ingresan a este chip y se convierten en pulsos para enviar a un emisor de infrarrojos. Los datos recibidos por un receptor de infrarrojos como el que describimos antes se ingresa a este integrado y son convertidos a datos para una UART estándar. La velocidad de transferencia (baud rate) se

define con unas entradas del chip, que permiten seleccionar entre un amplio rango de velocidades (dependiendo también de la frecuencia de reloj que se aplica al circuito). Por ejemplo, con un reloj de 14,7456 MHz se obtienen velocidades de 19.200, 38.400, 78.600, 115.200 y 230.400 baudios. Este último sería el valor máximo de velocidad que se puede alcanzar.

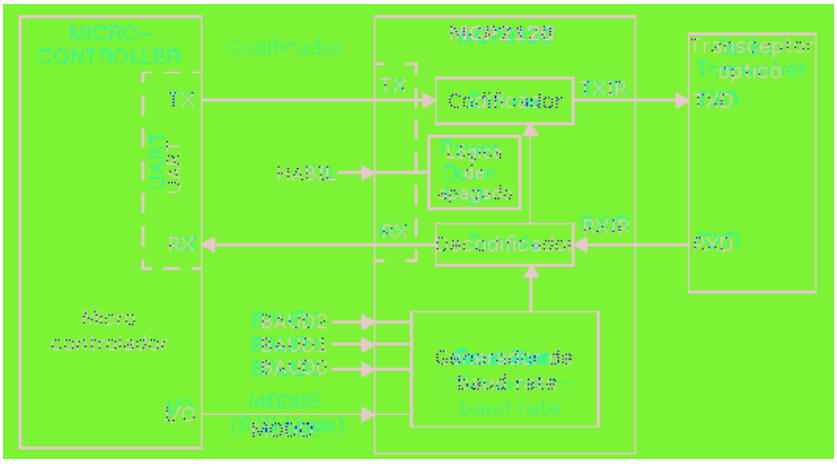



**Figura 2.24** *Circuito de aplicación del MCP2120 (La entrada **MODO** permite seleccionar un modo de prueba en el que los datos son retornados al microcontrolador)*

### 2.10.8 Módulos de transceptor para enlace infrarrojo

Se puede contar con unos módulos ya armados que permiten una comunicación bidireccional por infrarrojos. Son unas pequeñas plaquetitas, cosa que se observa en la **Figura** 2.12. Por lo que se lee en los datos de venta, trabajan con el estándar RC-5 de Philips para control remoto.

**Figura 2.24** *Juego de módulos bidireccionales de comunicación IR*

### 2.10.9 Módulos Tx-Rx por Radiofrecuencia (RF)

**Transmisor y Receptor TWS-# y RWS-#**

El módulo transmisor viene ya ajustado en una frecuencia, que puede ser de 303,875 MHz (**TWS-303**), 315 MHz (**TWS-315**), 418 MHz (**TWS-418**) y 433,92 MHz (**TWS-433**). Está listo para su uso. Sólo se debe colocar una antena, conectarle la alimentación y comenzar a enviarle datos.

Para facilitar la transmisión de datos codificados, existe un codificador que hace juego, que es el Holtek **HT12E** Pinescon componente metálico hacia delante.

Pin 1: Vcc
Pin 2: Vcc
Pin 3: GND
Pin 4: GND
Pin 5: Salida de RF
Pin 6: Entrada de datos

Potencia de salida de RF: 8 mW

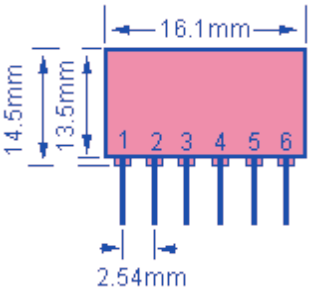
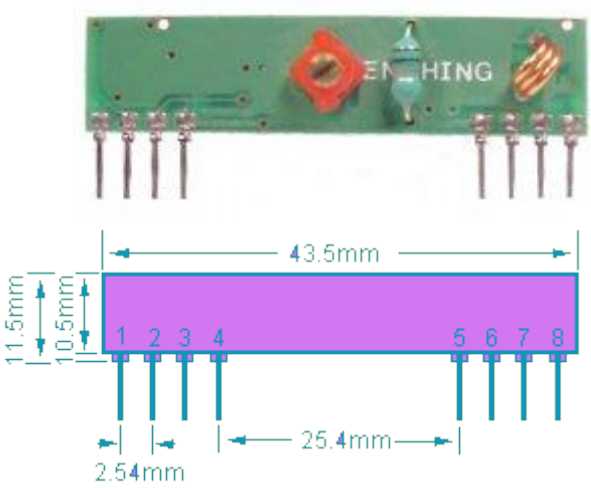




**Figura 2.25** *Transmisor de datos codificados*

| ESPECIFICACIONES | | | | | | |
|---|---|---|---|---|---|---|
| **Símbolo** | **Características** | **Condiciones** | **Mín.** | **Tip.** | **Máx.** | **Unidad** |
| $V_{cc}$ | Voltaje de alimentación | | 1,5 | - | 12 | V |
| $I_{cc}$ | Corriente máxima | | - | 5 | 9 | mA |
| $V_{ih}$ | Voltaje máximo entrada | $I_{dato}$=100µA (alto) | $V_{cc}$-0,5 | - | $V_{cc}$ | V |
| $V_{il}$ | Voltaje mínimo entrada | $I_{dato}$=0µA (bajo) | - | - | 0,3 | V |
| $P_{out}$ | Potencia RF sobre 50 ohm | | -3 | 0 | +2 | dBm |
| $T_{bw}$ | Ancho banda modulación | Codificación externa | - | 5 | - | kHz |
| $T_r$ | Flanco subida modulación | | - | - | 100 | µS |
| $T_f$ | Flanco bajada modulación | | - | - | 100 | µS |
| | Alcance | | | 20 | | m |

**Tabla 2.2** *Especificaciones técnicas del transmisor*

Los transmisores listados hacen juego con receptores de la misma frecuencia, que también vienen en un valor predeterminado entre 300 MHz a 434 MHz. Ver **Figura** 2.26. Puede ser de 303,875 MHz (**RWS-303**), 315 MHz (**RWS-315**), 418 MHz (**RWS-418**) y 433,92 MHz (**RWS-433**). Posee en diseño pasivo de alta sensibilidad, que no requiere componentes externos. Para decodificar las señales que llegan a este receptor se pueden utilizar el decodificador asociado Holtek **HT12D**.




Pines con el ajuste de sintonía hacia adelante (color naranja en la foto)

Pin 1: GND
Pin 2: Salida digital
Pin 3: Salida lineal
Pin 4: Vcc
Pin 5: Vcc
Pin 6: GND
Pin 7: GND
Pin 8: Antena (aprox 30-35cm)

Sensibilidad: 3uVrms

**Figura 2.26** *Receptor de datos codificados*

| ESPECIFICACIONES | | | | | | |
|---|---|---|---|---|---|---|
| **Símbolo** | **Características** | **Condiciones** | **Mín.** | **Tip.** | **Máx.** | **Unidad** |
| $V_{cc}$ | Voltaje de alimentación | | 4,9 | 5 | 5,1 | V |
| $I_{tot}$ | Corriente de operación | | | 4,5 | | mA |
| $V_{dato}$ | Salida datos | $I_{data}$=+200µA (alto) | $V_{cc}$-0,5 | - | $V_{cc}$ | V |
| | | $I_{dato}$=-10µA (bajo) | - | - | 0,3 | V |
| $F_c$ | Frecuencia operación | | 300 | | 434 | MHz |
| $P_{ref}$ | Sensibilidad | | | | -106 | dBm |
| | Ancho de canal | | ±500 | | | kHz |
| NEB | Ancho banda equival. ruido | | - | 5 | 4 | kHz |
| | Velocidad transferencia datos | | | | 3 | kb/s |
| | Tiempo de encendido | | - | - | 5 | mS |

**Tabla 2.3** *Especificaciones técnicas del receptor*

# CAPITULO III

# ENLACES

**PROTOCOLOS DE COMUNICACIÓN RS-232 Y RS-485**

## 3.1 Interfaz RS-232[5]

RS-232 (también conocido como Electronic Industries Alliance RS-232C) es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Data Communication Equipment, Equipo de terminación del circuito de datos), aunque existen otras situaciones en las que también se utiliza la interfaz RS-232.

En particular, existen ocasiones en que interesa conectar otro tipo de equipamientos, como pueden ser computadores. Evidentemente, en el caso de interconexión entre los mismos, se requerirá la conexión de un DTE (Data Terminal Equipment) con otro DTE.

El RS-232 consiste en un conector tipo DB-25 (de 25 pines), aunque es normal encontrar la versión de 9 pines (DB-9), más barato e incluso más extendido para cierto tipo de periféricos (como el ratón serie del PC).

La interfaz RS-232 está diseñada para distancias cortas, de unos 15 metros o menos, y para una velocidad de comunicación bajas, de no más de 20 Kb. A pesar de ello, muchas veces se utiliza a mayores velocidades con un resultado aceptable. La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal simplex, half duplex o full duplex. En un canal simplex los datos siempre viajarán en una dirección, por ejemplo desde DCE a DTE. En un canal half duplex, los datos pueden viajar en una u otra dirección, pero sólo durante un determinado periodo de tiempo; luego la línea debe ser conmutada antes que los datos puedan viajar en la otra dirección. En un canal full duplex, los datos pueden viajar en ambos sentidos simultáneamente. Las líneas de handshaking de la RS-232 se usan para

[5] Texto tomado de es.wikipedia/wiki/rs232

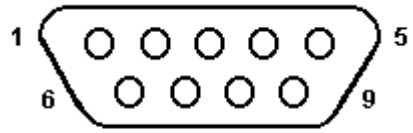
resolver los problemas asociados con este modo de operación, tal como en qué dirección los datos deben viajar en un instante determinado.

Si un dispositivo de los que están conectados a una interfaz RS-232 procesa los datos a una velocidad menor de la que los recibe deben de conectarse las líneas handshaking que permiten realizar un control de flujo tal que al dispositivo más lento le de tiempo de procesar la información. Las líneas de "hand shaking" que permiten hacer este control de flujo son las líneas RTS y CTS. Los diseñadores del estándar no concibieron estas líneas para que funcionen de este modo, pero dada su utilidad en cada interfaz posterior se incluye este modo de uso.

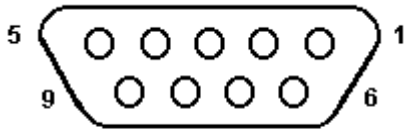
Las UART o U(S)ART (Transmisor y Receptor [Síncrono] Asíncrono Universal) se diseñaron para convertir las señales que maneja la CPU y transmitirlas al exterior. Las UART deben resolver problemas tales como la conversión de voltajes internos del DCE con respecto al DTE, gobernar las señales de control, y realizar la transformación desde el bus de datos de señales en paralelo a serie y viceversa. Debe ser robusta y deberá tolerar circuitos abiertos, cortocircuitos y escritura simultánea sobre un mismo pin, entre otras consideraciones. Es en la UART en donde se implementa la interfaz.
Para los propósitos de la RS-232 estandar, una conexión es definida por un cable desde un dispositivo al otro. Hay 25 conexiones en la especificación completa, pero es muy probable que se encuentren menos de la mitad de éstas en una interfaz determinada. La causa es simple, una interfaz full duplex puede obtenerse con solamente 3 cables.
Existe una cierta confusión asociada a los nombres de las señales utilizadas, principalmente porque hay tres convenios diferentes de denominación (nombre común, nombre asignado por la EIA, y nombre asignado por el CCITT).

En la siguiente tabla se muestran los tres nombres junto al número de pin del conector al que está asignado (los nombres de señal están desde el punto de vista del DTE (por ejemplo para Transmit Data los datos son enviados por el DTE, pero recibidos por el DCE):

**Conectores - RS232**

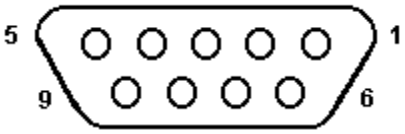



DB9 hembra - lado soldadura

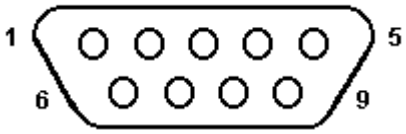



DB9 hembra - lado enchufar

**Pin 1 - Data Carrier Detect (DCD)**
**Pin 2 - Received Data (RD)**
**Pin 3 - Transmit Data (TD)**
**Pin 4 - Data Terminal Ready (DTR)**
**Pin 5 - Signal Ground (SG)**
**Pin 6 - Data Set Ready (DSR)**
**Pin 7 - Request To Send (RTS)**
**Pin 8 - Clear To Send (CTS)**
**Pin 9 - Ring Indicator (RI)**




DB9 macho - lado soldadura




DB9 macho - lado enchufar

**Figura 3.1** *Distribución de pines en formato DB9*

**Conectores-RS232**

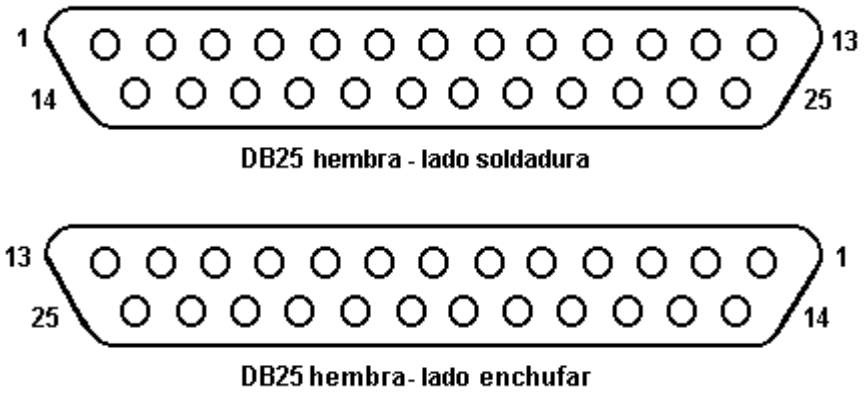



**Figura 3.2** *Distribución de pines en formato DB25 hembra*

Pin 1 - Shield Ground (GND)

Pin 2 - Transmit Data (TD)

Pin 3 - Received Data (RD)

Pin 4 - Request To Send (RTS)

Pin 5 - Clear To Send (CTS)

Pin 6 - Data Set Ready (DSR)

Pin 7 - Signal Ground (SG)

Pin 8 - Data Carrier Detect (DCD)

Pin 9 - Reservado

Pin 10 - Reservado

Pin 11 - No asignado

Pin 12 - Secondary Received Line Signal Detect

Pin 13 - Secondary CTS

Pin 14 - Secondary Transmited Data

Pin 15 - Transmiter Signal Timing

Pin 16 - Secondary Received Data

Pin 17 - Receiver Signal Timing

Pin 18 - Local Loopback

Pin 19 - Secondary RTS

Pin 20 - Data Terminal Ready (DTR)

Pin 21 - Remote Loopback

Pin 22 - Ring Indicator (RI)

Pin 23 - Data signal Rate Selector

Pin 24 - Transmiter Signal Timing

Pin 25 - Test Mode

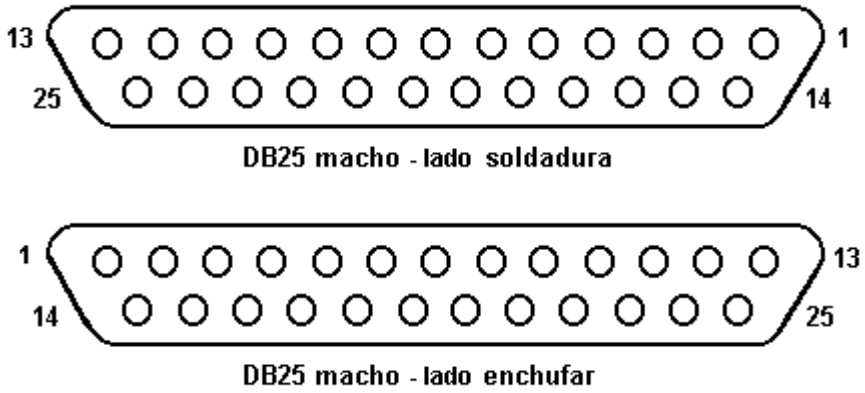



**Figura 3.3** *Distribución de pines en formato DB25 macho*

Por supuesto, existen muchos microcontroladores que integran entre sus periféricos una UART. En estos casos es mucho más sencilla la comunicación con el PC o con otros periféricos, pues la UART se encarga de casi todo. El problema surge cuando el micro que usamos no tiene el hardware adecuado (por problemas de espacio, de costo, etc.). En esos casos, es posible utilizar UART's hardware externas, pero que presentan el problema de tener que usar más hardware (encarecimiento, volumen...), por lo que quizá tampoco sea una solución. Para estas situaciones (u otras que se puedan presentar) se tiene la posibilidad de diseñar la UART e implementarla en software.

**Comunicación serie:**

En este caso se usa en el puerto serie de nuestro PC. Serie nos indica uno detrás de otro, y en efecto, cuando se transmite algo en serie, se lo hace bit a bit. En el caso del PIC, si queremos transmitir algo, hemos de usar uno de los pines configurado como salida. Para transmitir un '1' lógico, se pone el pin en nivel alto (+5V por ejemplo) y si se quiere enviar un '0', se pone el pin en nivel bajo (cerca del nivel GND). Así, variando en el tiempo el estado del pin, podemos se puede enviar todos los datos que se quiera. Así pues, se tiene diferentes velocidades 'estándar': 600, 1200, 2400...etc.

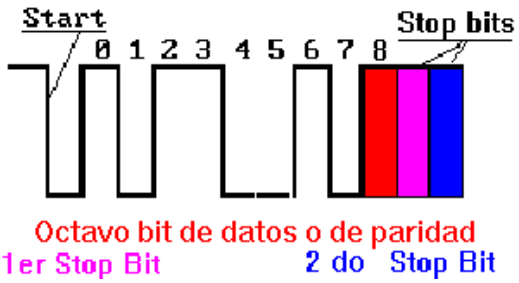



**Figura 3.4** *Comunicación serie*

S observa que si cada bit tiene una duración fija, constante, no es necesario un reloj que marque los cambios de bit. Cierto, no es necesario, es implícito. Pero esto implica que si se comete un error en la transmisión de un bit, el resto de la comunicación sería basura. Existen por tanto ciertos métodos y bits bandera para que esto no ocurra.

En primer lugar, el estado del pin de envío mientras no se está enviando es fijo (debe ser '1'). Así, cuando se empieza la transmisión, se envía un bit de inicio (que es un '0'). Después, se acuerda entre emisor y receptor cuantos bits van a conformar los datos: 5, 6, 7 u 8. Lo normal es 8 (un byte). Se envían respetando los tiempos de cada bit. Una vez enviados los datos, el siguiente bit es opcional y se usa como mecanismo para evitar errores: la paridad. Consiste en un bit que indica si el número de unos o ceros en los datos es par o impar.

Para terminar, se envía uno o dos bits de parada, de forma que el pin de transmisión se quede en el nivel lógico '1'. Así, se termina la transmisión del dato. Nos quedamos con la velocidad (9600 bps), los bits de datos (8bits), la paridad (N) y el bit de stop (1): 9600 8N1 **(que significa, 8 bits de datos, sin paridad y con 1 bit de Stop)**.

## 3.2 Comunicación RS-232 mediante integrado MAX232 (Conexión entre un Pic y una PC):

El circuito integrado MAX232 es un circuito especializado que genera los voltajes que necesita el estándar RS232 (+12 y -12). Usando este chip, se obtiene un mayor rechazo de ruido y una gran capacidad para soportar

cortocircuitos y descargas estáticas. En general, cuando se puede, es mejor usar este chip o uno similar (hay varios fabricantes).

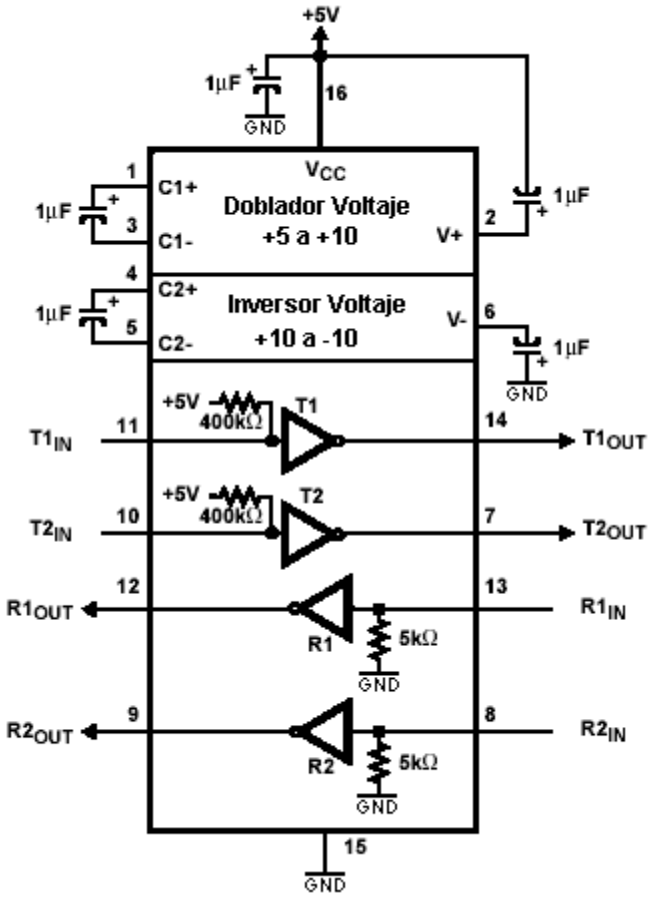



**Figura 3.5** *Conexionado del MAX232*

### 3.2.1 Integrado MAX232

Los circuitos con letra T son "Transmitters", que trasladan nivel TTL/CMOS en su entrada a nivel RS232 en su salida. Los circuitos con letra R son "Receivers", que trasladan señales RS232 en su entrada (que pueden ser de hasta +30/-30 V) a nivel TTL/CMOS en su salida.

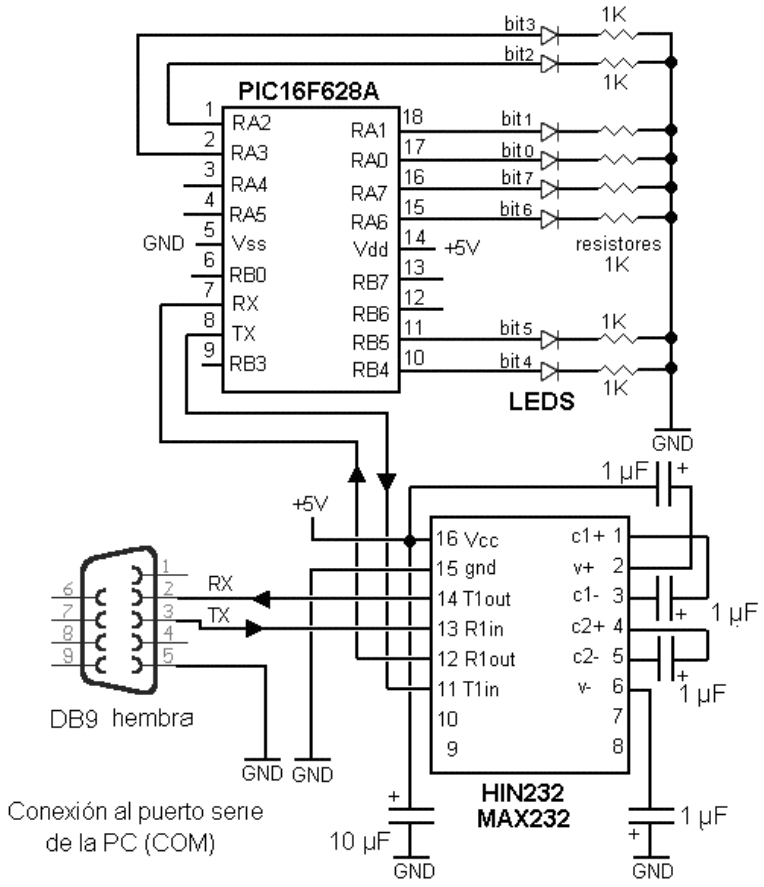



**Figura 3.6** *Circuito del interfaz de comunicación RS232 entre un PIC (16F628A) y una PC*.

## 3.3 Interfaz RS-485[6]

La interfaz RS485 ha sido desarrollada - analógicamente a la interfaz RS422 - para la transmisión en serie de datos de alta velocidad a grandes distancias y encuentra creciente aplicación en el sector industrial. Pero mientras que la RS422 sólo permite la conexión unidireccional de hasta 10 receptores en un transmisor, la RS485 está concebida como sistema Bus bidireccional con hasta 32 participantes. Físicamente las dos interfaces sólo se diferencian mínimamente. El Bus RS485 puede instalarse tanto como sistema de 2 hilos o de 4 hilos.

Dado que varios transmisores trabajan en una línea común, tiene que garantizarse con un protocolo que en todo momento esté activo como máximo

[6] Texto tomado de http:// electronics.alternatezone.com

un transmisor de datos. Los otros transmisores tienen que encontrarse en ese momento en estado ultraohmio.
La norma RS485 define solamente las especificaciones eléctricas para receptores y transmisores de diferencia en sistemas de bus digitales. La norma ISO 8482 estandariza además adicionalmente la topología de cableado con una longitud máxima de 500 metros.

En esta interfaz se utiliza una conexión balanceada sin conector físico. Con lo que se consigue mejorar la velocidad y distancia máxima.

Características principales:

- Velocidad máxima de 100Kbps hasta 1200m y de 10Mbps hasta 12m.
- Señales de cómo máximo 6V y de cómo mínimo 200mV.
- Amplificadores de triple estado, permiten interconectar hasta 64 dispositivos.
- El uso de tensiones elevadas de hasta 15V en RS-232 y de circuitos no balanceados hace que sea más susceptible al ruido.
  En cambio en RS-485 se utilizan voltajes de cómo máximo 6V y circuitos balanceados por lo que se reduce el factor de ruido.
- Con RS-485 se permiten conectar hasta 64 dispositivos.

RS485 sólo especifica características eléctricas de una unidad, pero no especifica o recomienda ningún protocolo de datos.

RS-485 soporta distintos tipos de conectores como DB-9 y DB-37.

Algunos usos del RS-485:

- SCSI-2 y SCSI-3 emplean esta especificación para implementar la capa física.
- RS-486 a menudo es usado en UARTs para implementar comunicaciones de datos a baja velocidad en cabinas de aviones comerciales.

- También es empleado en los edificios inteligentes.
- RS486 también es usado para controlar luces de discos y de teatros, donde es conocido como DMX.

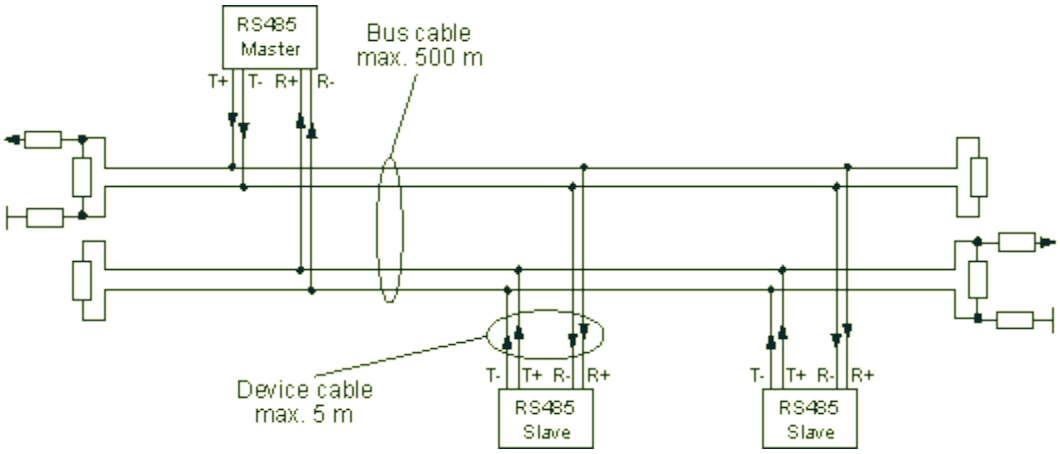



**Figura 3.7** *Red de Pic´s con conexión RS485*

El RS485 tiene la característica de que se puede alcanzar hasta 1.2 km de distancia con un baud rate de hasta 115200 bps.
La comunicación en RS485 puede ser utilizada en configuraciones de:

a) PUNTO A PUNTO (Pic a Pic)

b) PUNTO A MULTIPUNTO

Es decir se puede tener varios dispositivos colocados en el mismo par de hilos (como el I2C). Puedes tener hasta 256 elementos o dispositivos colgados en la red RS485. Esto depende del transceiver que se use, es decir existen algunos que tiene solamente 1/8 UL que son los que puede usar hasta 256. Existen de 1/4 de UL, 1/2 de UL y 1 UL.
Esto lo que equivale es a que se colocan menos dispositivos en la red, es decir; si tu transceiver tiene:

a) 1 UL se puede colocar solo 32 elementos

b) 1/2 UL se puede colocar 64 elementos

c) 1/4 UL se puede colocar 128 elementos

d) 1/8 UL se puede colocar 256 elementos

Existen también algunos que están limitados a un baud rate, existen otros que son full-duplex (4 hilos) o bien half-duplex (2 hilos).

El siguiente diseño permite comunicar dos Pic's mediante una interfaz Rs-485:

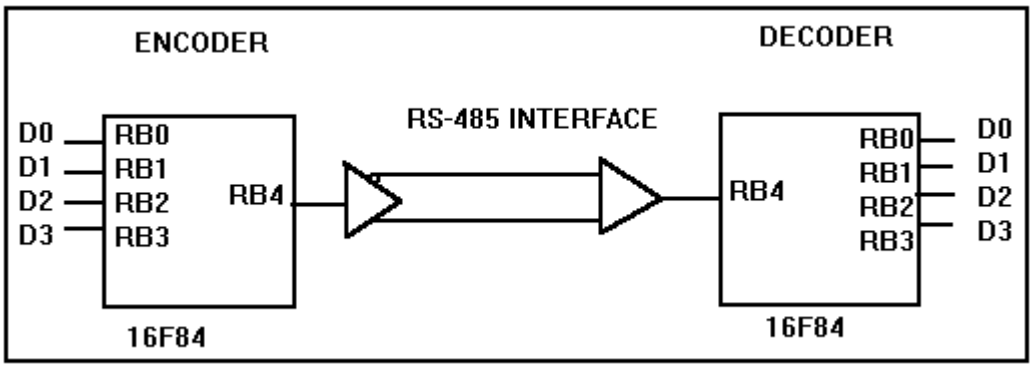



**Figura 3.8** *Diagrama de Bloques de un ejemplo con Interfaz Rs-485 con Pic 16F84*

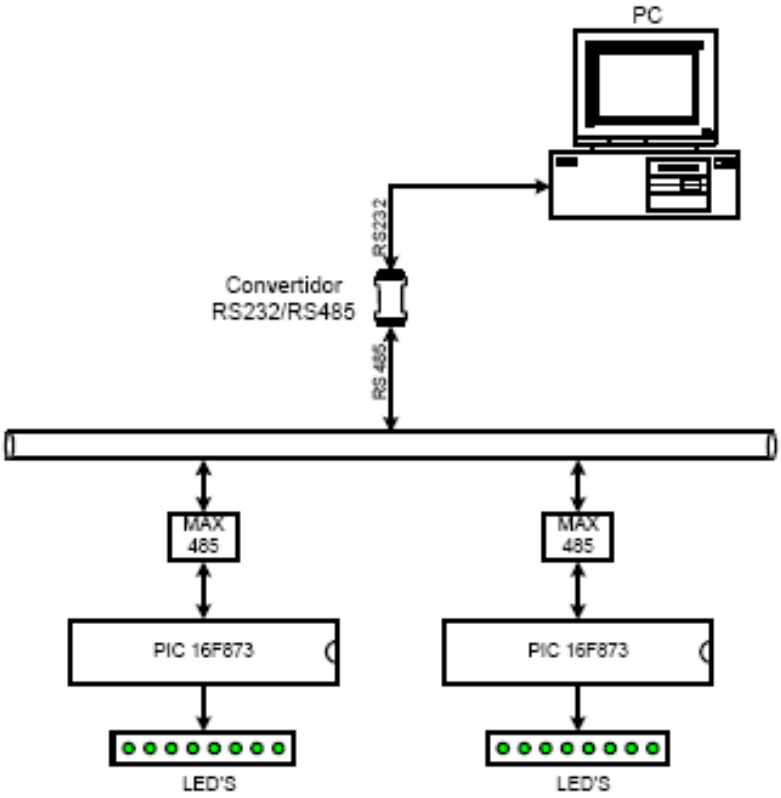



**Figura 3.9** *Sistema utilizando Pic 16F873 con Interfaz Rs-485 y CI MAX485*

## 3.4 TUTORIAL DE MPLAB 6.60 (Microchip)[7]

**Introducción.**

El MPLAB corre bajo Microsoft Windows 3.1 o superior. MPLAB proporciona funciones que permiten:

· Crear y Editar archivos fuentes

· Agrupar archivos fuente dentro de proyectos

· Depurar código fuente

[7] Texto tomado de http://www.simupic.com

### 3.4.1 Herramientas de MPLAB

MPLAB es un conjunto de herramientas para el desarrollo y depuración de aplicaciones en un proyecto. MPLAB incluye un editor de texto, un manejador de proyectos para mantener tu código organizado y un ambiente para depurar el software que desarrollas para tus propios diseños. Este ambiente incluye el simulador MPLAB-SIM, y soporta otras herramientas de Microchip tales como el emulador PICMASTER y el programador de dispositivos PICSTART Plus.

### 3.4.2 Manejador de Proyectos

El manejador de proyectos es la parte central del MPLAB. Sin la creación de un proyecto no se puede hacer depuración alguna. A través del manejador de proyectos se hacen las siguientes operaciones:

· Crear un proyecto
· Añadir un archivo de código fuente a un proyecto
· Ensamblar o compilar código fuente
· Editar código fuente
· Reconstruir todos los archivos fuente, o compilar un archivo sencillo
· Depurar el código fuente

### 3.4.3 El Editor MPLAB

El Editor de MPLAB permite a los programadores escribir y editar código fuente para las familias de microcontroladores PIC16/17, así como otros archivos de texto.

### 3.4.4 El Ensamblador MPASM

El Ensamblador Universal de Microcontroladores PIC16/17 MPASM ofrece grandes características completamente desarrolladas, ensamblado condicional y de diferentes fuentes y lista de formatos. MPASM permite generar varios formatos de código objeto que soportan las herramientas de

desarrollo de Microchip así como los programadores relacionados sin salir de MPLAB.

### 3.4.5 El Simulador MPLAB-SIM

El simulador MPLAB-SIM permite aislar problemas de código y depurar diseños en los microcontroladores PIC16/17. Simula las funciones principales así como la mayoría de los periféricos de las familias de microcontroladores PIC16/17.

**Otras Herramientas**

El MPLAB soporta herramientas de desarrollo tales como programadores, compiladores y emuladores ya sea de Microchip o de otros diseñadores.

### 3.4.6 Requerimientos De Software Y Hardware

Relativo al software el MPLAB funciona correctamente en un ambiente operativo de Windows, ya que soporta el entorno de multitareas. Debido a que MPLAB tiene capacidades de emulación de multiprocesador, soporta DDE (intercambio dinámico de datos) con programas cliente, los datos recolectados con MPLAB pueden ser intercambiados con más programas.
Para tomar ventaja de las características de emulación del sistema, se debe instalar el software de MPLAB en una computadora con la siguiente configuración (mínimo):

- PC 386 o superior. Se recomienda Pentium
- 4 MB de memoria en RAM, 16MB recomendado
- 8 MB de espacio libre en disco duro, 20 MB recomendado
- Monitor VGA o Súper VGA
- Microsoft Windows 3.1 o superior

### 3.4.7 Funciones De Mplab

Después de ajustar y compilar un proyecto en MPLAB, querrás ver como corre el código. Si se cuenta con un programador, se puede programar un microcontrolador y conectarlo en la aplicación actual para verificar que la aplicación funciona como se esperaba. Comúnmente, una aplicación no funciona correctamente la primera vez, y se tendrá que depurar el código. Se puede usar el MPLAB-SIM para simular el código o se puede usar el emulador PICMASTER para correr tu aplicación en la aplicación actual mientras lo depuras.

De otra manera, se pueden usar puntos de ruptura para ver como corre el código. Observar los valores de los registros en la ventana de Registros o la ventana de Registros de Funciones Especiales para ver el estado del procesador tal y como funcionaria tu código paso por paso.

El emulador PICMASTER corre el código en tiempo real en tu hardware objetivo, deteniéndose solamente en los puntos de ruptura que se hayan especificado. El MPLAB-SIM simula la ejecución de cualquier PIC16/17 y simula las condiciones de Entrada/Salida mas la velocidad depende de la velocidad de la PC en que se ejecuta.

Las siguientes funciones de depuración trabajan igual con el simulador o el emulador. Las funciones principales son:

· Emulación de la memoria (ventana Memoria de Programa)

· Puntos de Ruptura

· Paso-a-Paso

· Monitoreo de Registros (SFR y File Register)

Todas estas funciones utilizan información de un proyecto de MPLAB.

**Ejecución en Tiempo-Real**

La ejecución en Tiempo-Real solo es aplicable al emulador PICMASTER.

### 3.4.8 Ejecución en el Modo simulador MPLAB-SIM

Cuando se le dice al sistema que corre en tiempo real en el modo simulador, las instrucciones se ejecutan tan rápido como es posible por el software. Esto

comúnmente es mas lento que lo que en realidad podrían corren los microcontroladores PIC16/17 en su ciclo de reloj.

La velocidad a la cual el simulador corre depende de la velocidad de la computadora en la que corre y de las demás tareas que se estén ejecutando al mismo tiempo. El software simulador debe de actualizar todos los registros simulados en RAM, también monitorear las Entradas/Salidas, ajustar y limpiar las banderas, checar y buscar por puntos de ruptura en el software, y simular las instrucciones de los microcontroladores PIC16/17 tal y como están siendo ejecutadas en el CPU de la computadora.

Nota: Rutinas de tiempo pueden ser utilizadas en el código para generar retardos de tiempo. Cuando se utiliza el simulador, se podría decrementar el tiempo de esos retardos o condicionarlos removiendo esas secciones del código con el fin de incrementar la velocidad de simulación.

En general cuando este manual dice “tiempo real” y se esta en modo de simulador, quiere decir que la simulación del software del código de los PIC16/17 es ejecutada simulada tan rápido como la PC puede simular las instrucciones.

### 3.4.9 Modo Animado

El modo animado es un método en que el procesador camina automáticamente paso-paso. El simulador ejecuta pasos sencillos mientras corre, pero solamente actualiza los valores de los registros cuando es detenido. Para ir observando los cambios en las ventanas de Registros de Funciones Especiales y Archivos de Registros se utiliza en modo animado. El modo animado corre más lento que la función RUN, pero permite ver los cambios en los valores de los registros.

### 3.4.10 Ambiente del Simulador MPLAB-SIM

El MPLAB-SIM es un simulador de eventos discretos para las familias de microcontroladores PIC16/17 y se encuentra integrado al MPLAB IDE. La herramienta de simulador MPLAB-SIM esta diseñada para:

· Modelar las operaciones de las familias de microcontroladores PIC16C5X, PIC16CXX y PIC17CXX de Microchip Technology

· Asistir al usuario en la depuración del software que utilizan los microcontroladores de Microchip.

Un simulador de eventos discretos, así como un emulador in-circuit como el emulador PICMASTER, están diseñados para depurar software. El MPLAB-SIM permite modificar código objeto e inmediatamente re-ejecutarlo, inyectarle estímulos externos al procesador simulado, y trazar la ejecución del código objeto. Un simulado difiere de un emulador in-circuit en tres áreas importantes:

· Tiempo de las Entradas/Salidas

· Velocidad de ejecución

· Costo

### 3.4.11 Tiempo de las Entradas/Salidas

El tiempo en el MPLAB-SIM es procesado únicamente durante cada ciclo de instrucción. Las señales transitorias tales como los impulsos al MCLR, mas pequeñas que los ciclos de instrucción no pueden ser simuladas en un simulador pero si en un emulador in-circuit.

### 3.4.12 Velocidad de Ejecución

La velocidad de ejecución de un evento discreto en un software simulador es de acuerdo a la magnitud y a la orientación de la resolución del hardware. El usuario podría ver la velocidad de la ejecución tan lenta o tan rápida como la herramienta lo permita. MPLAB-SIM esta diseñado para proporcionar la simulación de ciclos los mas rápida posible, y depende de los modos de operación, puede operar en el orden de los milisegundos por instrucción.

### 3.4.13 Costo

La tecnología de Microchip ha desarrollado el simulador MPLAB-SIM para el ser la herramienta para depuración de software de aplicaciones más efectiva por su costo. MPLAB-SIM no requiere ningún tipo de software externo a la PC, y opera en la mayoría de sus funciones exactamente igual al emulador PICMASTER. A menos que se necesite depurar la aplicación en tiempo real

en el hardware actual, MPLAB-SIM puede ser utilizado para buscar y corregir la mayoría de los errores de codificación.

## 3.5 Herramienta de Depuración

MPLAB-SIM esta particularmente catalogado para la optimización de algoritmos. A diferencia de un emulador, el simulador hace visibles cualquier registro interno o puede proporcionar herramientas de software que son difíciles o caras de implementar en hardware con un emulador in-circuit. Para la mayoría de los casos MPLAB-SIM puede ser utilizado para depurar completamente el sistema a menos que se corra en situaciones donde un emulador in-circuit es requerido.

## 3.6 Consideraciones del Simulador

De cualquier manera, existen algunos eventos físicos que no pueden ser simulados apropiadamente. Estos caen dentro de las siguientes categorías:

· Eventos puramente asíncronos

· Eventos que tienen periodos mas cortos que un ciclo de instrucción (1 ciclo de instrucción = 4 ciclos de reloj)

En realidad, la red resultante simula únicamente ciclos de instrucción de todos los eventos sincronizados, y todos los eventos mas pequeños que un ciclo de instrucción no son reconocidos.

Los siguientes son una lista de operaciones de las familias de microcontroladores PIC16/17 las cuales se ven afectadas en la simulación:

· Entradas de reloj más pequeñas que un ciclo de reloj utilizadas en el Prescaler

· Salidas PWM cuya resolución es menor que un ciclo de reloj

· Comparaciones mayores a 8 bits no se soportan

· En modo contador no sincronizado, las entradas más pequeñas a un ciclo de reloj no se pueden usar.

· Las formas de onda del oscilador en RC0/RC1 no pueden mostrarse.

## 3.7 Como Usar El Programa Mplab Y El Ensamblador Mpasm

### 3.7.1 Creación y desarrollo

MPLAB es un Entorno de Desarrollo Integrado (IDE) fácil de aprender y fácil de usar. La característica IDE proporciona a los desarrolladores de software para aplicaciones la flexibilidad para editar, compilar, emular, simular, desarrollar y depurar su propio software para las familias de microcontroladores PIC16/17 de Microchip.

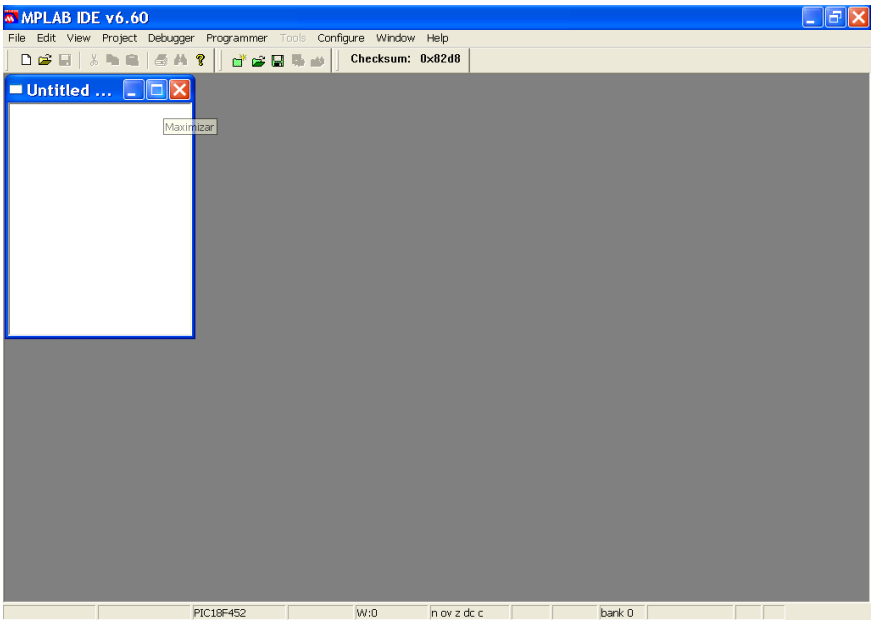



**Figura 3.10** *Pantalla de inicio de MPLAB*

El programa MPLAB es un software que contiene un editor, un ensamblador, un emulador y un simulador, todos ellos integrados en el mismo ambiente. El editor nos sirve para escribir un nuevo programa o modificarlo, para empezar a trabajar en el editor se necesita abrir o crear un nuevo archivo fuente como a continuación se muestra un ejemplo:

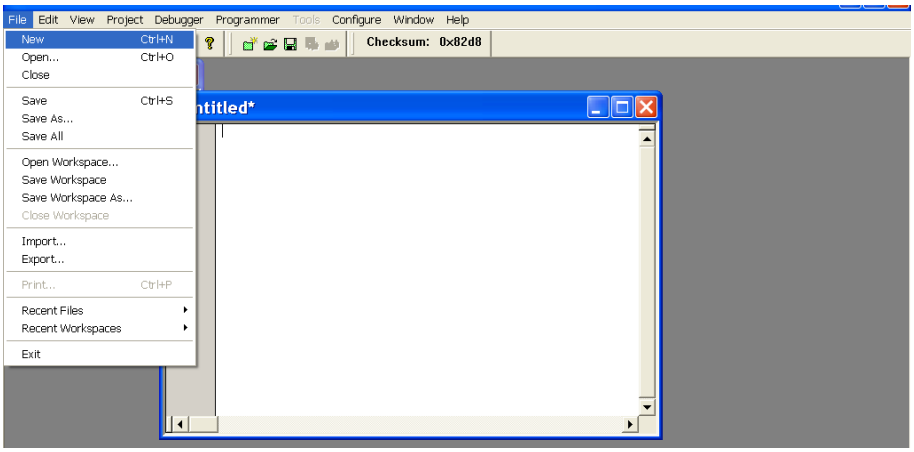



**Figura 3.11** *Editor de MPLAB*

Esta **figura 3.11** nos muestra la manera de como abrir un archivo mas la opción NEW es la adecuada para comenzar a realizarlo.

Una vez terminado de escribir el programa en el editor se procede a salvarlo (guardarlo) presionando CTRL+S, una vez ahí es tiempo de ponerle el nombre que uno desee, se recomienda que sea adecuado al programa o aplicación que se esta realizando, después de esto se procede a ejecutar el programa MPASM para pode ensamblar el archivo que acabamos de guardar/crear.

Utilizar el Asistente para proyecto nuevo n el Menú Project:

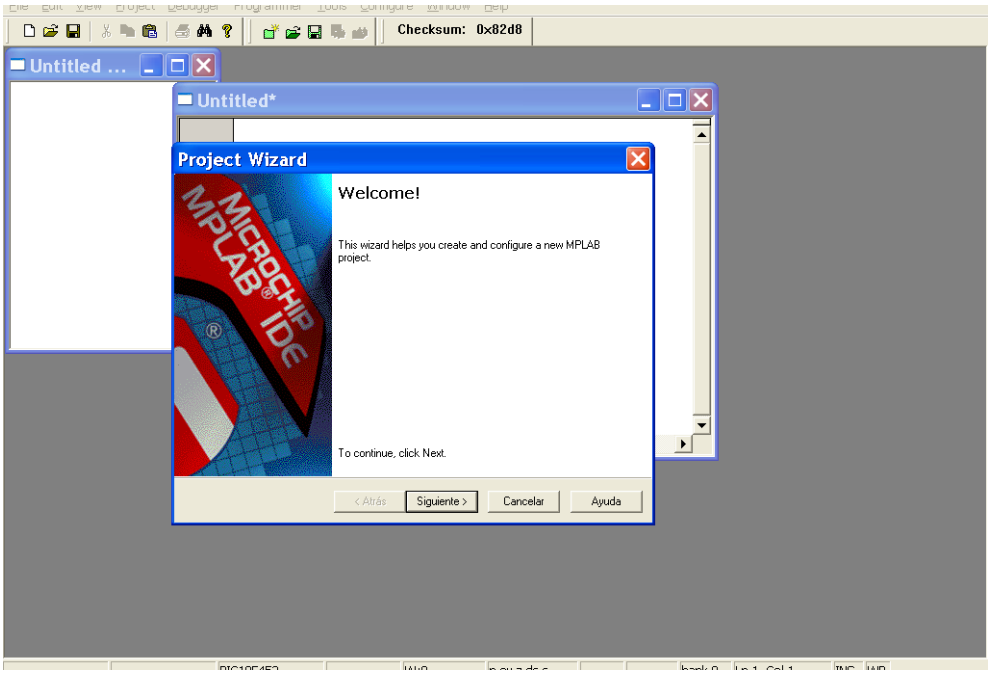


**Figura 3.12** *Creación de un nuevo proyecto*

Luego en uno delos pasos debemos utilizar el programa MPASM que es un ensamblador para microcontroladores PIC de Microchip. Para comenzar a ensamblar se localiza con el mouse/teclado el botón que dice Browse y lo presionamos para localizar/indicar nuestro archivo que vamos a ensamblar (*.ASM).

Una vez que ya localizamos el archivo buscaremos el numero/modelo de PIC que usaremos para nuestra aplicación o para simularlo, una vez establecido el microcontrolador PIC que se utilizara se procede a ensamblar el archivo presionando el botón Assembler y aparecerá el siguiente cuadro:

Cuando el momento de ensamblar no hay errores se obtendrá una pantalla parecida a esta:


**Figura 3.13** *Pantalla de ensamblaje aceptado.*

Este cuadro nos muestra los errores, warnings (advertencias), mensajes y líneas ensambladas, si no se tuvo ningún error el siguiente paso es simular y si no se procede a corregir los errores auxiliándose del archivo .ERR, que es generado por MPASM y se puede leer en el editor de MPLAB o en cualquier otro editor de texto, que nos muestra la línea en que nos equivocamos.

Para esto lo podemos abrir como cualquier otro archivo con el que se desea trabajar, indicado anteriormente, este archivo generalmente es generado junto con otros más con extensión diferente, por ejemplo. COD, .HEX, y se localizan en el mismo directorio o ubicación que el archivo fuente.

Para comenzar a simular el archivo después de haber sido correctamente ensamblado se procede a bajar el archivo .HEX del archivo que queremos simular a la memoria del simulador.

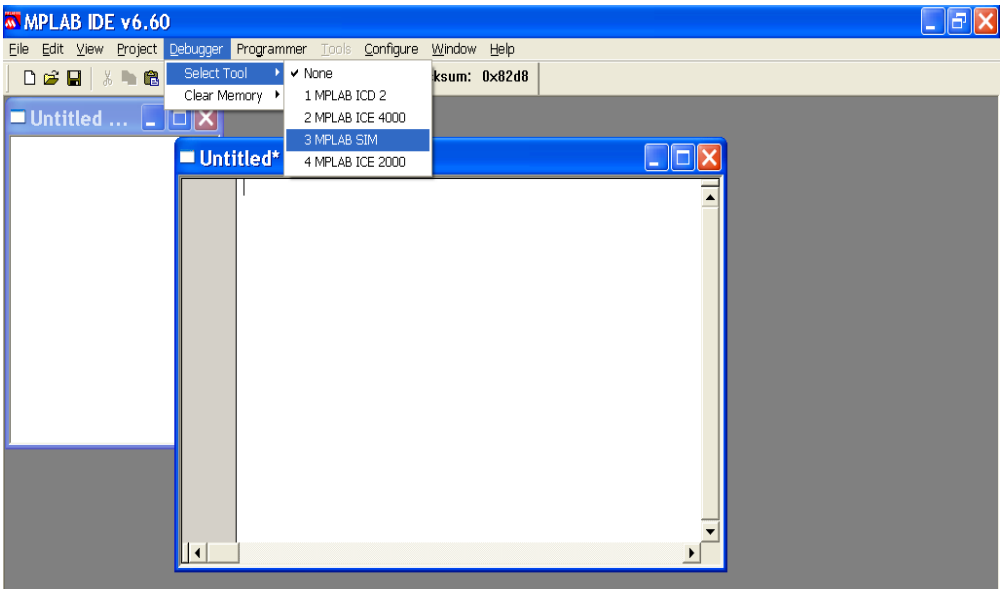



**Figura 3.14** *Pantalla de selección herramientas*

El siguiente paso es abrir una ventana con el programa ensamblado para poder simularlo.

También abrir Menú View: Watch

Menú: Debugger: Stimulus Controller

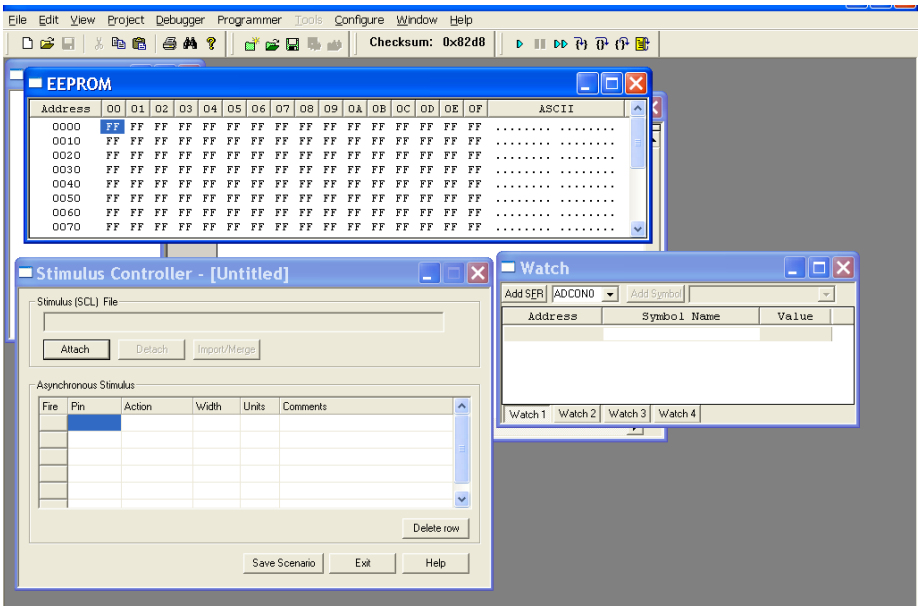



**Figura 3.15** *Pantallas de estímulos y visor*

Para poder simular en este programa es necesario cambiar la configuración de modo Editor a modo Simulador, encontrar tu modelo de microcontrolador PIC con el que se va a simular y apretar RESET.

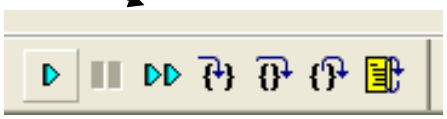


**Figura 3.16** *Herramientas*

El siguiente paso es comenzar a simular nuestro programa, primero tendremos que mover el mouse hacia DEBUG y se verán todas las barras de dicha tarea:

Ahora explicare para que sirve cada barra, para correr un programa automáticamente presionaremos ANIMATE, para detenerlo esta HALT y el RESET para inicializar nuestro programa hasta PC 0x00 (inicio del contador de programa). STEP es para ir simulando paso a paso cada instrucción.

Para resetear el programa .HEX que esta en memoria en caso de que se desee simular otro o de que al programa existente se le haya hecho algún cambio y se haya ensamblado de nuevo, se utiliza CLEAR PROGRAM

MEMORY y para limpiar todos los registros de memoria y registros especiales como: PORTB, PORTA, W(acumulador),TIMERS, etc., se utiliza CLEAR ALL POINTS.

A continuación se presenta una imagen con los registros especiales registro de memoria:

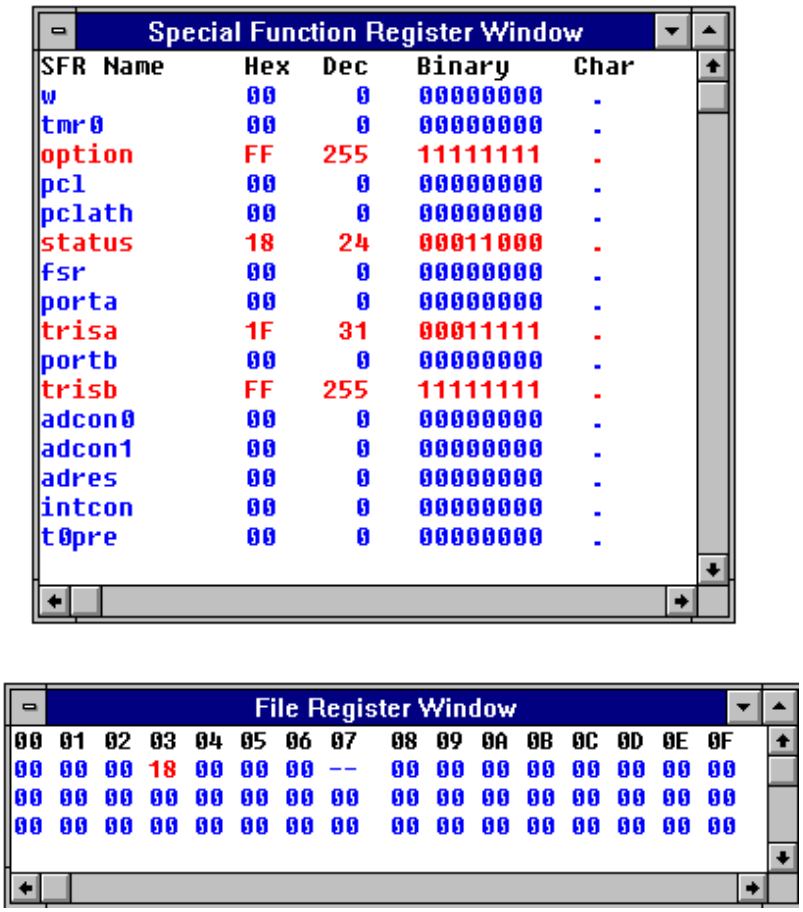



**Figura 3.17** *Ventanas de funciones especiales y archivos de registro*

**Modificar.**

Seleccione Window>Modify para desplegar o modificar los contenidos de la Memoria de Datos, Memoria de Programas, la Pila o la memoria EEPROM.

Modificar permite leer/escribir a una dirección especifica, leer/escribir mientras se esta incrementando a la siguiente dirección o llenar un bloque de direcciones. MPLAB permite dejar abierta la ventana de Modificar todo el tiempo. Existen cuatro maneras para abrir el cuadro de dialogo Modificar:

· Seleccionar Window>Modify

· Doble clic en un elemento en la ventana de registros de funciones especiales

· Doble clic en un elemento de la ventana Watch

· Seleccionar una dirección o un rango en la ventana File Register y dar un clic en el botón derecho del mouse para desplegar el botón Fill Register. Presione el botón Fill Register para desplegar el cuadro de dialogo Modify.


**Figura 3.18** *Elementos del cuadro de dialogo Modify*

Address: Introduce la dirección en la cual el dato va a ser leído o modificado. Se puede introducir una dirección numérica o un símbolo (etiqueta)

Data/Opcode: Presione Read para desplegar el valor del dato/Opcode de la dirección seleccionada y el área de memoria. Presione en Write para escribir el valor del dato/Opcode de la dirección seleccionada y área de memoria en hexadecimal, decimal o mnemónico.

Memory Area: Seleccione el área de memoria que se quiera modificar

Data Memory: Memoria RAM

Program Memory: Memoria ROM en el emulador

Stack: Memoria de la Pila del dispositivo

EEPROM: Datos de la memoria EEPROM

End Adr: La dirección final para llenar el rango

Fill Range: Llena el rango definido por las dos direcciones con el valor introducido en Data/Opcode

Auto Increment: Seleccione auto increment para incrementar a la siguiente dirección después de cada lectura/escritura

Nota: Auto Increment avanza a la siguiente dirección, despliega la siguiente dirección y lee el contenido de la dirección. Si se esta utilizando Auto Increment para leer un rango, introduzca la dirección del área de memoria menos uno, debido a que la primer lectura incrementara la dirección.

Write: Introduce un nuevo en el campo Data/Opcode, presione Write para modificar el dato de la dirección especifica. (Se pueden introducir datos en formato simbólico) Cuando un dato es modificado, todas las ventanas que lo utilizan se actualizan con la nueva información

Read: Presione Read para leer el dato de la dirección especificada

Close: Presione Close para salir de Modificar.

Tutorial De ICProg 1.05c

**Introducción.**[8]

La instalación es sencilla, ya que basta descomprimir el archivo en el directorio que se quiera y ejecutar el programa directamente pues esta compilado de forma estática por lo que no requiere de ningún archivo ni librería adicional si se está usando los sistemas operativos Windows 9X o Me, si se esta usando Windows NT, Windows 2000 o Windows XP que será es necesario descargase el driver "IC-Prog NT/2000 driver" que podemos obtener de la misma página. También se puede obtener el fichero de ayuda, este se tendrá que guardar en el mismo directorio que el programa.

[8] Texto y software tomado de http://www.ic-prog.com/.

### 3.7.2 Configuración.

### 3.7.3 Programador.

La primera vez que ejecutemos el programa se nos pedirá que configuremos el tipo de programador que vamos a usar, en nuestro caso deberemos seleccionar la opción ProPic 2 Programmer.

Después deberemos configurar el tipo de interfaz para ello seleccionaremos Direct I/O si usamos Windows 9x o Windows Me, si por el contrario usamos Windows NT, Windows 2000 o Windows XP seleccionaremos Windows API, en los parámetros de comunicación activaremos las opciones Invertir MCLR e Invertir VCC.

En cuanto al tiempo de retardo, si tuviésemos problemas se pueden probar con tiempos más largos. En ordenadores de última generación estos tiempos se podrán reducir, ahorrando tiempo en el grabado.

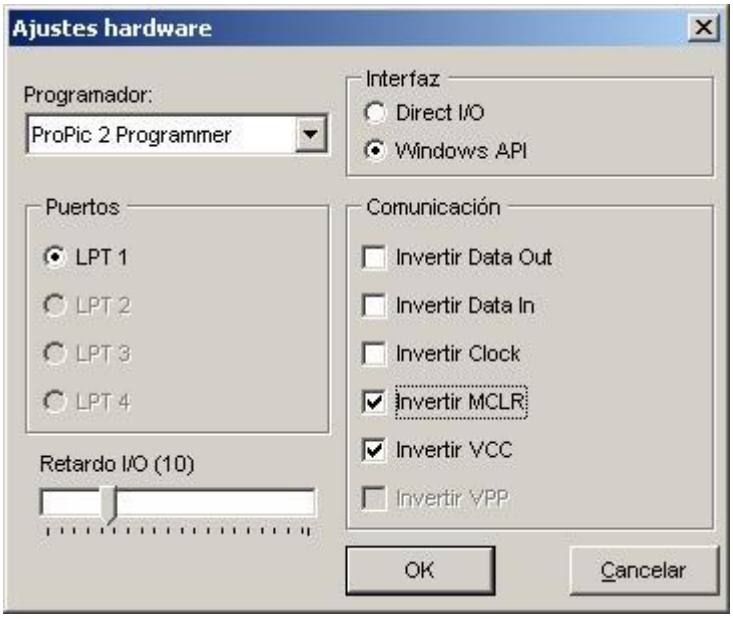



**Figura 3.19** *Ajustes de hardware ICPROG*

### 3.7.4 Programación

En el menú Ajustes / Opciones seleccionamos la pestaña Programación en la que desactivaremos las opciones de verificación de la programación. Ver figura 3

En el caso de trabajar con los sistemas operativos Windows NT, Windows 2000 o Windows XP, se deberá descargar el driver en el mismo directorio que el programa y en la pestaña Miscelánea activar la opción Habilitar Driver NT/2000. En sistemas operativos anteriores a estos la opción esta desactivada al no ser necesaria.

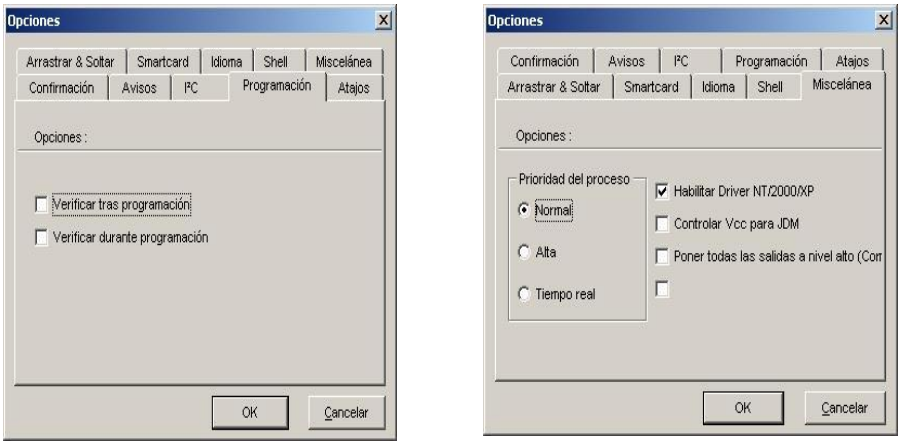



**Figura 3.20** *Ventanas de Ajustes*

## 3.8 Grabación de un PIC 16F87X

El proceso de grabación lo podemos esquematizar en los siguientes pasos:

- Seleccionar el dispositivo a programar PIC 16F876A.
- Abrir el archivo que contiene los datos a programar en el PIC.
- Ajustar la palabra de configuración y el tipo de oscilador.
- Programar el dispositivo.

### 3.8.1 Seleccionar el dispositivo

En el desplegable de la barra de herramientas tenemos todos los dispositivos que es capaz de grabar el IC-Prog, en el seleccionaremos el PIC 16F876A.

### 3.8.2 Abrir el archivo (* .hex)

En el menú Archivo seleccionamos Abrir archivo en el cuadro de diálogo que nos aparece seleccionamos el fichero que deseamos grabar en el PIC.

### 3.8.3 Ajuste de los bits de configuración

Los tipos de oscilador que se pueden elegir son (RC, LP, XT, HS). En Protección de código, podemos seleccionar la protección de todo el código, por páginas o no proteger el código.

El resto de los bits de configuración se pueden activar directamente, normalmente los tendremos desactivados.

Si al ensamblar o compilar el archivo fuente se activaron los bits de configuración, cuando se carga el archivo .hex se marcara los bits seleccionados automáticamente.

### 3.8.4 Programar el dispositivo

Lo único que nos resta es programarlo, para ello seleccionamos Programar todo del menú Comando. Con ello comenzará la grabación.

**PicBasic PRO**[9]

Acerca De Este Tutorial

Este manual no es un tratado completo del lenguaje BASIC. Describe el conjunto de instrucciones del micro PIC y brinda ejemplos de cómo utilizarlo. Si no está familiarizado con la programación de BASIC, deberá obtener un libro sobre dicho tema. O intentarlo directamente. BASIC está diseñado como un lenguaje fácil de utilizar.

Introducción

El compilador PicBasic Pro (PBP) es nuestro lenguaje de programación de nueva generación que hace más fácil y rápido para usted programar microcontroladores Pic de **Microchip Technology**.

El lenguaje Basic es mucho más fácil de leer y escribir que el lenguaje ensamblador Microchip.

[9] Texto tomado de http://www.todopic.com.ar

PBP por defecto crea archivos que corren en un PIC 16F84-04/P con un reloj de 4 Mhz. Solamente muy pocas partes son necesarias capacitores de dos capacitores de 22 pf para el cristal de 4Mhz un resistor de 4.7K en el pin/MCLR y una fuente de 5 voltios. Otros micros PIC además del 16F84, así como otros osciladores de frecuencias distintas pueden ser usados por este compilador.

## Los Micros

El PBP produce código que puede ser programado para una variedad de micro controladores PIC que tengan de 8 a 68 pines y varias opciones en el chip incluyendo convertidores A/D, temporizadores y puertos seriales.

El PIC16F870 además, tiene 128 bytes de memoria de datos no volátil que puede ser usada para archivar el dato de programa y otros parámetros, aun cuando no haya energía. A ésta área de datos, se puede acceder simplemente usando las órdenes “Read” y “Write” del PBP. (El código programa es permanentemente guardado en el espacio de código del micro PIC, tanto si hay o no energía.)

## Su Primer Programa

Para operar el PBP, necesitará un editor ó procesador de texto para crear su programa fuente, algún tipo de programador de micros PIC como el propio PBP. Por supuesto, también necesita un PC:

El siguiente programa provee un buen primer testeo de un micro PIC en el mundo real. Puede tipearlo o simplemente obtenerlo del subdirectorio SAMPLES. El archivo fuente BASIC debe ser creado o movido al mismo directorio donde se encuentra el archivo PBP.EXE

Ejemplo de programa para hacer parpadear un LED conectado al puerto PORTB.0, aproximadamente una vez por segundo ´

**loop: high PORTB.0** ´ enciende el LED

**pause 500** ´ demora de .5 segundos

**low PORTB.0** ´ apaga el LED

**pause 500** ´ demora de .5 segundos

**goto loop** ´ vuelve a loop y hace parpadear el LED indefinidamente

**end**

El compilador mostrará un mensaje de inicialización y procesará su archivo. Si lo acepta, creará un archivo de código fuente ensamblado (en este caso **PARPADEO.ASM**) y automáticamente invocará al ensamblador para completar la tarea. Si todo funciona bien, se crea un archivo de código microPIC (en este caso PARPADEO.HEX). Si existen errores, se emitirá un listado de los mismos, que deberán ser corregidos en su archivo fuente BASIC antes de ser compilados nuevamente.

> Recomendación: Para ayudarlo a asegurarse que su archivo original funcione sin errores, es mejor comenzar escribiendo y probando pequeñas partes de su programa y no escribir 100000 líneas de programa y luego tratar de depurarlas de principio a fin.

Si Ud, no le indica otra cosa, el PBP, por defecto, **crea código para el PIC16F84**. Para compilar códigos para otros micros PIC, simplemente especifique otro tipo de procesador.

## Programando El Micro

Hay otros dos pasos colocar su programa compilado dentro del micro controlador Picmicro y testearlo.

El PBP genera archivos de 8 bit standard de INTEL (.HEX) que pueden ser usados con cualquier programador de micros Pic incluyendo nuestro programador Universal Device Serial Programmer (ICPROG)

El siguiente es un ejemplo de cómo un microPic puede ser programado usando nuestro programador ICPROG.

Asegúrese de que no haya microsPic instalados en el zócalo de programación del programador ICPROG.

Conecte el programador ICPROG al puerto serial de del PC usando un cable de DB9 macho a DB9 hembra.

El LED en el programador puede estar encendido o apagado en este momento. No coloque un microPic en el zócalo de programación cuando el LED esté encendido o antes de que el software de programación dé comienzo.

El esquema de ejemplo siguiente da una idea de los pocos elementos que se necesitan conectar a un microPic para hacerlo funcionar. Básicamente se necesita un resistor de pull-up en la línea /MCLR, un cristal de 4 Mhz con 2 capacitores y una fuente de 5 voltios. Hemos agregado un LED y un resistor para proveer la salida para el programa BLINK.

Construya y verifique este simple circuito en una tarjeta y enchufe el microPic que usted programó. Nuestra tarjeta electrónica “Entradas – Salidas“ del banco de trabajo es perfecta para este tipo de práctica.

Conecte una fuente de alimentación su microPIC arrancará parpadeando el LED aproximadamente una vez por segundo. Si no lo hace verifique todas las conexiones y asegúrese que hay 5 voltios en los pines apropiados de su microPic

Con este simple comienzo, puede crear su propio mundo de aplicaciones.

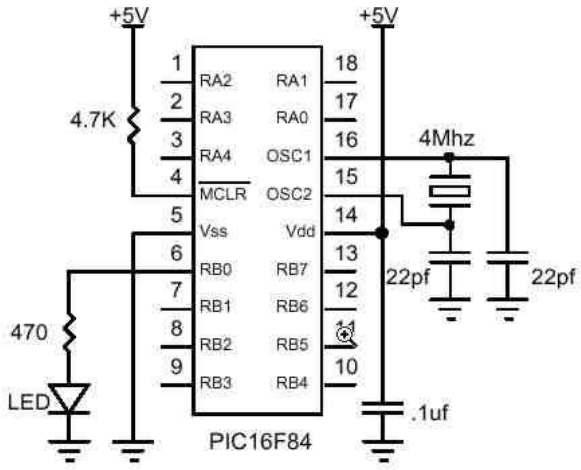



**Figura 3.21** *Esquema de conexión para el programa “PARPADEO”*

### 3.8.5 Recomendaciones Para Una Correcta Programación En Basic

A continuación se indica algunas recomendaciones para una correcta programación en BASIC

### 3.8.6 Nombres De Pin Y De Variable

Haga que los nombres sean algo más coherente que Pin0 o B1. Además de un uso libre de comentarios, nombres descriptivos ayuda mucho a la legibilidad. El siguiente fragmento lo demuestra:

**BattLED var portb.0** ´ LED de batería baja

**Nivel var byte** ´ la variable contiene el nivel de batería

**If Nivel< 10 then** ´ si el nivel de batería es bajo

**High battLED** ´ enciende el LED

**Endif**

### 3.8.7 Etiquetas

Las etiquetas (labels) deben indicar algo más significativo que “etiqueta 1” ó “aquí”. Aún una etiqueta “loop” es más descriptiva (pero poco).Usualmente la línea ó rutina a la que se está saltando hace algo único. Trate de dar un indicio de su función con la etiqueta y luego siga con un comentario.

### 3.8.8 Goto

Trate de no usar demasiados **GOTO**. Aunque pueden ser un mal necesario, trate de minimizar su uso en lo posible. Trate de escribir su código en secciones lógicas y no ir saltando a cualquier lado. Usar **GOSUB** puede ser útil para esto.

### 3.8.9 Identificadores

Un identificador es simplemente un nombre. Son usados en PBP como etiquetas de líneas y nombres de variables. Un identificador es cualquier secuencia de letras, dígitos y símbolos, aunque no deben comenzar con un dígito. Los identificadores no distinguen las letras mayúsculas de las minúsculas, por lo que **etiqueta**, **ETIQUETA**, **EtiquEtA**, son todas tratadas

como equivalentes. Aunque las etiquetas pueden tener cualquier número de caracteres de longitud PBP solamente reconoce los primeros 32.

### 3.8.10 Etiquetas De Línea (Labels)

Para marcar líneas que el programa puede desear referenciar con comandos **GOTO** ó **GOSUB**, PBP usa etiquetas de línea. PBP no permite número de línea y no requiere que cada línea sea etiquetada. Cualquier línea PBP puede comenzar con una etiqueta de línea que es simplemente un identificador seguido por un punto y coma (;)

**mostrar: Serout 0, N2400, [“Hello, World!”, 13, 10]**

**Goto mostrar**

**VARIABLES**

Variables es donde se guardan datos en forma temporaria en un programa PBP. Son creadas usando la palabra clave VAR. Pueden bits, bytes ó word. Espacio para cada variable es automáticamente destinado en la memoria del micro controlador por PBP. El formato para crear una variable es el siguiente:

**Etiqueta VAR tamaño (.modificadores)**

Etiqueta es cualquier identificador excluyendo palabras claves como se describe anteriormente. Tamaño es bit, byte ó word. Modificadores opcionales agregan control adicional acerca de cómo se crea la variable. Algunos ejemplos de creación de variables son:

**perro var byte**

**gato var bit**

**W0 var word**

No hay variables predefinidas de usuarios de PBP.

El número de variables disponibles depende de la cantidad de RAM en un dispositivo en particular y el tamaño de las variables y los arrays. PBP reserva aproximadamente 24 posiciones RAM para su propio uso. También puede

crear variables temporarias adicionales para usar en ordenamiento de ecuaciones complejas.

**Etiqueta VAR tamaño (número de elementos)**

Etiqueta es cualquier identificador, excluyendo palabras claves, como se describió anteriormente. Tamaño es BIT, BYTE ó WORD. Número de elementos es cuantos lugares en el arreglo se desean. Algunos ejemplos de creación de arreglo son los siguientes:

**sharks var byte[10]**

**fish var bit [8]**

La primera ubicación dentro del arreglo es el elemento cero. En el arreglo fish anterior los elementos están numerados fish (0) a fish (7) conteniendo 8 elementos en total.

Dada la forma en que los arreglos están localizados en memoria hay límites de tamaño para cada tipo.

| Tamaño | Número máximo de elementos |
|---|---|
| BIT | 128 |
| BYTE | 64 |
| WORD | 32 |

**Tabla 3.1** *Tamaño de localidades de memoria*

### 3.8.11 Constantes

Las llamadas constantes pueden ser creadas de manera similar a las variables. Puede ser más conveniente usar un nombre de constante en lugar de un número constante. Si el número necesita ser cambiado, únicamente puede ser cambiando en un lugar del programa donde se define la constante. No pueden guardarse datos variables dentro de una constante.

**Etiqueta CON expresión constante**

Algunos ejemplos son:

**Mice con 3**

**Traps con mice *1000**

### 3.8.12 Constantes Numéricas

PBP permite definir constantes numéricas en tres bases: decimal, binario y hexadecimal. Valores binarios son definidos usando el prefijo “%” y valores hexadecimales usando el prefijo “$”. Los valores decimales se toman por defecto y no requieren prefijo.

´ valor decimal 100

**%100 ´ valor binario para el decimal 4.**

**$100 ´ valor hexadecimal para el decimal 256.**

Para facilitar la programación, los caracteres son convertidos en sus equivalentes ASCII. La constante debe ser puesta entre comillas y contener sólo un carácter (de lo contrario, ellas son una cadena de constantes).

**“A” ‘ ASCII valor para el decimal 65**

**“d” ‘ ASCII valor para el decimal 100**

**CADENA DE CONSTANTES**: (string)

PBP no provee capacidad de manejo de cadenas, pero las cadenas pueden ser usadas con algunos comandos. Una cadena contiene uno o más caracteres y es delimitado entre comillas. No se soportan secuencias de escape para caracteres no-ASCII (aunque, la mayoría de los comandos PBP tienen este manejo incorporado)

**“Hello” ‘ String (forma abreviada de “H”, “e”, “l”, “l”, “o”)**

Las cadenas son usualmente tratadas como una lista de valores de caracteres individuales.

### 3.8.13 Puertos Y Otros Registros

Todos los registros inclusive los puertos del PICmicro MCU, pueden ser accedidos como cualquier otra variable en PicBasic. Esto significa que pueden ser leídos, ser escritos o ser utilizados en ecuaciones directamente:

**PORTA = %01010101**. ' Escribe el valor en el PUERTO A
**PORTA anyvar = PORTB y $0F**. ' Aísle los 4 dígitos binarios bajos de PORTB y ponga el resultado en anyvar

### 3.8.14 Pins

A los pines se puede acceder de diferentes modos. El mejor camino para especificar un pin para una operación, es simplemente usar sus nombres **PORT** y un número de bit:

**PORTB.1= ‘ Colocar PORTB, bit 1 a 1**

Para recordar fácilmente para qué puede ser usado un pin, debe asignarse un nombre usando el comando **VAR**. De esta manera, el nombre puede ser utilizado luego en cualquier operación:

**Led var PORTA.O ‘ Renombra PORTA.O como led**

**High led ‘ Coloca led (PORTA.O) en valor alto**

Para colocar un pin o port como salida (ó entrada) debe dar valores al registro **TRIS**. Colocando el bit de **TRIS** como **0**, hace su pin una **salida**, y colocándolo en **1** lo hace una **entrada**. Por ejemplo:

**TRISA = %00000000 ´ O TRISA = 0**

Coloca todos los pines PORTA como salidas.

**TRISB = % 11111111 ´ O TRISB = 1**

Coloca todos los pines PORTB como entradas.

**TRISC = % 10101010**

Coloca todos los pines pares como salidas y los impares como entradas.

Cada bit individual puede ser manejado de la misma manera

**TRISA.0 = 0**

Coloca el PORTA, pin 0 como salida. Todos los demás pin permanecen sin cambio.

### 3.8.15 Los Comentarios

Un comentario de PBP comienza con la palabra clave REM o el apóstrofe (´). Todos los demás caracteres de esa línea se ignoran.

REM es una única palabra clave y no es una abreviación de REMark, por lo tanto, los nombres de variables pueden comenzar con REM (aunque REM por sí mismo no es válido).

### 3.8.16 Declaraciones Múltiples

Para permitir programas más compactos y agrupamientos lógicos de comandos relacionados, PBP soporta el uso de (:) para separar comandos ubicados en la misma línea. Los siguientes dos ejemplos son equivalentes.

**W2 = W0**

**W0 = W1**

**W1 = W2**

Es lo mismo que:

**W2 = w0 : W0 = W1 : W1 = W2**

En los dos casos, el tamaño del código generado es el mismo.

### 3.8.17 Include

Se puede agregar archivos fuente BASIC a un programa PBP usando **INCLUDE**. Usted puede tener su rutina standard, definiciones u otros archivos que desee guardar en forma separada. Los archivos de definición de modo serial y de stamp son ejemplo de esta. Estos archivos pueden ser incluidos en programas donde ser necesario, pero no en programas donde no se los necesita.

Las líneas de código fuente del archivo incluido son insertadas dentro del programa exactamente donde se coloca el INCLUDE.

**INCLUDE “modedefs.bas”**

### 3.8.18 Define

Algunos elementos, como el oscilador y las ubicaciones de los pin LCD, están predefinidos en PBP. DEFINE le permite a un programa PBP cambiar estas definiciones si así lo desea.

Define puede ser usado para cambiar el valor predefinido del oscilador, los pines de DEBUG y el baud rate y las ubicaciones de los pin LCD además de otras cosas. Estas definiciones deben estar en mayúsculas

| | |
|---|---|
| DEFINE BUTTON_PAUSE 50 | demora en el anti-rebote del botón en ms |
| DEFINE CHAR_PACING 1000 | paso de la salida serial en us |
| DEFINE DEBUG_REG _PORTL | depuración del pin port |

| | |
|---|---|
| DEFINE DEBUG_BIT 0 | depuración del pin bit |
| DEFINE DEBUG_BAUD 2400 | depuración del baud rate |
| DEFINE DEBUG_MODE 1 | modo depuración: 0=CIERTO,1=INVERTIDO |
| DEFINE DEBUG_PACING 1000 | paso de depuración en us |
| DEFINE HSER_RCSTA 90 h | setear registro receive |
| DEFINE HSER_TXSTA 20 h | setear registro transmit |
| DEFINE HSER_BAUD 2400 | setear baud rate |
| DEFINE HSER_EVEN 1 | usar solo si se desea paridad par |
| DEFINE HSER_ODD 1 | usar solo si se desea paridad impar |
| DEFINE I2C_INTERNAL 1 | usar para EEPROM interno en 16CEXX y 12CEXX |
| DEFINE I2C_SLOW 1 | usar para OSC > 8 Mhz con dispositivos de velocidad standard |
| DEFINE LCD_DREG PORTB | port de data LCD |
| DEFINE LCD_DBIT 0 | datos LCD comenzando en bit 0 o 4 |
| DEFINE LCD_RSREG PORTB | port de selección de registro LCD |
| DEFINE LCD_RSBIT 4 | bit de selección de registro LCD |
| DEFINE LCD_EREG PORTB | port de habilitación LCD |
| DEFINE LCD_EBIT 5 | bit de habilitación LCD |

| | |
|---|---|
| DEFINE LCD_BITS 4 | bus del LCD de 4 u 8 bits |
| DEFINE LCD_LINES 2 | numero de líneas en LCD |
| DEFINE OSC 4 | 3 (3.58) 4 8 10 12 16 20 Mhz. |
| DEFINE OSCCAL_1K 1 | setea OSCCAL para PIC12C671 |
| DEFINE OSCCAL_2K 1 | setea OSCCAL para PIC12C672 |

**Tabla 3.2** *Opciones DEFINE*

## 3.9 Operadores Matemáticos

PBP efectuar todas las operaciones matemáticas en orden jerárquico. Esto significa que existe precedencia para los operadores. Multiplicación y división son efectuados antes que suma y resta, por ejemplo.. Para asegurarse que las operaciones son efectuadas en el orden que se desea, use paréntesis para agrupar las operaciones.

**A = ( B + C ) * ( D - E )**

Todas las operaciones matemáticas se realizan sin signo y con una precisión de 16 bit.

Los operadores soportados son:

| **Operador matemático** | **Descripción** |
|---|---|
| **+** | **Suma** |
| **-** | **Resta** |

| | |
|---|---|
| * | Multiplicación |
| ** | 16 bits superiores de la multiplicación |
| */ | 16 bits medios de la multiplicación |
| / | División |
| // | Resto (módulo) |
| << | Desplazamiento izquierdo |
| >> | Desplazamiento derecho |
| ABS | Valor absoluto |
| COS | Coseno |
| DCD | 2m decodificador |
| DIG | Digito |
| MAX | Máximo * |
| MIN | Mínimo * |
| NCD | Codificar |
| REV | Invertir bits |
| SIN | Seno |
| SQR | Raíz cuadrada |
| & | Bit inteligente AND |
| ÷ | Bit inteligente OR |
| ^ | Bit inteligente EXCLUSIVE OR |
| ~ | Bit inteligente NOT |
| & / | Bit inteligente NOT AND |
| ÷ / | Bit inteligente NOT OR |
| ^ / | Bit inteligente NOT EXCLUSIVE OR |

**Tabla 3.3** *Operadores matemáticos*

### 3.9.1 Operadores De Comparación

Se usan en declaraciones **IF ... THEN** para comparar una expresión con otra .Los operadores soportados son :

| Operador | Descripción |
|---|---|
| = o == | Igual |
| <> o != | No igual |
| < | Menor |
| > | Mayor |
| <= | Menor o igual |
| >= | Mayor o igual |

**Tabla 3.4** *Operadores de comparación*

**If i > 10 then loop**

### 3.9.2 Operadores Lógicos

Los operadores lógicos difieren de las operaciones de bit inteligente. Entregan un resultado CIERTO / FALSO de su operación .Valores 0 son tratados como falso. Cualquier otro valor es cierto. Se usan junto a operadores de comparación en una declaración IF .. THEN .Los operadores soportados son:

| Operador | Descripción |
|---|---|

| | |
|---|---|
| **AND o &&** | **AMD lógico** |
| **OR o I I** | **OR lógico** |
| **XOR o ^ ^** | **OR exclusivo lógico** |
| **NOT AND** | **NAND lógico** |
| **NOT OR** | **NOR lógico** |
| **NOT XOR** | **NXOR lógico** |

**Tabla 3.5** *Operadores lógicos*

**If (A == big) AND ( B > mean) then run**

Asegúrese de usar paréntesis para indicarle a PBP el orden en que quiere que se realicen las operaciones.

**REFERENCIA DE DECLARACIONES PBP**

| | |
|---|---|
| **@** | **Inserta una línea de código ensamblador** |
| **ASM...ENDASM** | **Inserta una sección de código ensamblador** |
| **BRANCH** | **GOTO computado(equiv. a ON..GOTO)** |

| | |
|---|---|
| **BRANCHL** | **BRANCH fuera de pagina(BRANCH largo)** |
| **BUTTON** | **Anti-rebote y auto-repetición de entrada en el pin especificado** |
| **CALL** | **Llamada a subrutina de ensamblador** |
| **CLEAR** | **Hace cero todas las variables** |
| **COUNT** | **Cuenta el numero de pulsos en un pin** |
| **DATA** | **Define el contenido inicial en un chip EEPROM** |
| **DEBUG** | **Señal asincrónica de salida en un pin fijo y baud** |
| **DISABLE** | **Deshabilita el procesamiento de ON INTERRUPT** |
| **DTMFOUT** | **Produce tonos en un pin** |
| **EEPROM** | **Define el contenido inicial en un chip EEPROM** |
| **ENABLE** | **Habilita el procesamiento de ON INTERRUPT** |
| **END** | **Detiene la ejecución e ingresa en modo de baja potencia** |
| **FOR...NEXT** | **Ejecuta declaraciones en forma repetitiva** |
| **FREQOUT** | **Produce hasta 2 frecuencias en un pin** |
| **GOSUB** | **Llama a una subrutina BASIC en la etiqueta especificada** |
| **GOTO** | **Continua la ejecución en la etiqueta especificada** |
| **HIGH** | **Hace alto la salida del pin** |

| | |
|---|---|
| **HSERIN** | **Entrada serial asincrónica(hardware)** |
| **HSEROUT** | **Salida serial asincrónica(hardware)** |
| **I2CREAD** | **Lee bytes de dispositivo I2C** |
| **I2CWRITE** | **Graba bytes en dispositivo I2C** |
| **IF..THEN..ELSE..ENDIF** | **Ejecuta declaraciones en forma condicional** |
| **INPUT** | **Convierte un pin en entrada** |
| **(LET)** | **Asigna el resultado de una expresión a una variable** |
| **LCDOUT** | **Muestra caracteres en LCD** |
| **LOOKDOWN** | **Busca un valor en una tabla de constantes** |
| **LOOKDOWN2** | **Busca un valor en una tabla de constantes o variables** |
| **LOOKUP** | **Obtiene un valor constante de una tabla** |
| **LOOKUP2** | **Obtiene un valor constante o variable de una tabla** |
| **LOW** | **Hace bajo la salida de un pin** |
| **NAP** | **Apaga el procesador por un corto periodo de tiempo** |
| **ON INTERRUPT** | **Ejecuta una subrutina BASIC en un interrupt** |
| **OUTPUT** | **Convierte un pin en salida** |
| **PAUSE** | **Demora (resolución 1mseg.)** |
| **PAUSEUS** | **Demora (resolución 1 useg.)** |
| **PEEK** | **Lee un byte del registro** |
| **POKE** | **Graba un byte en el registro** |

| | |
|---|---|
| **POT** | **Lee el potenciómetro en el pin especificado** |
| **PULSIN** | **Mide el ancho de pulso en un pin** |
| **PULSOUT** | **Genera pulso hacia un pin** |
| **PWM** | **Salida modulada en ancho de pulso a un pin** |
| **RANDOM** | **Genera numero pseudo-aleatorio** |
| **RCTIME** | **Mide el ancho de pulso en un pin** |
| **READ** | **Lee byte de un chip EEPROM** |
| **RESUME** | **Continua la ejecución después de una interrupción** |
| **RETURN** | **Continua en la declaración que sigue al ultimo GOSUB** |
| **REVERSE** | **Convierte un pin de salida en entrada o uno de entrada en salida** |
| **SERIN** | **Entrada serial asincrónica (tipo BS!)** |
| **SERIN2** | **Entrada serial asincrónica (tipo BS2)** |
| **SEROUT** | **Salida serial asincrónica (tipo BS1)** |
| **SEROUT2** | **Salida serial asincrónica (tipo BS2)** |
| **SHIFTIN** | **Entrada serial sincrónica** |
| **SHIFTOUT** | **Salida serial sincrónica** |
| **SLEEP** | **Apaga el procesador por un periodo de tiempo** |
| **SOUND** | **Genera un tono o ruido blanco en un pin** |
| **STOP** | **Detiene la ejecución del programa** |
| **SWAP** | **Intercambia los valores de dos variables** |
| **TOGGLE** | **Hace salida a un pin y cambia su estado** |

| | |
|---|---|
| **WHILE..WEND** | **Ejecuta declaraciones mientras la condición sea cierta** |
| **WRITE** | **Graba bytes a un chip EEPROM** |
| **XIN** | **Entrada X - 10** |
| **XOUT** | **Salida X - 10** |

**Tabla 3.6** *Declaraciones*

### 3.9.3 Funcionamiento de los comandos más utilizados

**ADCIN**

**ADCIN** Channel , Var

Lee el conversor analógico del micro y guarda el resultado en el Var. Mientras que los registros del ADC se pueden alcanzar directamente, ADCIN hace el proceso un poco más fácil.

Antes de que ADCIN pueda ser utilizado, el registro de TRIS se debe fijar como entradas. ADCON1 también necesita ser asignado como entradas de información analógicas y en algunos casos para fijar el formato del resultado y la fuente del reloj. Vea las hojas de datos del microchip para más información sobre estos registros y cómo fijarlos para el dispositivo específico.

Dependiendo del dispositivo, puede tener 8 -, 10 o 12-bit ADC. El bit alto de ADCON1 controla si el resultado está a la izquierda o a la derecha. En la mayoría de los casos, los resultados 8-bit se deben dejar alineados (ADCON1.7 = 0) y 10 y los resultados 12-bit justificados a la derecha (ADCON1.7 = 1).

Varios DEFINE pueden también ser utilizados. Los valores por defecto se muestran abajo:

DEFINE ADC_BITS 8 ' Fije el número de BITS en el resultado (8, 10 o 12)

DEFINE ADC_CLOCK 3 'Fije EL CLOCK (rc = 3)

DEFINE ADC_SAMPLEUS 50 ' Fije el tiempo de muestreo en microsegundos

ADC_SAMPLEUS es el número de microsegundos que el programa espera entre fijar el canal y comenzar la conversión analógica/digital.

TRISA = 255 ' Fije PORTA todas entradas

ADCON1 = 0 ' PORTA es analógico

ADCIN 0, B0 ' Lea el canal 0 a B0

**BUTTON** , Pin , Down , Delay , Rate , Bvar , Action , Etiqueta

Lee Pin y opcionalmente ejecuta anti-rebote y auto-repetición. Pin automáticamente se toma como entrada .Pin debe ser una constante, 0 - 15, o una variable que contenga un número 0 - 15 (p.ej. B0) ó un número de pin (p.ej. PORTA ,0)

| | |
|---|---|
| **Down** | **Estado del pin cuando se oprime el pulsador ( 0 ..1)** |
| **Delay** | **Contador de ciclos antes de que comience la auto-repetición(0..255). Si es 0, no se efectúa anti-rebote ni autorepetición .Si es 255 se eliminan rebotes, pero no auto-repetición.** |
| **Rate** | **Valor de auto-repetición (0..255)** |
| **Bvar** | **Variable con tamaño de byte usada internamente para conteo de demoras y repeticiones, Debe ser inicializada a 0 antes de ser usada** |

| | |
|---|---|
| | **y no ser usada en cualquier lugar del programa.** |
| **Action** | **Estado del pulsador a ser actuado.** |
| **Etiqueta** | **La ejecución comienza en esta etiqueta si es cierto Action.** |

**Tabla 3.7** *Opciones de botton*

´ goto notpressed if button not pressed on Pin2

**BUTTON PORTB ,2,0,100,10,b2,0,notpressed**

BUTTON necesita ser usado dentro de un loop para auto-repetición para funcionar adecuadamente.

BUTTON permite eliminar rebotes, demorando la ejecución de un programa por un período de milisegundos para permitir que los contactos se asienten .La demora por defecto es 10 ms. Para cambiarlo a otro valor use DEFINE.

´ setea la demora de anti-rebote a 50 ms

**DEFINE BUTTON_PAUSE 50**

**BUTTON_PAUSE debe estar en mayúsculas.**

En general, es más fácil leer el estado del pin con un IF..THEN que usar el comando BUTTON.

**IF PORTB,2 = 1 THEN notpressed**

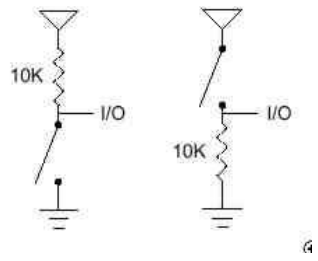



**Figura 3.22** *Formas de configurar un BUTTON físico*

**CALL etiqueta**

Ejecuta la subrutina ensamblador llamada etiqueta.

Normalmente se usa GOSUB para ejecutar una subrutina PBP .La principal diferencia entre GOSUB y CALL, es que con ésta última no se chequea la existencia de etiquetas hasta el momento de ensamblar .Usando CALL se puede acceder a una etiqueta en una sección de lenguaje ensamblador, lo que es inaccesible mediante PBP.

CALL pass ´ ejecuta la subrutina ensamblada ,denominada _pass

**COUNT Pin,Period,Var**

Cuenta el numero de pulsos en un Pin , durante un período periodo, y guarda el resultado en Var. Pin es automáticamente colocado como entrada .Pin debe ser una constante , 0-15 , ó una variable que contenga un número de 0 a 15 (p.ej. B0) .ó un numero de pin .

La resolución de periodo está dada en milisegundos.Sigue la frecuencia del oscilador basado en DEFINE OSC .

VCOUNT chequea el estado de Pin mediante un loop y cuenta las transiciones de bajo a alto .Con un oscilador de 4 Mhz chequea el estado del pin cada 20 us .Con un oscilador de 20 Mhz chequea el estado cada 4 us .De esto ,se infiere que la mayor frecuencia de pulsos que puede ser contada ,es de 25 Khz con un oscilador de 4 Mhz y de 125 Khz con un oscilador de 20 Mhz si la frecuencia tiene un ciclo útil del 50 % ( los tiempos altos son iguales a los bajos).

´ cuenta el número de pulsos en Pin1 en 100 ms

**COUNT PORTB.1,100,W1** ´ determinar la frecuencia en un Pin

**COUNT PORTA.2,1000,W1** contar por 1 segundo

**Serout PORTB.0,N2400, [ W1 ]**

**DTMFOUT**

**DTMFOUT** Pin, { Onms ,Offms,} [ Tone {,Tone} ]

Produce una secuencia DTMF Touch Tone en Pin, Pin automáticamente se convierte en salida. Pin debe ser una constante, 0 - 15, ó una variable que contenga un número de 0 a 15 (p.ej. B0) ó un número de pin (p.ej. B0)

Onms es el número de milisegundos que suena cada tono y Offms es el número de milisegundos de pausa entre cada tono .Si no están especificados, por defecto Onms es 200 ms y Offms es 50 ms.

Tones tiene un valor de 0 - 15 .Los tonos de 0 - 9 son los mismos que en un teclado telefónico .Tone 10 es la clave *, Tone 11 es la clave #, y los Tones 12 - 15 corresponden a las teclas extendidas A -D.

DTMFOUT usa **FREQOUT** para generar los tonos duales.FREQOUT genera tonos usando una forma de modulación de ancho de pulso. Los datos en bruto que salen del pin son bastante horribles. Usualmente se necesita algún tipo de filtro para suavizar la señal hasta una forma de onda senoidal quitándole algunas armónicas generadas:

DTMFOUT trabaja mejor con un oscilador de 20 Mhz. También puede trabajar con uno de 10 Mhz y aún con uno de 4 Mhz, aunque será muy difícil de filtrar y tendrá muy baja amplitud. Cualquier otra frecuencia causará que DTMFOUT genere una frecuencia proporcional al oscilador comparado a 20 Mhz, lo que no será muy útil para enviar touch tones.

´ enviar DTMF tones para 212 en Pin1

**DTMFOUT PORTB.1 , [ 2,1,2 ]**

**EEPROM**

**EEPROM** {Location ,} [ constante {,constante ...} ]

Guarda constantes en un chip EEPROM. Si se omite el valor opcional Location, la primera declaración se guarda en la dirección 0 del EEPROM y las subsiguientes en las siguientes direcciones del mismo. Si se indica un valor Location, éste indica la dirección de comienzo para guardar los datos.

Constante puede ser una constante numérica ó una cadena de constantes. Solo se guardan los bytes menos significativos de los valores numéricos. Las cadenas son guardadas como bytes consecutivos d valores ASCII. No se agregan automáticamente terminadores, ni se completa el largo.

EEPROM solo trabaja con micro controladores con EEPROM incorporado como el PIC16F84 y PIC16C84. Dado que el EEPROM es una memoria no volátil, los datos permanecerán intactos aún sin alimentación.

Los datos son guardados en el EEPROM solo una vez, cuando el micro controlador es programado, no cada vez que se ejecuta el programa. Se puede usar **WRITE** para colocar valores en el EEPROM en el momento de la ejecución.

´ Guardar 10 ,20 , 30 comenzando en la dirección 5

**EEPROM 5, [ 10,20,30 ]**

**END**

Detiene la ejecución del proceso y entra en modo de baja potencia .Todos los pines de I/O permanecen en el estado en que se encuentran, END trabaja ejecutando una instrucción **SLEEP** continua dentro de un loop.

Un END, STOP ó GOTO deben ser colocados al final de un programa para evitar pasar del límite de la misma u comience nuevamente .END

**FOR .. NEXT**

**FOR** Count = Start **TO** End {**STEP** {-} Inc}

{Body}

**NEXT** {Count}

El loop FOR .. NEXT permite a los programas ejecutar un número de declaraciones ( Body ) un número de veces, usando una variable como contador. Debido a su complejidad y versatilidad, es mejor describirla paso a paso.

El valor de Start se asigna a la variable índice, Count, que puede ser una variable de cualquier tipo.

Se ejecuta el Body. Body es opcional y puede ser omitido (quizás por un loop de demora).

El valor de Inc es sumado a (ó restado si se especifica "-") Counr. Si no se define una cláusula STEP, se incrementa Count en uno.

Si Count no pasó End ó desbordó el tipo de variable, la ejecución vuelve al paso 2).

Si el loop necesita contar más de 255 (Count > 255), se debe usar una variable de tamaño word.

**FOR i=1 TO 10** ´ cuenta de 1 a 10

Serout 0,N2400, [ **#** i," "] ´ envía cada número al pin0 en forma serial

**NEXT i** ´ vuelve y efectúa la próxima cuenta

Serout 0,N2400, [ 10 ] ´ envía un avance de línea

**FOR** B2=20 **TO** 10 **STEP** -2 ´ cuenta de 20 a 10 de a 2

Serout 0,N2400, [ **#** B2 , " "] ´ envía cada número al pin0 en forma serial

**NEXT** B2 ´ vuelve y efectúa la próxima cuenta

Serout 0,N2500, , [ 10 ] ´ envía un avance de línea

**FREQOUT**

**FREQOUT** Pin,Onms,Frequency1{,Frequency2}

Produce la ó las frecuencias especificadas en el Pin, durante milisegundos.Pin se convierte automáticamente en salida.Pin puede ser una constante, 0-15, ó una variable que contenga un número 0 - 15.(p.ej. B0) ó un número de pin (p.ej. PORTA.0).

Puede producir una ó dos frecuencias de 0 a 32767 Hz al mismo tiempo.

FREQOUT genera tonos usando una forma de modulación de ancho de pulso. Los datos en bruto que salen del pin son bastante horribles. Usualmente se necesita algún tipo de filtro para suavizar la señal hasta una forma de onda senoidal quitándole algunas armónicas generadas:

FREQOUT trabaja mejor con un oscilador de 20 Mhz. También puede trabajar con uno de 10 Mhz y aún con uno de 4 Mhz, aunque será muy difícil de filtrar y tendrá muy baja amplitud .Cualquier otra frecuencia causará que FREQOUT genere una frecuencia proporcional al oscilador comparado a 20 Mhz.

´ Enviar un tono de 1 Khz al Pin1 durante 2 segundos

FREQOUT PORTB.1 ,2000,1000

**GOSUB**

**GOSUB** etiqueta

Salta a la subrutina indicada en la etiqueta, guardando su dirección de regreso en la pila (stack) .A diferencia del GOTO, cuando se llega a un **RETURN**, la ejecución sigue con la declaración siguiente al último GOSUB ejecutado.

Se puede usar un número ilimitado de subrutinas en un programa y pueden estar anidadas. En otras palabras, las subrutinas pueden llamar a otra subrutina. Cada anidamiento no debe ser mayor de cuatro niveles.

**GOSUB** beep ´ ejecuta la subrutina beep

beep: high 0 ´ enciende el LED conectado a Pin0

sound 1, [ 80 , 10 ] ´ hace sonar el parlante conectado a Pin1

low 0 ´ apaga el LED conectado a Pin0

**return** ´ vuelve a la rutina principal

**GOTO**

**GOTO** etiqueta

La ejecución del programa continúa en la declaración de la etiqueta.

**GOTO** send ´ salta a la declaración etiquetada send

send: serout 0,N2400, [ ” Hi” ] ´ envía “ Hi” como salida al Pin0 en forma serial

**HIGH**

**HIGH** Pin

Hace de valor alto el Pin especificado y lo convierte automáticamente en salida. Pin puede ser una constante, 0 - 15, ó una variable que contenga un número de 0-15 (p.ej. B0) ó un número de Pin (p.ej. PORTA.0)

**HIGH 0** ´ convierte Pin0 en salida y lo coloca en valor alto

(-5 volt)

**HIGH PORTA.0** ´ convierte PORTA,Pin0 en salida y lo coloca en valor

alto (-5 volt)

led var PORTB.0 ´ define el pin LED

HIGH led ´ convierte el Pin LED en salida y lo coloca en valor

alto (-5 volt)

Como alternativa, si el pin ya es salida, hay una forma más rápida y corta de setearlo en valor alto (desde un código generado standpoint ):

PORTB.0 = 1 ´ setea PORTB Pin0 a valor alto.

**HPWM**

**HPWM**, Channel, Dutycycle, frecuencia

Hace salir un tren de pulso modulado en anchura usando PWM por hardware, disponible en algunos PICmicro. Puede ejecutarse continuamente en segundo plano mientras que el programa está ejecutando otras instrucciones.

Channel especifica qué canal físico PWM se va a utilizar.
Algunos dispositivos tienen 1, 2 o 3 canales de PWM.
En los dispositivos con 2 canales, la frecuencia debe ser igual en ambos canales.

Dutycycle especifica la relación de (alta-baja) con./desc. de la señal. Se extiende a partir de 0 a 255, donde 0 está apagado todo el tiempo y 255 es alto todo el tiempo. Un valor de 127 da un ciclo de 50% (onda cuadrada). La frecuencia es la frecuencia deseada de la señal de PWM. No todas las frecuencias están disponibles en todas las configuraciones del oscilador. La frecuencia más alta a cualquier velocidad del oscilador es 32767Hz. La frecuencia usable más baja de HPWM en cada configuración del oscilador se muestra en la tabla 3.8:

| OSC | 14-bit core y 18CXXX | 17Cxxx |
|---|---|---|
| 4 Mhz. | 245 hz. | 3907 hz. |

| | | |
|---|---|---|
| 8 Mhz. | 489 hz. | 7813 hz. |
| 10 Mhz. | 611 hz. | 9766 hz. |
| 12 Mhz. | 733 hz. | 11719 hz. |
| 16 Mhz. | 977 hz. | 15625 hz. |
| 20 Mhz. | 1221 hz. | 19531 hz. |
| 24 Mhz. | 1465 hz. | 23438 hz. |
| 33 Mhz. | 2015 hz. | 32227 hz. |
| 40 Mhz. | 2442 hz. | na |

**Tabla 3.8** *Frecuencias para HPWM*

Algunos dispositivos, tales como el PIC18C452, tienen pines alternos que se puedan utilizar para HPWM. Los DEFINE siguientes permiten el usar de estos pines:

```
DEFINE CCP1_REG PORTC ‘Hpwm 1 pin port
DEFINE CCP1_BIT 2 ‘Hpwm 1 pin bit
DEFINE CCP2_REG PORTC ‘Hpwm 2 pin port
DEFINE CCP2_BIT 1 ‘Hpwm 2 pin bit
```

Los siguientes DEFINE especifican qué temporizador, 1 o 2, utilizar con el canal 2 de PWM y el canal 3 de PWM para los dispositivos de PIC17C7xx. El valor por defecto es el temporizador 1 si ningún DEFINE se especifica.

```
DEFINE HPWM2_TIMER 1 ‘Hpwm 2 timer select
DEFINE HPWM3_TIMER 1 ‘Hpwm 3 timer select
```

HPWM 1,127,1000 ‘ envie a 50% ciclo de trabajo PWM a 1kHz
HPWM 1,64,2000 ‘ envie a 25% ciclo de trabajo PWM a 2kHz

**HSERIN**

**HSERIN** {ParityLabel , }{Timeout ,Label ,} [ Item { . .  } ]

Recibe uno ó más Ítems de un port serial ( de hardware ) en dispositivos que soportan comunicaciones seriales asincrónicas por hardware.

HSERIN es una de varias funciones seriales asincrónicas pre-construidas. Sólo puede ser usada en dispositivos que posean hardware USART. Vea la hoja de datos del dispositivo para información de los pin seriales de entrada y otros. Los parámetros seriales y el baud-rate son especificados usando DEFINE:

´ coloque el registro receptor en receptor habilitado

DEFINE HSER_RCSTA 90h

´ coloque el registro de transmisión en transmisión habilitada

DEFINE HSER_TSTA 20h

´ coloque baud rate

DEFINE HSER_BAUD 2400

HSERIN asume un oscilador de 4 Mhz cuando calcula el baud rate. Para mantener una relación de baud rate apropiada con otros valores de oscilador, use DEFINE para especificar el nuevo valor OSC.

Timeout y Label pueden ser incluidos en forma opcional para permitir al programa continuar si un carácter no es recibido dentro de un límite de tiempo. Timeout está especificado en unidades de 1 milisegundo.

El formato por defecto de los datos seriales es 8N1, 8 bits de datos, sin paridad y 1 stop bit.7E1 (7 bits de datos, paridad par, 1 stop bit) ó 7 O 1 (7 bits de datos, paridad impar, 1 stop bit) pueden ser habilitados usando los siguientes DEFINE:

´ use solo si se desea paridad par

DEFINE HSER_EVEN 1

´ use solo si se desea paridad impar

DEFINE HSER_ODD 1

El seteo de paridad igual que todos los DEFINE HSER afectan tanto a HSERIN como a **HSEROUT**

Se puede incluir ParityLabel como opcional en la declaración. El programa continuará en este punto si se recibe un carácter con error de paridad. Solo debe ser usado si se habilitó paridad con un DEFINE anterior.

Dado que la recepción serial se realiza por hardware, no es posible invertir los niveles para eliminar un driver RS - 232. Por esto debe usarse un driver adecuado con HSERIN.

HSERIN soporta los mismos modificadores de datos que **SERIN2**. Refiérase a la sección de SERIN2 para mayor información.

HSERIN [B0, dec W1]

**HSEROUT**

**HSEROUT** [Item {,Item } ]

Envía uno ó más Ítems al port serial de hardware en dispositivos que soportan comunicaciones seriales asincrónicas por hardware.

HSEROUT es una de varias funciones seriales asincrónicas pre-construidas. Sólo puede ser usada en dispositivos que posean hardware USART .Vea la hoja de datos del dispositivo para información de los pin seriales de entrada y otros. Los parámetros seriales y el baud-rate son especificados usando DEFINE:

´ coloque el registro receptor en receptor habilitado

DEFINE HSER_RCSTA 90h

´ coloque el registro de transmisión en transmisión habilitada

DEFINE HSER_TSTA 20h

´ coloque baud rate

DEFINE HSER_BAUD 2400

HSEROUT asume un oscilador de 4 Mhz cuando calcula el baud rate. Para mantener una relación de baud rate apropiada con otros valores de oscilador, use DEFINE para especificar el nuevo valor OSC.

El formato por defecto de los datos seriales es 8N1, 8 bits de datos, sin paridad y 1 stop bit. 7E1 (7 bits de datos, paridad par, 1 stop bit) ó 7 O 1 (7 bits de datos, paridad impar, 1 stop bit) pueden ser habilitados usando los siguientes DEFINE:

´ use solo si se desea paridad par

DEFINE HSER_EVEN 1

´ use solo si se desea paridad impar

DEFINE HSER_ODD 1

El seteo de paridad igual que todos los DEFINE HSER afectan tanto a **HSERIN** como a **HSEROUT**

Dado que la recepción serial se realiza por hardware, no es posible invertir los niveles para eliminar un driver RS - 232. Por esto debe usarse un driver adecuado con HSEROUT.

HSEROUT soporta los mismos modificadores de datos que SEROUT2 .Refiérase a la sección de **SEROUT2** para mayor información.

´ enviar el valor decimal de B0 seguido por un linefeed a través del USART

**HSEROUT [ dec B0 , 10 ]**

**I2CREAD**

**I2CREAD** DataPin ,ClockPin,Control,{Address,}

[ Var {,Var ...} ] { . Label }

Envía los bytes de Control y opcionalmente los de Address, a través del ClockPin y el DataPin y guarda los bytes recibidos dentro de Var. ClockPin y dataPin pueden ser constantes, 0-15, una variable que contenga un número (p.ej. B0), ó un número de Pin (p.ej. PORTA.0)

I2CREAD y I2CWRITE pueden ser usados para leer y grabar datos de un EEPROM serial usando una interface I2C de 2 cables, como Microchip 24LC01B ó similar .Esto permite guardar datos en una memoria externa no volátil, para que sean mantenidos aún sin energía conectada .Estos comandos funcionan en modo I2C master y también son usados para comunicarse con otros dispositivos con interfase I2C, como sensores de temperatura y convertidores A/D.

Los 7 bits superiores del byte de Control contienen el código de control junto con la selección del chip e información adicional de dirección, dependiendo de cada dispositivo. El bit inferior es una bandera interna que indica si es un comando de lectura ó escritura y no se debe usar.

Este formato para el byte de Control es diferente al usado por el PBP original. Asegúrese de usar este formato en operaciones PBP I2C.

Por ejemplo, cuando comunicamos con un 24LC01B, el código de control es %1010 y no se usa la selección de chip, por lo que el byte de Control será %10100000 ó $A0. Algunos formatos de Control son:

| Dispositivo | Capacidad | Control | Tamaño dirección |
|---|---|---|---|
| **24LC01B** | **128 bytes** | **%1010xxx0** | **1 byte** |
| **24LC02B** | **256 bytes** | **%1010xxx0** | **1 byte** |
| **24LC04B** | **512 bytes** | **%1010xxb0** | **1 byte** |
| **24LC08B** | **1 Kbytes** | **%1010xbb0** | **1 byte** |
| **24LC16B** | **2 Kbytes** | **%1010bbb0** | **1 byte** |
| **24LC32B** | **4 Kbytes** | **%1010ddd0** | **2 bytes** |
| **24LC65** | **8 Kbytes** | **%1010ddd0** | **2 bytes** |

**Tabla 3.9** *Formatos de control*

**bbb** = bits de selección de block (direcciones de orden alto)

**ddd** = bits de selección de dispositivo

**xxx** = no importa

El tamaño de dirección enviado (byte ó word) es determinado por el tamaño de la variable usada. Si se usa una variable con tamaño byte se envía una dirección de 8 bits. Si se envía una variable de tamaño word, se envía una dirección de 16 bits. Asegúrese de usar una variable apropiada al dispositivo a comunicar.

Si se especifica Var con tamaño word, se leen 2 bytes y se guarda primero el de mayor orden y luego el de orden inferior dentro de Var .Este orden es el inverso al que se osa normalmente con variables.

Si se usa la opción Label, se saltará a ella, si no se recibe un reconocimiento del dispositivo I2C.

Las instrucciones I2C pueden ser usadas para acceder al EEPROM incorporado en los dispositivos 12CExxx y 16CExxx. Simplemente especifique los nombres de pin de las líneas internas adecuadas como parte del comando I2C y coloque el siguiente DEFINE en el principio del programa.

DEFINE I2C_INTERNAL 1

Vea las hojas de datos de **Microchip** para más información.

El tiempo de las instrucciones I2C es tal que los dispositivos de velocidad standard (100 Khz) pueden ser accedidos a velocidad de clock de hasta 8 Mhz. Dispositivos rápidos (400 Mhz) pueden ser usados hasta 20 Mhz. Si se desea acceder un dispositivo de velocidad standard a 8 Mhz, se debe usar el siguiente DEFINE en el programa:

DEFINE I2C_SLOW 1

El clock I2C y las líneas de datos pueden ser empujados a Vcc con un resistor de 4-7 K de acuerdo al siguiente esquema, ya que ambos trabajan en modo de colector abierto.

addr var byte

cont con %10100000

addr =17 ´ coloca la dirección en 17

´ lee datos de la dirección 17 y los deja en B2

I2CREAD PORTA.0,PORTA.1,cont,addr, [ B2 ]

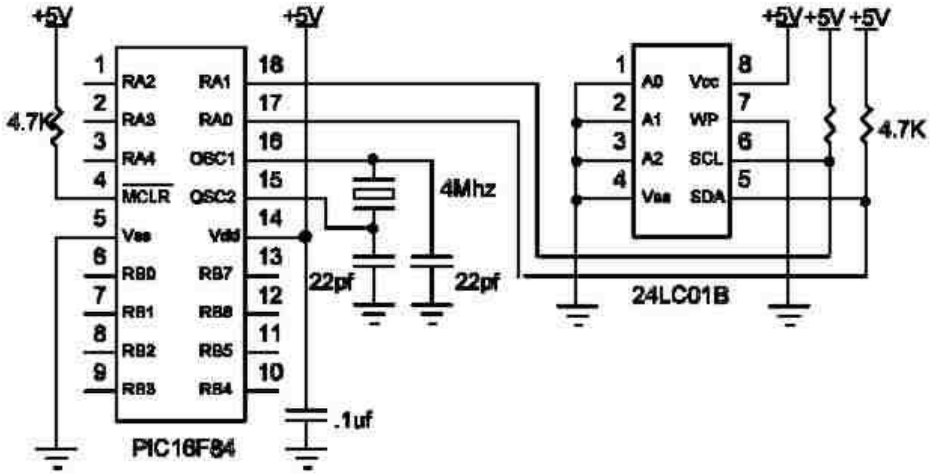



**Figura 3.23** *Forma típica para conectar el Pic con la interfaz I2C*

**I2CWRITE**

**I2CWRITE** DataPin ,ClockPin,Control,{Address,}

[ Value {,Value ...} ] { . Label }

Envía los bytes de Control y opcionalmente los de Address, a través del ClockPin y el DataPin seguidos por Value. ClockPin y DataPin pueden ser constantes, 0-15, una variable que contenga un número (p.ej. B0), ó un número de Pin (p.ej. PORTA.0)

El tamaño de dirección enviado (byte ó word) es determinado por el tamaño de la variable usada. Si se usa una variable con tamaño byte se envía una dirección de 8 bits. Si se envía una variable de tamaño word, se envía una dirección de 16 bits. Asegúrese de usar una variable apropiada al dispositivo a comunicar.

Cuando se escribe un EEPROM serial, es necesario esperar 10ms (dependiendo del dispositivo) para completar la grabación, antes de intentar comunicarse nuevamente con el dispositivo. Si se intenta un I2CWRITE ó I2CREAD antes que se complete la grabación, se ignorará el acceso.

Aunque una sola declaración I2CWRITE puede ser usada para grabar múltiples bytes simultáneamente, se puede violar los requerimientos de tiempo de grabación para los EEPROM seriales. Algunos permiten grabar múltiples bytes en una página simple antes de necesitar una espera. Revise la hoja de datos del dispositivo que esté usando. La opción de grabación múltiple puede ser útil con dispositivos I2C que no deban esperar entre grabaciones.

Si se usa la opción Label, se saltará a ella, si no se recibe un reconocimiento del dispositivo I2C.

Las instrucciones I2C pueden ser usadas para acceder al EEPROM incorporado en los dispositivos 12CExxx y 16CExxx. Simplemente especifique los nombres de pin de las líneas internas adecuadas como parte del comando I2C y coloque el siguiente DEFINE en el principio del programa.

DEFINE I2C_INTERNAL 1

El tiempo de las instrucciones I2C es tal que los dispositivos de velocidad standard (100 Khz) pueden ser accedidos a velocidad de clock de hasta 8 Mhz. Dispositivos rápidos (400 Mhz) pueden ser usados hasta 20 Mhz. Si se desea acceder un dispositivo de velocidad standard a 8 Mhz, se debe usar el siguiente DEFINE en el programa:

DEFINE I2C_SLOW 1

Vea el siguiente comando I2CREAD

addr var byte

cont con %10100000

addr =17 ´ coloca la dirección en 17

´ envía el byte 6 a la dirección 17

I2CWRITE PORTA.0,PORTA.1,cont,addr, [ 6 ]

Pause 10 ´ espera 10 ms que se complete la grabación

addr =1 ´ coloca la dirección en 1

´ envía el byte en B2 a la dirección 1

I2CWRITE PORTA.0,PORTA.1,cont,addr, [ B2 ]

Pause 10 ´ espera 10 ms que se complete la grabación

**IF ...THEN**

IF Comp { AND/OR Comp ... } THEN Label

IF Comp { AND/OR Comp ... } THEN

Declaración

ELSE

Declaración

ENDIF

Efectúa una ó más comparaciones. Cada término Comp puede relacionar una variable con una constante ú otra variable e incluye uno de los operadores listados anteriormente.

IF ... THEN evalúa la comparación en términos de CIERTO o FALSO. Si lo considera cierto, se ejecuta la operación posterior al THEN. Si lo considera falso, no se ejecuta la operación posterior al THEN. Las comparaciones que dan 0 se consideran falsos. Cualquier otro valor es cierto. Todas las comparaciones son sin signo, ya que PBP solo soporta operaciones sin signo.

Asegurase de usar paréntesis para especificar el orden en que se deben realizar las operaciones .De otra manera, la prioridad de los operadores lo determina y el resultado puede no ser el esperado.

IF...THEN puede operar de dos maneras. De una forma, el THEN en un IF..THEN es esencialmente un GOTO. Si la condición es cierta, el programa irá hacia la etiqueta que sigue al THEN. Si la condición es falsa, el programa

va a continuar hacia la próxima línea después del IF..THEN. Otra declaración no puede ser puesta después del THEN; sino que debe ser una etiqueta.

**If Pin0 = 0 Then pushd** ‘ si el botón conectado al pin 0 es oprimido (0), salta a la etiqueta pushd

**If B0 >=40 Then old** ‘ si el valor en la variable B0 es mayor ó igual a 40, salta a old

**If PORTB.0 Then itson** ‘si PORTB, pin 0 es alto (1), salta a itson

**If (B0 = 10) AND (B1 = 20) Then loop**

En la segunda forma, IF..THEN puede ejecutar condicionalmente un grupo de declaraciones que sigan al THEN. Las declaraciones deben estar seguidas por un ELSE o un ENDIF para completar la estructura.

**If B0 <> 10 Then**

**B0 = B0 + 1**

**B1 = B1 - 1**

**Endif**

**If B0 = 20 Then**

**led = 1**

**Else**

**led = 0**

**Endif**

**LCDIN**

**LCDIN** { Dirección, }[Var{, Var... } ]

Lee la RAM del LCD en el direccionamiento y salva los datos en Var.

LCDs tienen RAM onboard, eso se utiliza para la memoria de caracter.

La mayoría de los LCDs tienen más RAM disponible que la necesaria para el área mostrable.

Esta RAM se puede escribir usando la instrucción de **LCDOUT**.
La instrucción de LCDIN permite que esta RAM sea leída.
Los funcionamientos de la RAM del CG (generador de carácter) del LCD están en la dirección $40 a $7F. Comienzan en la dirección $80. Vea la hoja de datos del LCD específico para estos direccionamientos y funciones.

Es necesario conectar la línea de lectura/grabación del LCD con un contacto de PICmicro MCU para poder seleccionar si el modulo es leído (LCDIN) o para escribir (LCDOUT).

Dos DEFINE se emplean en las direcciones:

DEFINA LCD_RWREG PORTE. 'lectura/grabación del pin port del LCD
DEFINE LCD_RWBIT 2. ' pin bit LCD lectura/grabación

LCDOUT, considere la información para conectar un LCD con un PICmicro MCU.

LCDIN [ B0 ]

**LCDOUT**

**LCDOUT** Item{ , Item...}

Muestra Items en un visor de cristal líquido inteligente (LCD). PBP soporta módulos LCD con un controlador Hitachi 44780 o equivalente. Estos LCD, usualmente, tienen un cabezal de 14 o 16 pines simples o duales en un extremo.

Si el signo (#) está colocado antes de un Ítem, la representación ASCII para cada dígito es enviada al LCD. LCDOUT también puede usar cualquiera de los modificadores usados con SEROUT2.

Un programa debe esperar, por lo menos, medio segundo antes de enviar el primer comando a un LCD. Puede tomar bastante tiempo a un LCD arrancar.

Los comandos son enviados al LCD, enviando un $FE seguido por el comando. Algunos comandos útiles se muestran en la tabla 3.10:

| Comando | Operación |
|---|---|
| **$FE, 1** | **Limpia visor** |
| **$FE, 2** | **Vuelve a inicio (comienzo de la primera línea)** |
| **$FE, $0C** | **Cursor apagado** |
| **$FE, $0E** | **Subrayado del cursor activo** |
| **$FE, $0F** | **Parpadeo del cursor activo** |
| **$FE, $10** | **Mueve cursor una posición hacia la izquierda** |
| **$FE, $14** | **Mueve cursor una posición hacia la derecha** |
| **$FE, $C0** | **Mueve cursor al comienzo de la segunda línea** |

**Tabla 3.10** *Comandos para LCD*

Note que hay un comando para mover el cursor al comienzo de la segunda línea en un visor de dos líneas. Para muchos LCD, los caracteres y líneas mostrados no son consecutivos en la memoria del visor - puede haber un salto entre las localizaciones. Para muchos visores 16x2, la primera línea comienza en $0 y la segunda, en $40. El comando:

**LCDOUT $FE, $C0**

Hace que el visor comience a escribir caracteres en el principio de la segunda línea. Los visores 16x1 usualmente están formateados como visores de 8x2, con un salto entre las locaciones de memoria para los primeros y segundos caracteres de 8. Los visores de 4 líneas, también tienen un mapa de memoria no ordenado.

Vea la hoja de datos para el dispositivo LCD, en particular el que usted esté usando, para las locaciones de memoria de caracter y comandos adicionales.

**LCDOUT $FE, 1, "Hello"** 'limpia el visor y muestra "Hello"

**LCDOUT B0, #B1**

El LCD puede estar conectado al micro Pic, usando un bus de 4 bit o uno de 8 bit. Si se usa un bus de 8 bit, todos los 8 bits deben estar en un port. Si se usa un bus de 4 bit, debe estar conectado o a los 4 bit inferiores o a los 4 bit superiores de un port. Enable y Register Select deben estar conectados a algún pin del port. R/W debe estar colocado a tierra, ya que el comando de LCDOUT solamente es de grabación.

PBP supone que el LCD está conectado a pines específicos, a menos que se le diga de otra manera. Asume que el LCD va a ser usado con un bus de 4 bits, con las líneas de data DB4 - DB7 conectadas en el micro Pic a PORTA.0 - PORTA.3, Register Select a PORTA.4 y Enable a PORTB.3. Además, inicializa el LCD como un visor de dos líneas.

Para cambiar este seteo, coloque uno o más de los siguientes DEFINEs, todos en mayúsculas, en el comienzo de su programa PBP:

```
‘ Setea el port de datos LCD

DEFINE LCD_DREG PORTB

‘ Setea el bit de comienzo de datos (0 o 4) si el bus es de 4-bit

DEFINE LCD_DBIT 0

‘ Setea el port LCD Register Select

DEFINE LCD_RSREG PORTB

‘ Setea el bit LCD Register Select

DEFINE LCD_RSBIT 4

‘ Setea el port LCD Enable
```

DEFINE LCD_EREG PORTB

‘ Setea el bit LCD Enable

DEFINE LCD_EBIT 5

‘ Setea el tamaño del bus LCD (4 o 8 bits)

DEFINE LCD_BITS 4

‘ Setea el numero de líneas en el LCD

DEFINE LCD_LINES 2

Este seteo, le dirá a PBP que hay conectado un LCD de 2 líneas en modo de 4 bit con el bus de datos en los 4 bit inferiores de PORTB, Register Select en el PORTB.4, y Enable en el PORTB.5.

El siguiente esquema muestra una forma de conectar un LCD a un micro Pic:

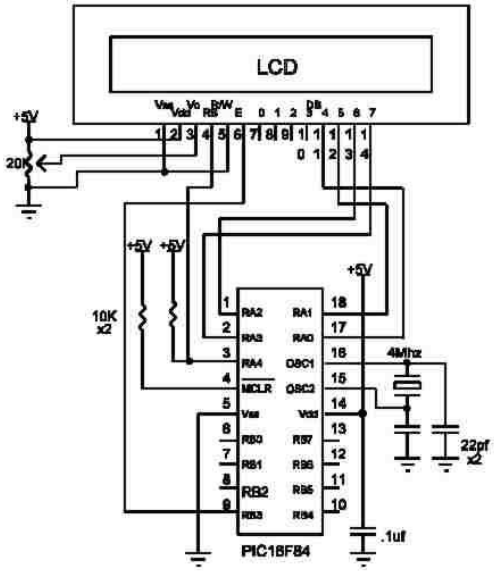



**Figura 3.24** *Forma típica de conexión del LCD al PIC*

**LOOKUP**

**LOOKUP** Index, [Constant{ , Constant...}] , Var

LOOKUP puede ser usado para obtener valores de una tabla de constantes de 8 bits, Si Índex es cero, Var toma el valor de la primer Constanet. Si Índex es 1, Var toma el valor de la segunda Constante y así sucesivamente. Si Índex es mayor ó igual que el número de entradas en la lista de constantes, no se toma ninguna acción y Var permanece sin cambios.

La lista de constantes puede ser una mezcla de constantes numéricas y cadenas. Cada carácter en una cadena es tratado como una constante separada con el valor del carácter ASCII. Las variables de array con índice variable no pueden ser usadas en LOOKUP, aunque son permitidas las variables de array con índice constantes.

For B0=0 to 5 ´ cuenta de 0 a 5

**LOOKUP B0, [ "Hello " ],B1** ´ obtiene el carácter B0 de la cadena y lo deja en B1

Serout 0,N2400, [ B1 ] ´ envía el carácter en B1 al Pin0 en forma Serial

Next B0 ´ va al segundo carácter

**LOW**

**LOW** Pin

Coloca el pin especificado en valor bajo y automáticamente lo convierte en salida .Pin puede ser una constante, 0-15, +o una variable que contenga un número 0-15 (p.ej. B0) ó un nombre de pin (p.ej. PORTA.0)

**LOW 0** ´ Coloca el Pin0 en salida y nivel bajo ( 0 volt)

**LOW PORTA.0** ´ Coloca PORTA.0 como salida y en nivel bajo (0 volt)

Led var PORTB.0 ´ define un pin LED

**LOW led** ´ coloca el pin LED como salida y en valor bajo (0 volt)

Si el pin ya es una salida, es más rápido usar un código ya generado:

**PORTB.0 = 0** ´ coloca en nivel bajo el pin0 de PORTB

**NAP**

**NAP** Periodo

Coloca al micro controlador en modo de baja potencia por períodos de tiempo reducidos (siesta). Durante este NAP, se reduce al mínimo el consumo de energía. Los períodos indicados son solo aproximados, porque el tiempo se deriva del Watchdog Timer que está controlado por R/C y puede variar de chip a chip y también con la temperatura. Como NAP usa el Watchdog Timer es independiente de la frecuencia del oscilador.

| **Periodo** | **Demora (aprox.) en milisegundos** |
|---|---|
| **0** | **18** |
| **1** | **36** |
| **2** | **72** |
| **3** | **144** |
| **4** | **288** |
| **5** | **576** |
| **6** | **1152** |
| **7** | **2304** |

**Tabla 3.11** *Demora en milisegundos en función del periodo*

NAP 7 ´ pausa en baja potencia por aprox. 2,3 segundos

**PAUSE**

**PAUSE** Periodo de milisegundos

Detiene el programa por un Periodo milisegundos. Periodo tiene 16 bit, por lo que los retardos pueden ser de hasta 65.535 milisegundos (un poco mas de 1 minuto). No coloca el micro controlador en modo de baja potencia como las otras funciones de retardo (NAP y SLEEP). Inclusive, consume mayor potencia, pero es más exacto. Tiene la misma precisión que el clock.

PAUSE asume la frecuencia de 4 Mhz del oscilador. Si se usa un oscilador de otra frecuencia, se debe indicar usando el comando DEFINE OSC. Vea la sección sobre velocidad para mayores detalles.

**PAUSE 1000** ' demora de 1 segundo

**PAUSEUS**

**PAUSEUS** Periodo

Detiene el programa por Periodo milisegundos. Periodo tiene 16 bit, por lo que los retardos pueden ser de hasta 65.535 milisegundos. No coloca el micro controlador en modo de baja potencia como las otras funciones de retardo (NAP y SLEEP). Inclusive, consume mayor potencia, pero es más exacto. Tiene la misma precisión que el clock.

PAUSE tiene un número mínimo de ciclos para operar .Como depende de la frecuencia del oscilador, no es posible obtener demoras menores a un número mínimo de microsegundos usando PAUSEUS. Para obtener demoras precisas, menores que esto use una rutina ensambladora tipo ASM...ENDASM. La tabla siguiente muestra el número mínimo de microsegundos obtenible para una determinada frecuencia de oscilador.

| OSC | Demora mínima |
|---|---|
| 3(3.58) | 20us |
| 4 | 24us |
| 8 | 12us |
| 10 | 8us |
| 12 | 7us |
| 16 | 5us |
| 20 | 3us |

**Tabla 3.12** *Demora en microsegundos en función del oscilados*

PAUSEUS asume la frecuencia de 4 Mhz del oscilador. Si se usa un oscilador de otra frecuencia, se debe indicar usando el comando DEFINE OSC. Vea la sección sobre velocidad para mayores detalles.

**PAUSEUS 1000** ' demora de 1 segundo

**POT**

**POT** Pin,Scale,Var

Lee un potenciómetro (ú otro dispositivo resistivo) en Pin. Pin puede ser una constante, 0 - 15, ó una variable que contenga un número de 0-15 (p.ej. B0) ó un número de Pin (p.ej. PORTA.0)

La resistencia se mide tomando el tiempo de descarga de un capacitor a través de un resistor. ( 5 K a 50 K ). Scale se usa para ajustar distintas constantes RC. Para constantes RC grandes, Scale debe ser baja (valor mínimo 1). Para constantes RC pequeñas, Scale debe ser máxima (255). Si el valor de Scale es correcto, Var debe ser cero para mínima resistencia y 255 para máxima resistencia.

Desafortunadamente, Scale debe ser determinada en forma experimental. Para esto, coloque el dispositivo a medir en máxima resistencia y mídalo con Scale=255. En estas condiciones, Var tendrá un valor apropiado de Scale. (Este es el mismo tipo de proceso que efectúa la opción ALT-P en BS1).

**POT,3,255,B0** ´ lee el potenciómetro en pin 3 para determinar Scale

Serout 0,N2400,[#B0] ´ envía el valor del potenciómetros en forma serial al pin 0

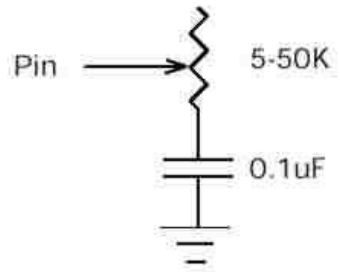



**Figura 3.25** *Esquema de conexión de un potenciómetro*

**PWM**

**PWM** Pin,Duty,Cycle

Envía un tren de pulsos modulados en ancho a Pin .Cada ciclo de PWM está compuesto de 256 pasos. El ciclo útil Duty para cada ciclo varía de 0 (0%) a 255 (100%= .El ciclo PWM es repetido Cycle veces. Pin puede ser una constante, 0 - 15, ó una variable que contenga un número de 0-15 (p.ej. B0) ó un número de Pin (p.ej. PORTA.0)

Cycle depende de la frecuencia del oscilador .Con un oscilador de 4 Mhz, cada Cycle será de aproximadamente 5 mseg. de largo. Con un oscilador de 20 Mhz el largo aproximado será de 1 mseg. Definir un valor de OSC no tiene efecto sobre **PWM**. El tiempo de Cycle siempre cambia con la velocidad del oscilador en uso.

Pin se convierte en salida justo antes de la generación del pulso y vuelve a ser entrada, cuando cesa .La salida de PWM en un pin tiene mucho ruido, y no tiene forma de onda cuadrada .Es necesario usar algún tipo de filtro para convertirla en algo útil. Un circuito R/C se puede usar como un simple convertidor D/A.

**PWM PORTB.7,127,100** ´ envía una señal PWM con un ciclo útil del 50% al pin 7 , durante 100 ciclos

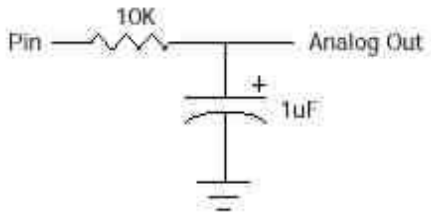



**Figura 3.26** Esquema de conexión para la salida de un PIN PWM

**READ**

**READ** Address,Var

Lee el EEPROM incorporado en la dirección Address, y guarda el resultado en Var. Esta instrucción solo puede ser usada con un microPIC que tenga un EEPROM incorporado como el PIC16F84 ó PIC16C84

**READ 5,B2** ´ coloca en B2 el valor de la dirección 5 del EEPROM

**RETURN**

Vuelve desde una subrutina. Retoma la ejecución en la declaración que sigue al GOSUB que llamó la subrutina.

Gosub sub1 ´ va a la subrutina denominada sub1

. . .

sub1: serout 0,N2400,[“Lunch”] ´ envía “Lunch” al pin 0 en forma serial

RETURN ´ vuelve al programa principal después del gosub

**SERIN**

**SERIN** Pin,Mode, {Timeout,Label,}{[Qual...],} {Item...}

Recibe uno ó más Ítems en Pin, en formato standard asincrónico, usando 8 bit de datos, sin paridad y un stop bit (8N1). SERIN es similar al comando Serin de BS1 con el agregado de Timeout. Pin automáticamente se convierte en entrada. Pin puede ser una constante, 0 - 15, ó una variable que contenga un número de 0-15 (p.ej. B0) ó un número de Pin (p.ej. PORTA.0)

Los nombres Mode (p.ej. T2400 ) están definidos en el archivo MODEDEFS.BAS .Para usarlos ,agregue la línea:

Include “modedefs.bas”

Al comienzo del programa PBP. BS1DEFS,BAS y BS2DEFS.BAS ya incluyen MODEDEFS.BAS .No lo incluya, si ya está usando alguno de estos archivos. Los números Mode pueden ser usados sin incluir este archivo.

| Mode | Mode N° | Baud rate | State |
|---|---|---|---|
| T2400 | 0 | 2400 | VERDADERO |
| T1200 | 1 | 1200 | |
| T9600 | 2 | 9600 | |
| T300 | 3 | 300 | |
| N2400 | 4 | 2400 | FALSO |
| N1200 | 5 | 1200 | |
| N9600 | 6 | 9600 | |
| N300 | 7 | 300 | |

**Tabla 3.13** *Baud Rate en función del Mode*

Aunque los chips convertidores de nivel RS-232 son comunes y baratos, las excelentes especificaciones de I/O de los microPIC permiten ejecutar muchas aplicaciones sin usar convertidores de nivel. Más aún, se pueden usar entradas invertidas ( N300...N9600) junto con un resistor limitador de corriente de 22k.

SERIN 1,N2400, [“A”],B0 ´ espera hasta que el carácter “A” sea recibido en forma serial en el pin 1 y coloca el próximo caracter en B0


**Figura 3.27** *Esquema de conexión para la transmisión en interface RS-232*

Algunos baud rate standard se muestran en la tabla 3.14

| Baud rate | Bits 0 - 12 |
|---|---|
| 300 | 3313 |
| 600 | 1646 |
| 1200 | 813 |
| 2400 | 396 |
| 4800 | 188 |
| 9600 | 84 |
| 19200 | 32 |

**Tabla 3.14** *Baud Rate Standard*

**SEROUT**

SEROUT Pin,Mode,[Item[,Item...}]

Envía uno ó más Ítems a Pin, en formato standard asincrónico usando 8 bits de datos, sin paridad y 1 stop bit (8N1).Pin es automáticamente colocado como salida. Pin puede ser una constante, 0 - 15, ó una variable que contenga un número de 0-15 (p.ej. B0) ó un número de Pin (p.ej. PORTA.0)

SEROUT soporta 3 tipos distintos de datos, que pueden ser combinados libremente dentro de una declaración SEROUT.

Una cadena de constantes es enviada como una cadena de caracteres literales.

Un valor numérico (constante ó variable) va a enviar el correspondiente carácter ASCII. Más aún, 13 es retorno de carro ( Carriage Return ó CR ) y 10 es avance de línea ( Line Feed ó LF ).

Un valor numérico precedido por el signo # va a enviar la representación ASCII de su valor decimal. Por ejemplo, si W0=123, entonces #W0 (ó #123) va a enviar “1”,”2”,”3”.

SEROUT asume un valor de oscilador de 4 Mhz cuando genera sus tiempos de bit .Para mantener los valores de baud rate adecuados con otro oscilador, asegúrese de usar DEFINE OSC con el nuevo valor de oscilador.

En algunos casos, los rangos de transmisión de SEROUT pueden presentar los caracteres demasiado rápidamente en el dispositivo receptor. Un DEFINE agrega tiempo entre caracteres en la transmisión de salida .Esto permite un tiempo adicional entre caracteres a medida que son transmitidos. Se puede lograr una demora entre cada carácter transmitido de 1 a 65535 microsegundos (0.001 a 65,535 milisegundos).

Por ejemplo, para pausar 1 milisegundo entre cada carácter transmitido:

DEFINE CHAR_PACING 1000

Aunque los chips convertidores de nivel RS-232 son comunes y baratos gracias a la implementación de corriente RS-232 y las excelentes especificaciones de I/O del microPIC, no se requieren convertidores de nivel en muchas aplicaciones. Se puede usar TTL invertido ( N300 ...N9600 ). Se sugiere el uso de un resistor limitador de corriente de 1K (se supone que RS-232 es tolerante a los cortocircuitos).


**Figura 3.28** *Esquema de conexión para la recepción en interface RS-232*

**SEROUT 0,N2400,[#B0,10]** ´ envía el valor ASCII de B0 ,seguido por un LF al pin 0 , en forma serial

**SLEEP**

**SLEEP** Periodo

Coloca al micro controlador en modo de baja potencia por periodos segundos. Periodo tiene 16 bit, por lo que los retardos pueden ser de hasta 65535 segundos (aprox. 18 horas).

SLEEP usa el WatchDog Timer, por lo que es independiente de la frecuencia del oscilador utilizado .La granulación es aproximadamente 2.3 segundos y puede variar de acuerdo al dispositivo y la temperatura. Esta variación es distinta a la de BASIC Stamp. Se necesitó este cambio, porque cuando el micro PIC pone a cero (resetea) el WatchDog Timer, también pone valores predefinidos en los registros internos. Estos valores pueden diferir de los esperados por su programa. Ejecutando el comando SLEEP sin calibrar, este paso se deja de lado.

**SLEEP 60** ´ duerme por aprox. 1 minuto

**SOUND**

**SOUND** Pin,[Note,Duration{,Note,Duration...}]

Genera un tono y/o ruido blanco en el Pin especificado. Pin es automáticamente colocado como salida. Pin puede ser una constante, 0 - 15, ó una variable que contenga un número de 0-15 (p.ej. B0) ó un número de Pin (p.ej. PORTA.0)

Note 0 es silencio. Nte 1-127 son tonos. Notes 128-255 son ruido blanco. Los tonos y el ruido blanco están en una escala ascendente (p.ej. 1 y 128 son las

frecuencias menores, 129 y 266 las mayores). Note 1 es aprox. 78,74 Hz y Note 127 es aprox. 10000 Hz.

Duration es 0-255 y determina el largo de la nota, en incrementos de 12 milisegundos. Note y Duration no necesitan ser constantes.

SOUND entrega como salida ondas cuadradas con nivel TTL. Gracias a las características del micro PIC, se puede manejar un parlante a través de un capacitor -El valor del capacitor debe ser determinado en función de las frecuencias a usar y la carga del parlante. Parlantes piezoeléctricos pueden ser conectados directamente.

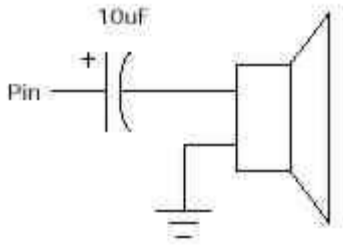



**Figura 3.29** *Esquema para conectar un parlante al pin de salida del PIC*

**SOUND PORTB.7,1[100,10,50,10]** ´envía 2 sonidos consecutivos a pin7

**WHILE...WEND**

**WHILE** Condition

Statement

**WEND**

Ejecuta las declaraciones Statement en forma repetida, mientras la condición Condition sea cierta. Cuando Condition deja de ser cierta, la ejecución continúa con la declaración siguiente al WEND. Condition puede ser cualquier expresión de comparación.

I=1

**WHILE i < = 10**

Serout 0,N2400,[“No:”,#i,13,10]

**WEND**

**WRITE**

**WRITE** Address,Value

Graba valores Value en el EEPROM incorporado en la dirección Address especificada. Esta instrucción solo puede ser usada con un microPIC que tenga un EEPROM incorporado como el PIC16F84 ó PIC16C84

Es usado para colocar datos en el EEPROM durante el momento de la ejecución .Para grabar datos en el EEPROM durante la programación, se usan las declaraciones DATA y EEPROM.

Cada WRITE se auto regula en tiempo y toma aproximadamente 10 milisegundos ejecutarlo en un microPIC.

**WRIT 5,B0** ´ envía el valor de B0 al EEPROM pin 5

### 3.9.4 Velocidad Del Procesador

Por defecto PBP genera programas sobre la base de un microPIC con un cristal de 4 Mhz ó un resonador cerámico.

Todas las instrucciones sensibles al tiempo, asumen un tiempo de instrucción de 1 microsegundo para sus demoras. Esto permite a PAUSE 1000, por ejemplo, esperar 1 segundo y a los comandos SERIN y SEROUT, baud rates exactos.

Sin embargo, puede ser útil hacer funcionar al microPIC a otra frecuencia distinta de 4 Mhz. Aunque los programas compilados son rápidos, es mejor

hacerlos funcionar más rápido. O quizás se desea entrada y salida serial con un baud rate de 19200 baud en lugar de usar el tope común de 9600 baud

Los programas PBP pueden funcionar a frecuencias de clock distintas de 4 Mhz de dos maneras. La primera, es simplemente usar un oscilador de frecuencia distinta de 4 Mhz y no indicárselo a PBP. Esta es una técnica útil si usted prestó atención a lo que sucede con las instrucciones dependientes del tiempo.

Si desea hacer funcionar el bus serial a 19200 baud, simplemente cambie el cristal de 4 Mhz, por uno de 8 Mhz. Esto, en efecto, hace funcionar todo dos veces más rápido, incluyendo los comandos SERIN y SEROUT. Si le indica a los comandos SERIN y SEROUT que operen a 9600 baud, doblando la velocidad del oscilador, funcionarán a 19200 baud.

Sin embargo, tenga en cuenta que comandos como PAUSE y SOUND se ejecutarán dos veces más rápido. PAUSE 1000 sólo esperará medio segundo con un cristal de 8 Mhz antes de permitir la continuación del programa.

La otra manera es usar una frecuencia de oscilador diferente e indicárselo al PBP. Esto se hace usando DEFINE, como se demostró con el comando LCDOUT anteriormente, se usa para indicarle a PBP que debe usar otros parámetros que no son los usados por defecto.

Normalmente, por defecto PBP usa un oscilador de 4 Mhz0. Agregando la declaración:

**DEFINE OSC 8**

Cerca del comienzo del programa PBP, se asume que se usará un oscilador de 8 Mhz. Las definiciones aceptables son:

| **OSC** | **Oscilador usado** |
|---|---|
| **3** | **3.58 Mhz** |

| | |
|---|---|
| **4** | **4 Mhz** |
| **8** | **8 Mhz** |
| **10** | **10 Mhz** |
| **12** | **12 Mhz** |
| **16** | **16 Mhz** |
| **20** | **20 Mhz** |

**Tabla 3.15** *Define OSC en función del oscilador*

Indicando a PBP la frecuencia del oscilador se le permite compensar y producir los tiempos correctos para COUNT0, DEBUG, DTMFOUT, FREQOUT, HSERIN, HSEROUT, I2CREAD, I2CWRITE, LCDOUT, PAUSE, PAUSEUS, SERIN, SERIN2, SEROUT, SEROUT2, SHIFTIN, SHIFTOUT, SOUND, XIN y XOUT.

Cambiando la frecuencia del oscilador puede ser usado para mejorar la resolución de las instrucciones PULSIN, PULSOUT y RCTIME. A 4 Mhz, operan con una resolución de 10 microsegundos. Si se usa un cristal de 20 Mhz, la resolución será de 2 microsegundos. Pero, el ancho del pulso es medido en una variable de 16 bit. Con una resolución de 2 microsegundos, el máximo ancho de pulso medible será de 131070 microsegundos.

Yendo en la otra dirección y utilizando un oscilador de 32768 Khz es problemático. Puede ser deseable, si se desea reducir el consumo de potencia. Los comandos SERIN y SEROUT son inutilizables, y el WatchDog Timer puede hacer que el programa recomience sólo en cualquier momento. Experimente si su aplicación funciona con esta velocidad del oscilador.

## 3.10 Vida Después De 2k

Si hay vida después de 2K usando el compilador PBP.

Los microPIC tienen un espacio de código segmentado. Las instrucciones microPIC, como Call y Goto ya tienen suficiente código como para llenar 2K de espacio de programa. Para llegar al código que está fuera del límite de 2 K, el registro PCLATH debe ser activado antes de cada Call ó Goto.

PBP automáticamente setea estos bits PCLATH por usted. Sin embargo, hay algunas restricciones. La librería PBP debe entrar completa dentro de la página 0 del espacio de código. Normalmente, no es problema, ya que la librería es el primer elemento en un programa PBP y la librería completa es menor de 2K. Sin embargo, se debe prestar atención a esto, si se usan librerías adicionales.

Los handler de interrupción de lenguaje ensamblador también deben entrar en la página 0 del espacio de código. Esto se logra colocándolos al principio del programa PBP.

Agregar instrucciones para setear los bits PCLATH agrega overhead al código generado. PBP va a setear los bits de PCLATH para cualquier código que cruce el límite de 2K ó para cualquier referencia en los microPIC con más de 2K de espacio de código.

Hay instrucciones PBP específicas para ayudar al uso de más de 2K.

BRANCHL se creó para permitir saltos a etiquetas que están del otro lado del límite de 2K. Si el microPIC tiene 2K ó menos de espacio de código, se debe usar BRANCH, ya que ocupa menos espacio que BRANCHL. Si el micro controlador tiene más de 2 K de espacio de código, y no está seguro de que los saltos siempre serán en la misma página, use BRANCHL.

El ensamblador puede enviar un aviso acerca de que el límite de página ha sido cruzado. Esto es normal y es aconsejable que usted controle por cualquier BRANCH que cruce el límite de página.

## 3.11 Interrupciones

Las interrupciones pueden ser una forma de hacer que su programa sea realmente difícil de depurar.

Las interrupciones son disparadas por eventos de hardware, ya sea un pin de I/O cambiando su estado ó un tiempo terminado ó cualquier otro. Si está habilitada (por defecto no lo está), la interrupción causa que el procesador detenga lo que está haciendo y salte a una rutina específica en el micro controlador, llamada handler de interrupciones. Pueden ser difíciles de implementar adecuadamente, pero también pueden proveer funciones muy útiles.

Por ejemplo, una interrupción puede ser usada para acumular datos seriales de entrada, mientras el programa principal está haciendo otra tarea. (Este uso particular requiere un micro controlador con un port serial).

Hay muchas formas de evitar usar las interrupciones. Un polling rápido de un pin ó un bit de registro hace el mismo trabajo en forma rápida. O se puede verificar el valor de una bandera de interrupción sin tener que habilitarlo. Sin embargo, si usted desea hacerlo, le damos algunas ideas de cómo hacerlo. El compilador PBP tiene dos mecanismos diferentes para manejar interrupciones. La primera es simplemente escribir el handler de interrupción en ensamblador y colocarlo en el frente de un programa PBP. El segundo método es usar la declaración ON INTERRUPT. Cada método será explicado por separado, después de explicar las interrupciones en general.

Cuando ocurre una interrupción, el microPIC guarda la dirección de la próxima instrucción que debería ejecutar en el stack (pila) y salta a la dirección 4. Esto significa que se necesita una dirección extra en el stack de hardware, que solamente tiene 8.

Las librerías de rutinas de PBP pueden usar hasta 4 direcciones del stack, ellas solas. Las 4 restantes están reservadas para CALLs y GOSUBs anidados. Debe asegurarse de que sus GOSUB no estén anidados en más de tres niveles, y sin CALL dentro de ellos, para tener una dirección de stack disponible para la dirección de regreso Si su handler de interrupciones usa el stack (haciendo un CALL ó un GOSUB,p.ej.) necesitará espacio adicional disponible en el stack.

Después, usted necesita habilitar las interrupciones apropiadas. Eso significa dar valores al registro INTCON. Setee los bits de habilitación necesarios junto con el Global Interupt Enable. Por ejemplo;

**INTCON = %10010000**

Habilita la interrupción para RB0/INT. Dependiendo de la interrupción deseada, puede necesitar setear el registro PIE.

Refiérase a los manuales de Microchip para información adicional acerca de cómo usar las interrupciones.

**INTERRUPCIONES EN BASIC**

La forma más fácil de escribir un handler de interrupción, es escribirlo en PBP junto a una declaración ON INTERRUPT. ON INTERRUPT le indica a PBP que active su handler interno de interrupción (el de PBP) y salte tan pronto pueda a su handler de interrupción BASIC (creado por el usuario) después de recibir una interrupción.

Usando ON INTERRUPT, cuando ocurre una interrupción, PBP simplemente marca el evento y vuelve a la tarea que estaba realizando. No salta inmediatamente al handler. Como las declaraciones de PBP no son re-entrantes (PBP debe terminar con la declaración en curso antes de ejecutar otra) puede haber considerable demora (latencia) antes de manejar a la interrupción. Como ejemplo, digamos que el programa PBP recién comenzó la ejecución de PAUSE 10000 cuando ocurre una interrupción. PBP marca la interrupción y continúa con el PAUSE, pueden transcurrir 10 segundos antes de que se ejecute el handler de interrupción. Si se están acumulando caracteres en un port serial, muchos de ellos pueden perderse.

Para minimizar este problema, use declaraciones que no tomen mucho tiempo de ejecución. Por ejemplo, en lugar de PAUSE 1000 use PAUSE 1 dentro de un loop FOR...NEXT. Esto permite completar cada declaración más rápidamente y manejar cualquier interrupción pendiente. Si se necesita un proceso de interrupción más rápido que el provisto por ON INTERRUPT, se deben usar interrupciones en lenguaje ensamblador.

Lo que sucede cuando se usa ON INTERRUPT es lo siguiente: un corto handler de interrupción es colocado en la dirección 4 del microPIC. Este handler de interrupción es simplemente un RETURN. Esto envía el programa de vuelta a lo que estaba haciendo antes de ocurrir la interrupción. No requiere guardar ningún contexto del procesador. Lo que esto no hace es re-habilitar el Global Interrupts, como hace el Retfie.

Un Call a una pequeña subrutina es colocado después de cada declaración en el programa PBP cuando se encuentra un ON INTERRUPT. Esta pequeña subrutina verifica el estado del bit de Global Interrupt Enable. Si está apagado hay una interrupción pendiente, por lo que apunta al handler de interrupción del usuario. Si no el programa continúa con la próxima declaración BASIC, después de lo cual se verifica nuevamente el bit GIE, y así sucesivamente.

Cuando se encuentra una declaración RESUME después del handler de interrupción BASIC, setea el bit GIE para rehabilitar las interrupciones y vuelve donde estaba el programa cuando ocurrió la interrupción. Si se indica en RESUME una etiqueta hacia donde saltar, la ejecución continuará en esa dirección. En ese caso, se pierden todas las direcciones previas de regreso.

DISABLE detiene PBP insertando un Call al verificador de interrupción después de cada declaración. Esto permite que se ejecuten secciones de código sin la posibilidad de ser interrumpidas. ENABLE permite la inserción para continuar.

DISABLE debe ser colocado antes que el handler de interrupción para que éste no sea arrancado cada vez que se chequee el bit GIE.

Si por alguna razón se desea apagar las interrupciones después que se encuentra un ON INTERRUPT, no debe apagar el bit GIE. Apagando este bit, se le indica a PBP que ha sucedido una interrupción y esto ejecutará el handler de interrupción por siempre. En su lugar haga:

**INTCON = $80**

Esto deshabilita todas las interrupciones individuales, pero deja seteado el bit **GIE**.

Una nota final acerca de las interrupciones en BASIC. Si el programa usa:

**loop: goto loop**

Y espera ser interrumpido, eso no sucederá. Recuerde que la bandera de interrupción es chequeada después de cada instrucción. Realmente no hay un lugar donde chequear después de un GOTO. Inmediatamente salta al loop, sin chequear la interrupción. Debe colocarse alguna declaración dentro del loop, para que haya un chequeo de interrupciones.

## 3.12 Tutorial De Dxp 2004.

### 3.12.1 Página de Inicio (HOME)

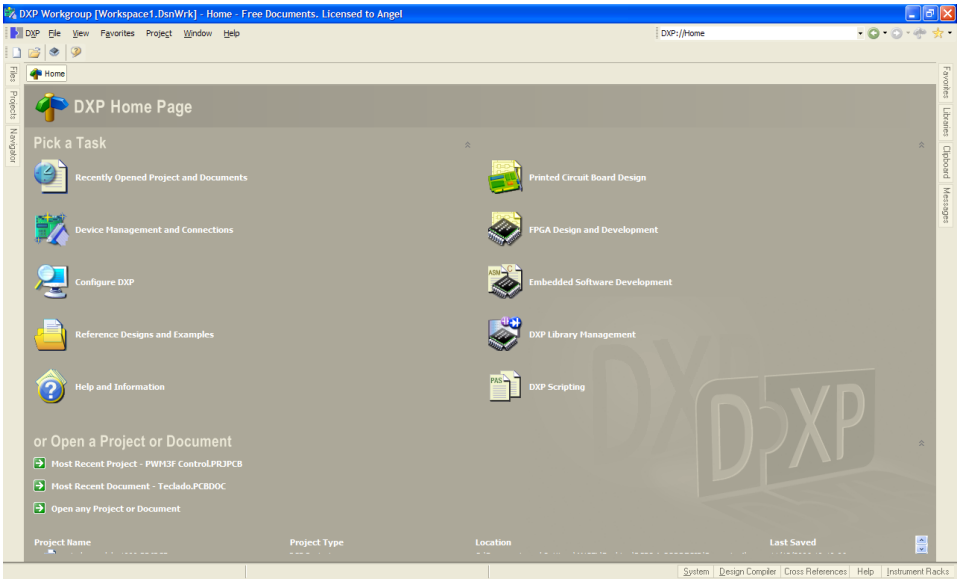



**Figura 3.30** *Página principal de DXP*

### 3.12.2 Diseño de hoja esquemática.

1. Comenzar abriendo un nuevo documento Esquemático (Schematic):

Menú: File ⟶ New ⟶ Schematic

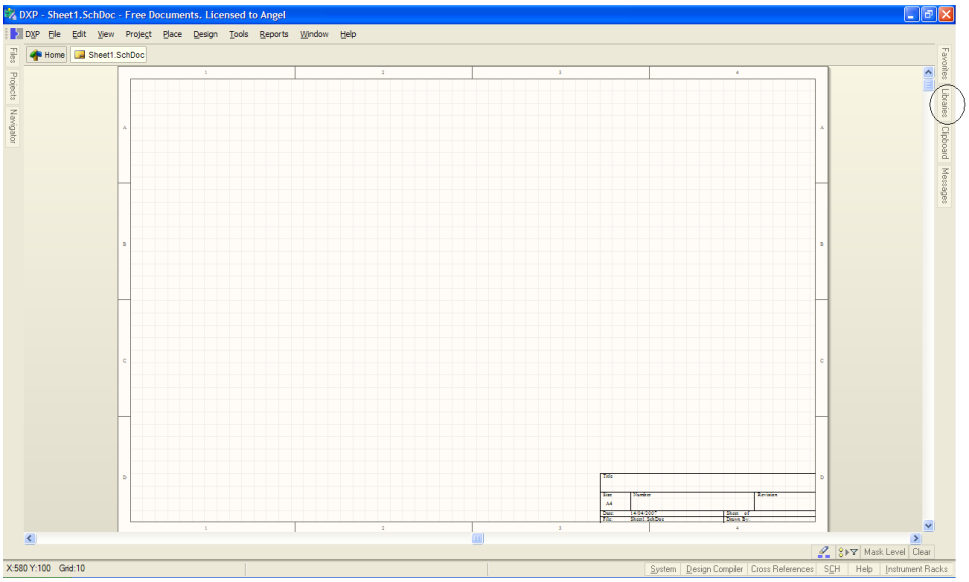


**Figura 3.31** *Página de Esquemático*

En esta Hoja se realiza el diseño del Circuito Electrónico completo mediante el uso de Librerías las cuales se encuentran en las barra de herramientas de la derecha.

Se puede recobrar las barras de herramientas en el lugar que tienen por defecto (Default) mediante el siguiente procedimiento:

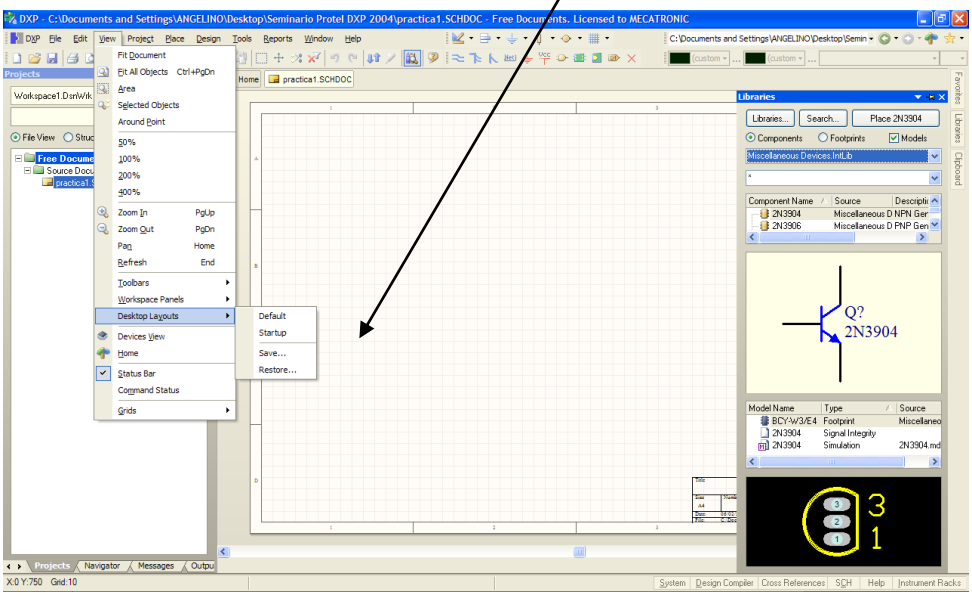


**Figura 3.32** *Herramienta Default*

El diseño de una hoja Esquemática se guarda con la extensión *.SCHDOC

### 3.12.3 Configurar el documento para el Proyecto:

Creación de la Plantilla:

Abrir plantilla A4 Schematic.

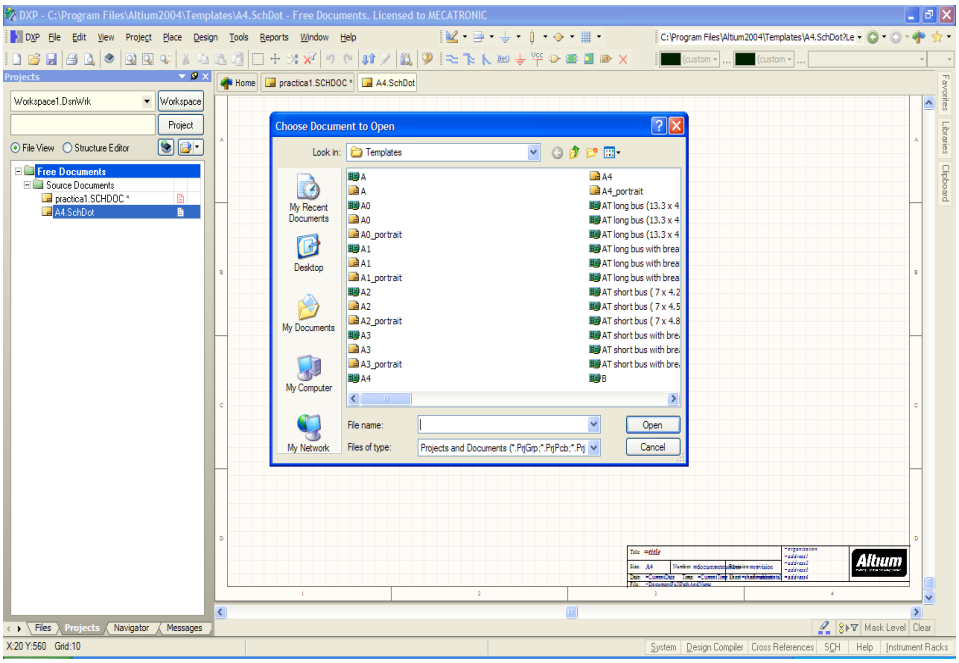



**Figura 3.33** *Creación de plantilla*

Menú: File ⟶ Open ⟶ C:\Program Files\Altium2004\Templates

Guardar el documento como plantilla *.DOT

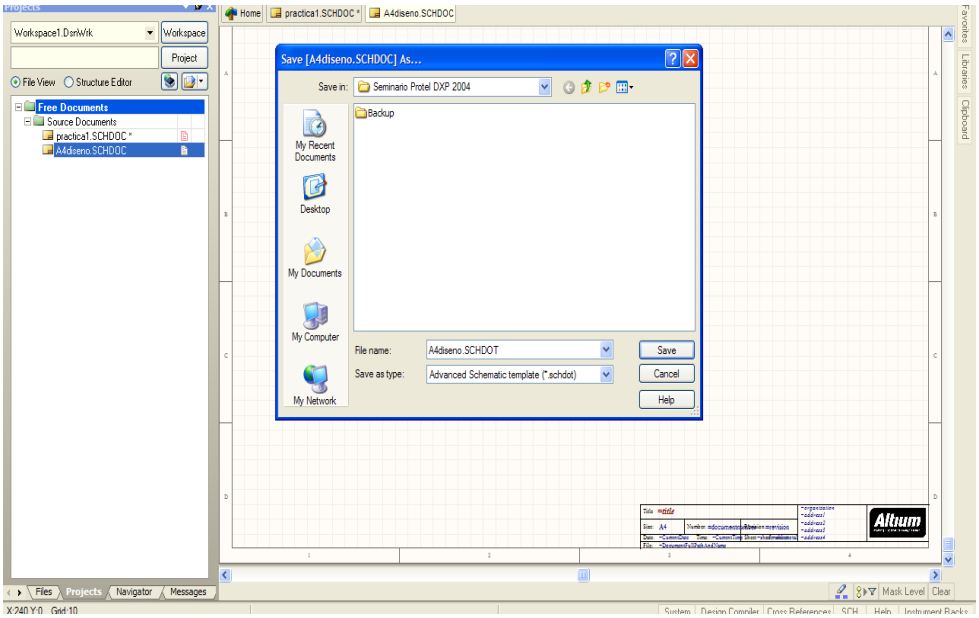



**Figura 3.34** *Guardar documento *.DOT*

Editar mediante las Preferencias del Esquemático:

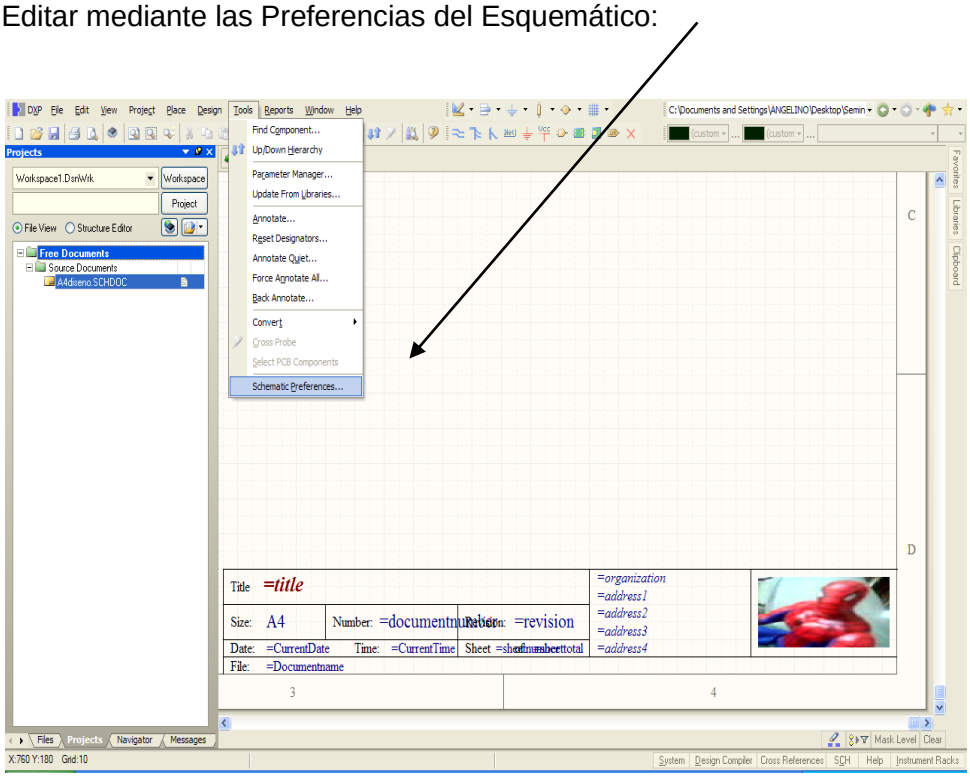



**Figura 3.35** *Preferencias del esquemático*

En la pestaña Graphical Editing verificar la casilla Convert Special Strings:

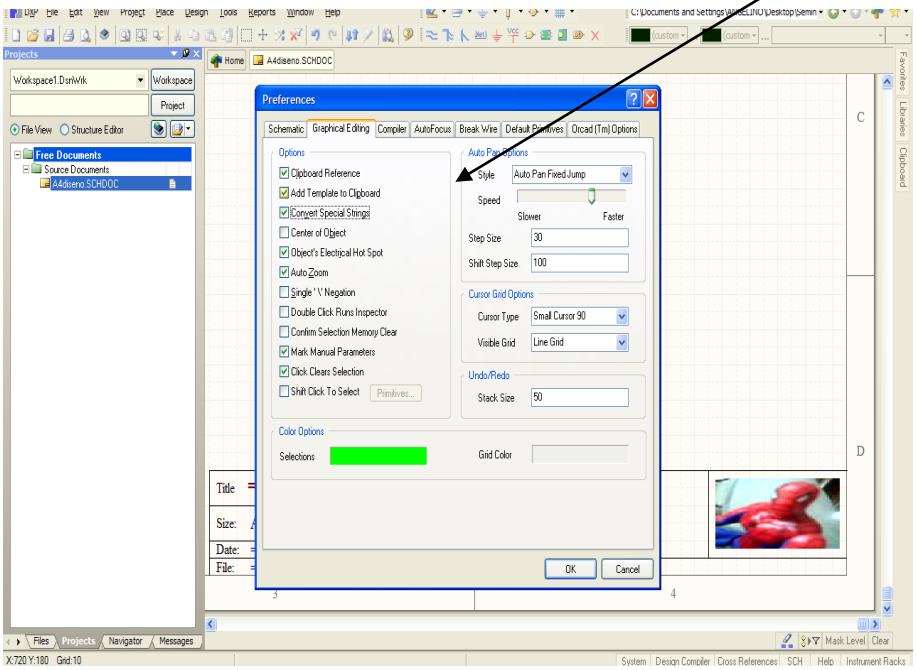
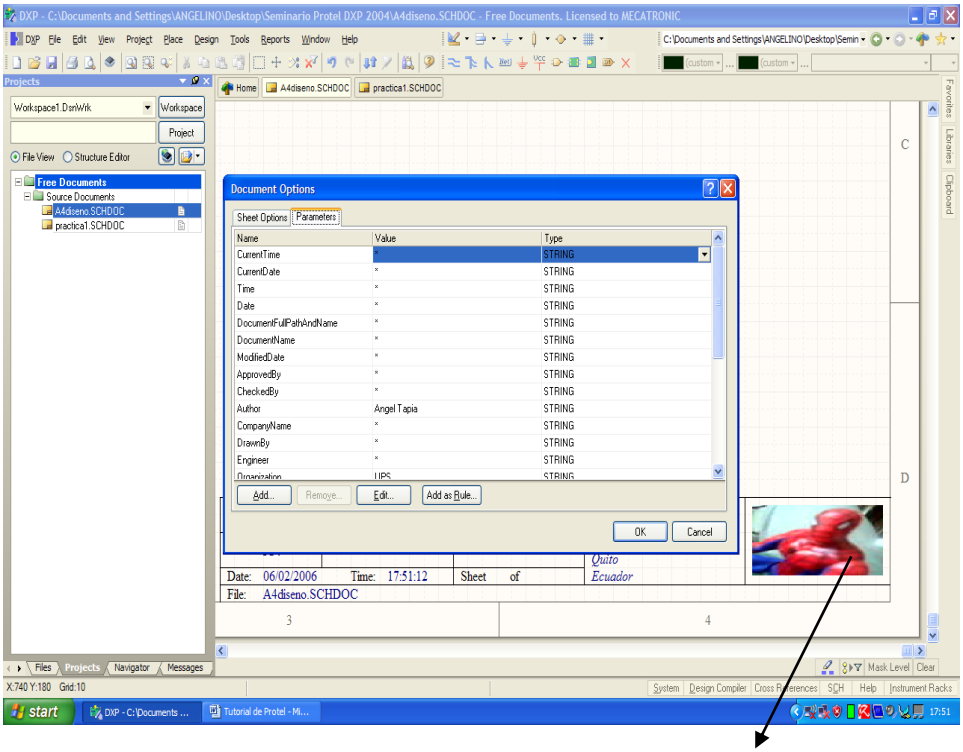



**Figura 3.36** *Preferencias*

En el submenú Document Options editar el rotulado completamente:




Colocar aquí el gráfico para la plantilla

**Figura 3.37** *Opciones del documento*

Menú: Design ⟶ Document Options

### 3.12.4 Esquemático

El Diseño Esquemático (Schematic) se lo realiza mediante la inserción de los componentes que se encuentran en las librerías y se conectan mediante cables o puertos:

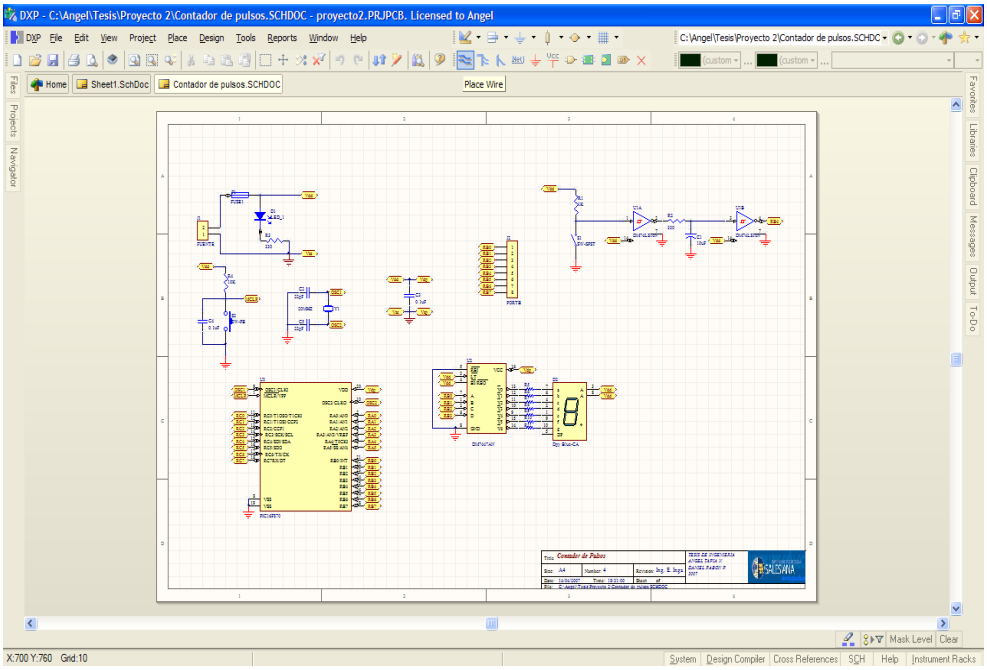



**Figura 3.38** *Página de esquemático*

Observaciones:

- Si se observa un subrayado zigzag rojo debajo de algún puerto o conector se sugiere cambiar la configuración del Terminal del elemento y/o del puerto a modo bidireccional en las Propiedades:

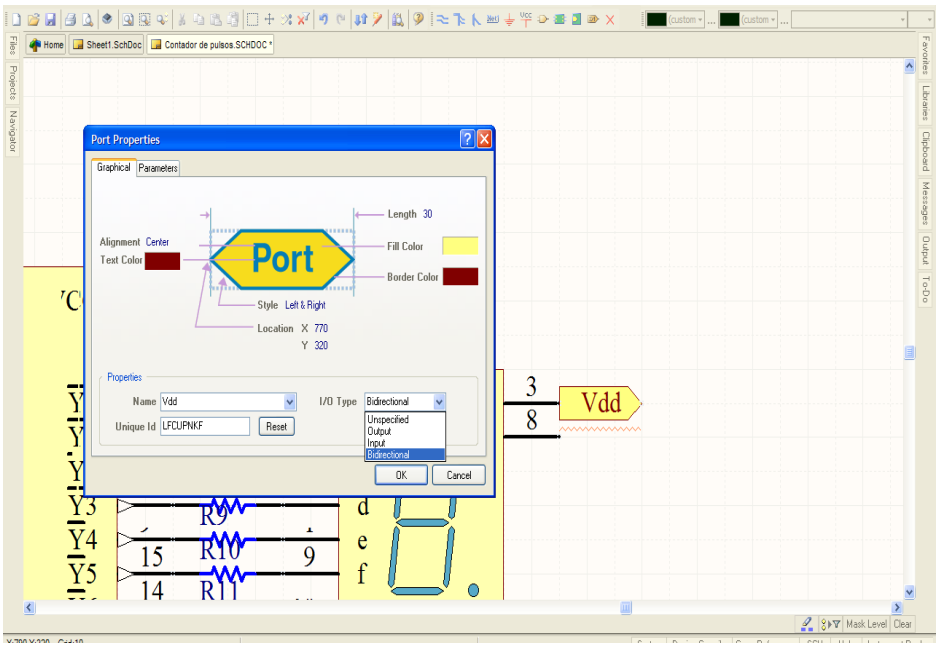



**Figura 3.39** *Propiedades del puerto*

- Todos los nodos negativos deben ir a Tierra (GND):

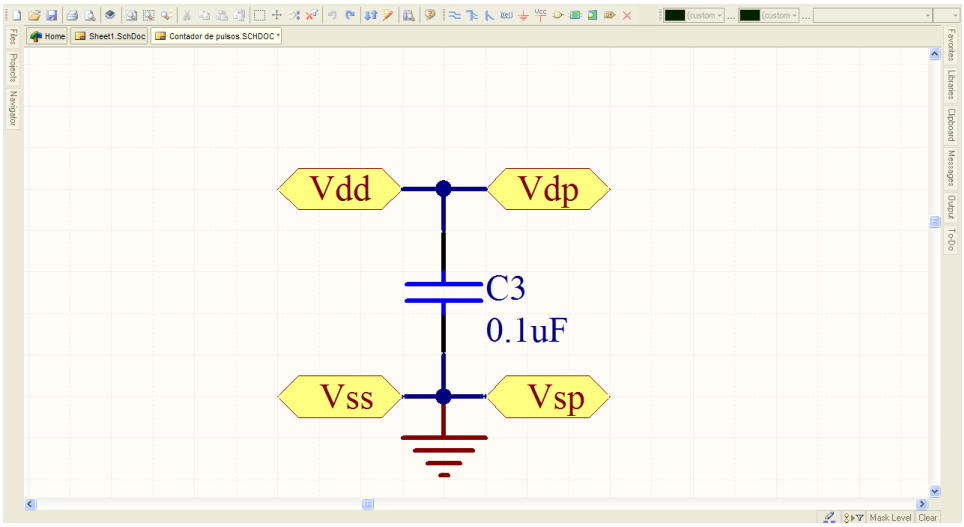



**Figura 3.40** *Conexión típica de GND*

### 3.12.5 Anotate

Herramienta para enumerar automáticamente los elementos del Circuito Esquemático (Annotate):

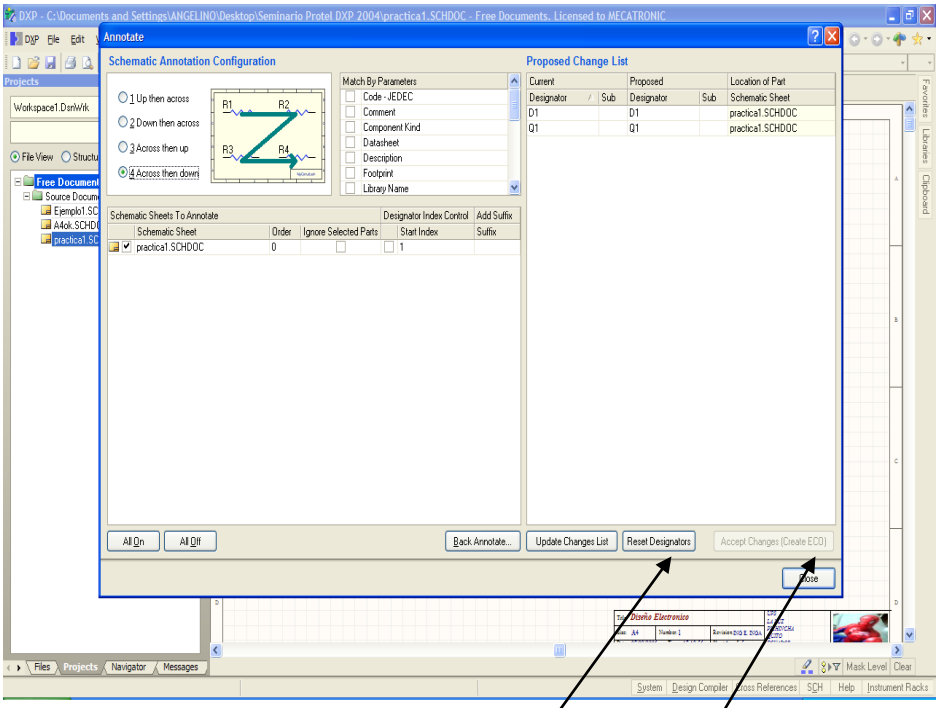



**Figura 3.41** *Opción ANOTATE*

Observaciones:

- Se debe reiniciar los designadores (Reset Designators) antes de actualizar las listas de cambios (Update Change List).
- Cada vez que se agregue o se elimine un componente se debe realizar este procedimiento para que los cambios en el proyecto se actualicen correctamente.

### 3.12.6 Rotulado

Colocar Plantilla A4 (Rotulado) en el diseño esquemático.

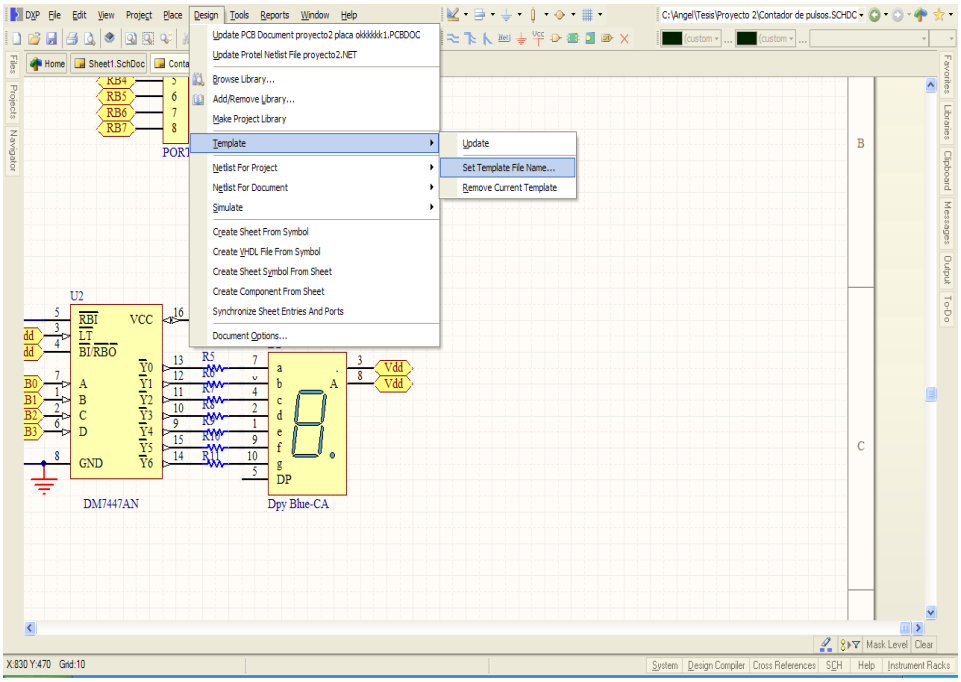



**Figura 3.42** *Colocar Plantilla en el diseño*

Se busca la ubicación de la plantilla guardada anteriormente con la extensión *.DOT y automáticamente se actualiza la platilla con el rotulado:

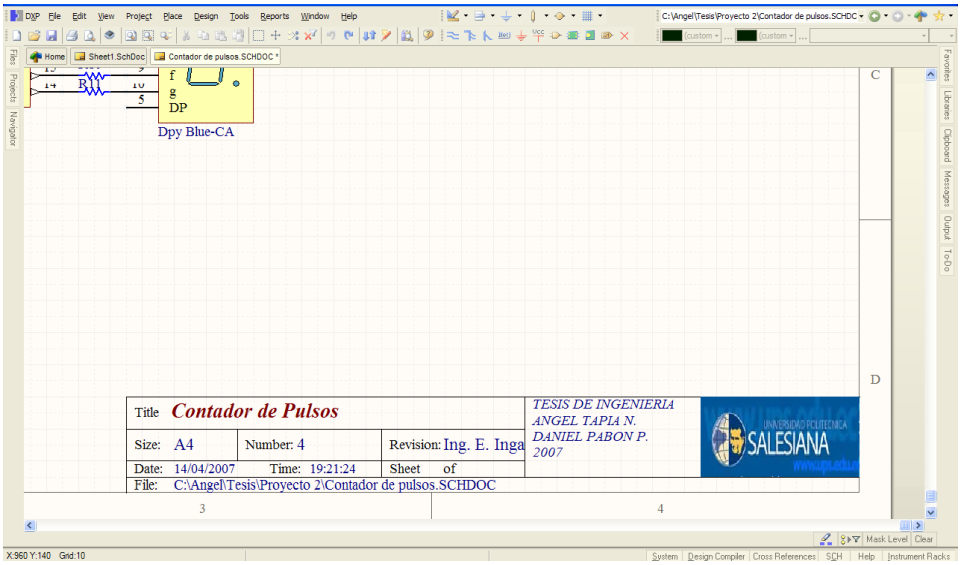



**Figura 3.43** *Plantilla insertada*

### 3.12.7 Diseño de lista de uniones (netlist).

1. Crear un NetList para la hoja esquemática creada anteriormente.

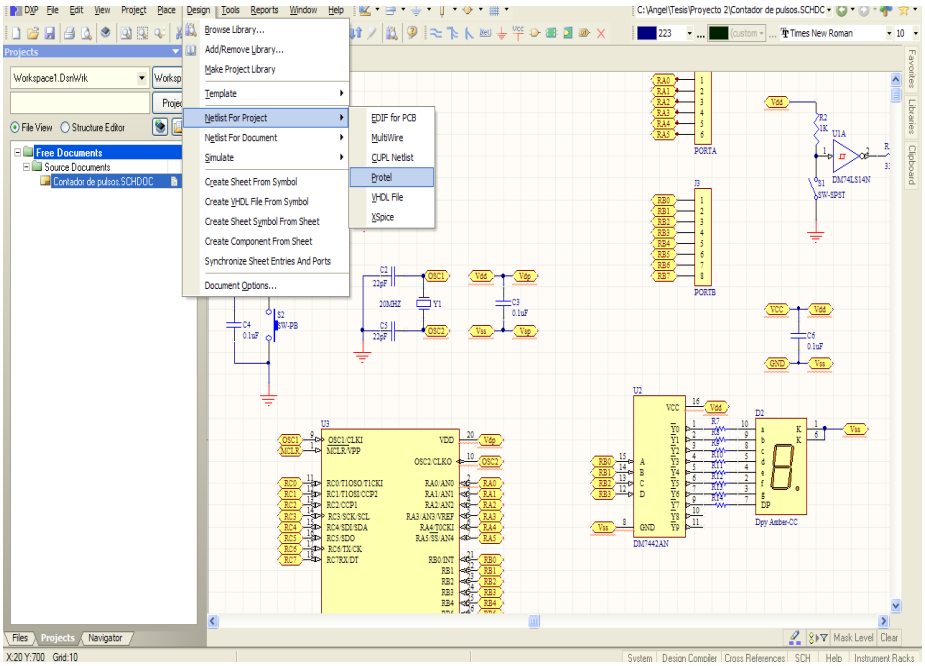



**Figura 3.44** *Creación del NETLIST*

### 3.12.8 Diseño de proyecto (prjpcb).

- Crear un nuevo Proyecto PCB PROJECT que contiene los archivos: Esquemático (Schematic) y Lista de Uniones (Netlist).

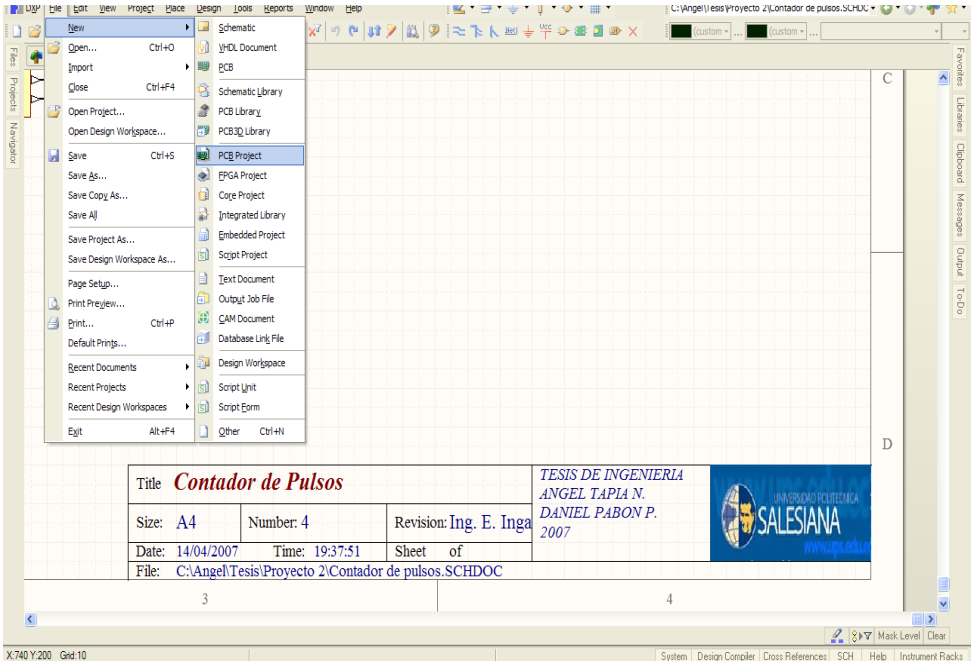



**Figura 3.45** *Diseño de proyecto*

- Arrastrar la hoja esquemática (*.SCHDOC) y la lista de uniones (*.NET) en la barra de herramientas PROJECT (Izquierda) hacia el Proyecto nuevo.

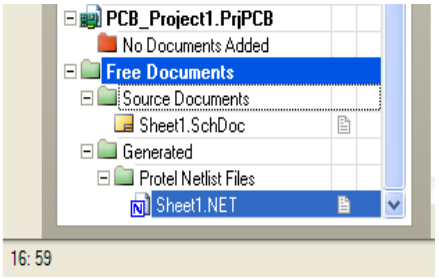



**Figura 3.46** *Barra de herramientas PROJECT*

### 3.12.9 Diseño De Placa Impresa (*.PCB).

- Ingresar a Pcb Board Wizard para diseñar el circuito impreso paso a paso:

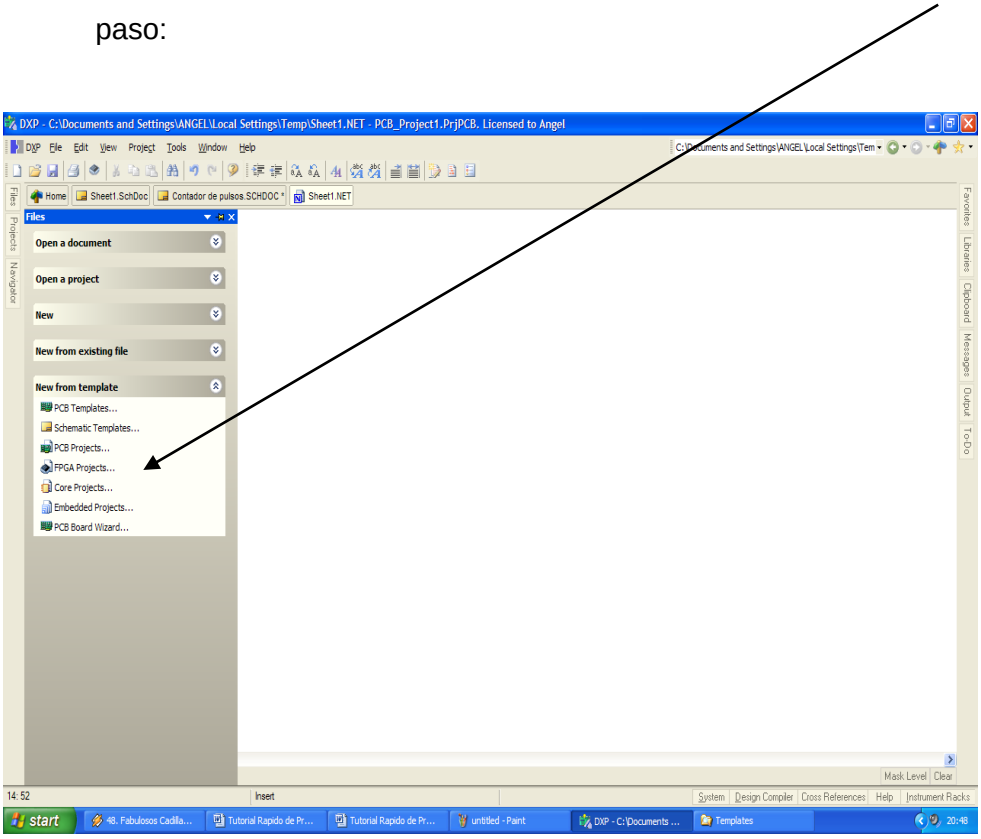



**Figura 3.47** *Creación de PCB-BOARD WIZARD*

- Utilizar medidas Imperial (Pulgadas) para configurar en milésimas de pulgada las dimensiones de la placa impresa y luego Personalizar (Custom).

- Utilizar las siguientes medidas por defecto (Default) para el diseño de la placa:

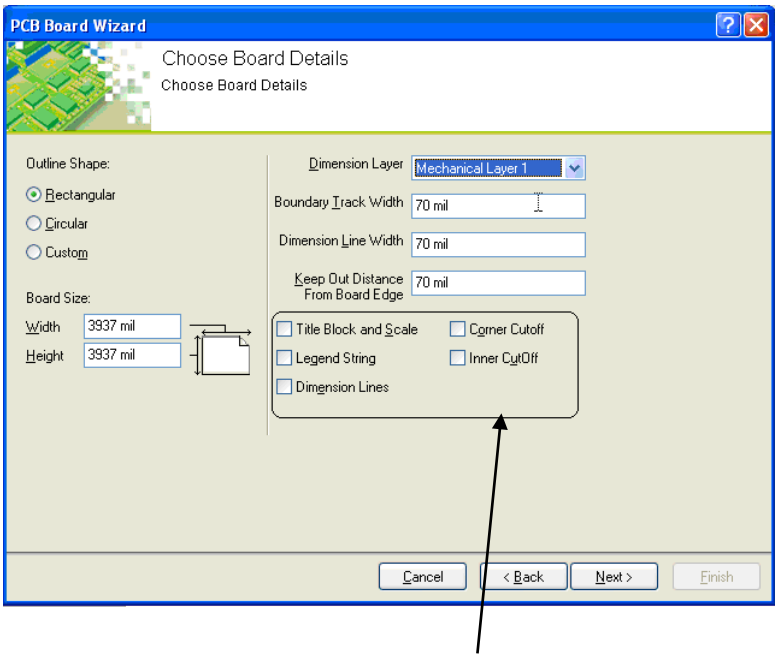



**Figura 3.48** *Opciones de PCB Board Wizard*

Quitar la verificación en las casillas siguientes:

**Observaciones:**

- Las unidades para las medidas desde este punto se toman en milésimas de pulgada (**mil**) para lo cual se debe hacer un cálculo dependiendo de las dimensiones de la placa a realizar: 1000mil = 1 pulgada

Número de capas y Planos activos:

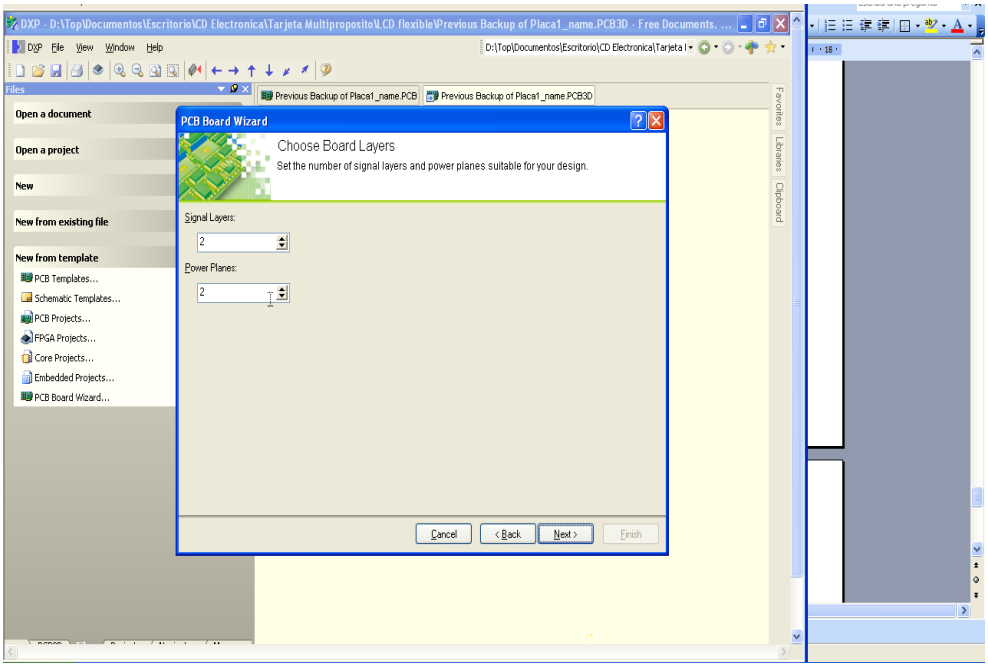



**Figura 3.49** *Número de capas y Planos activos*

- Realizar Vías blindadas: con agujeros metalizados.

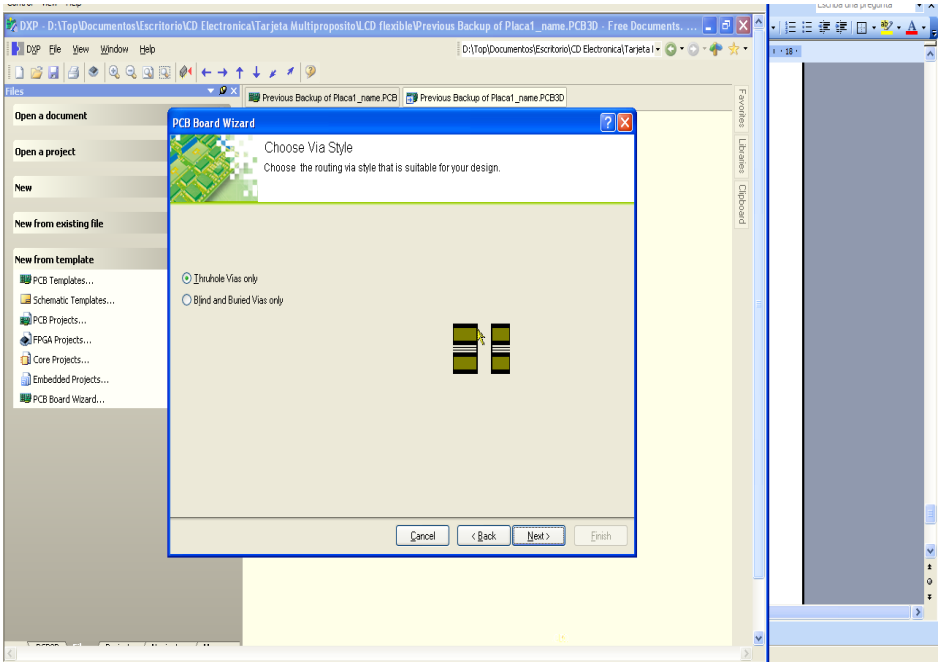



**Figura 3.50** *Vías Blindadas*

- Utilizar la siguiente configuración para que los hoyos permanezcan entre los componentes y solo exista una vía entre nodos:

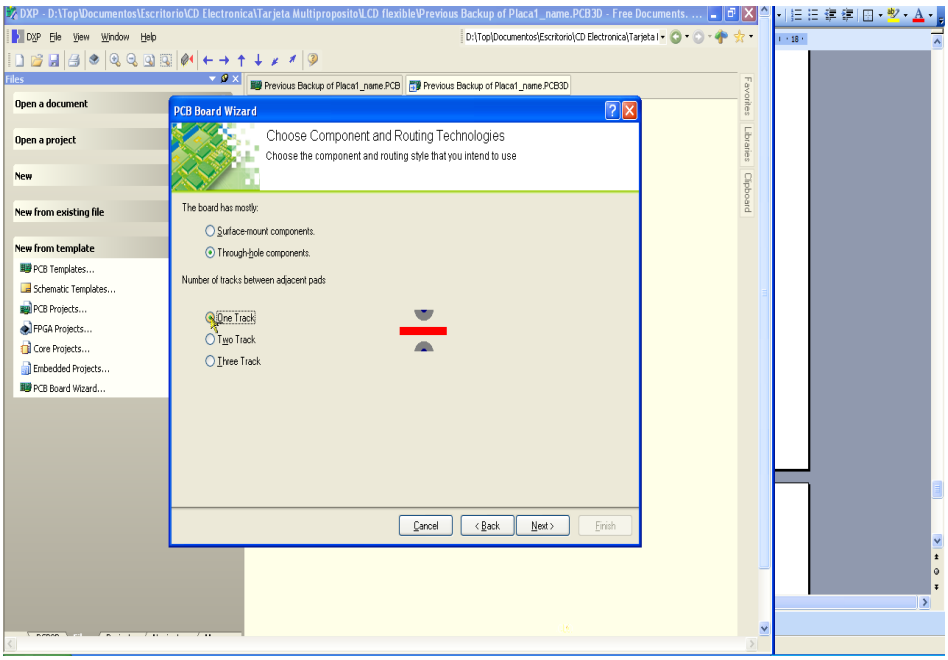



**Figura 3.51** *Una vía entre nodos*

- Distancia de hoyos recomendada:

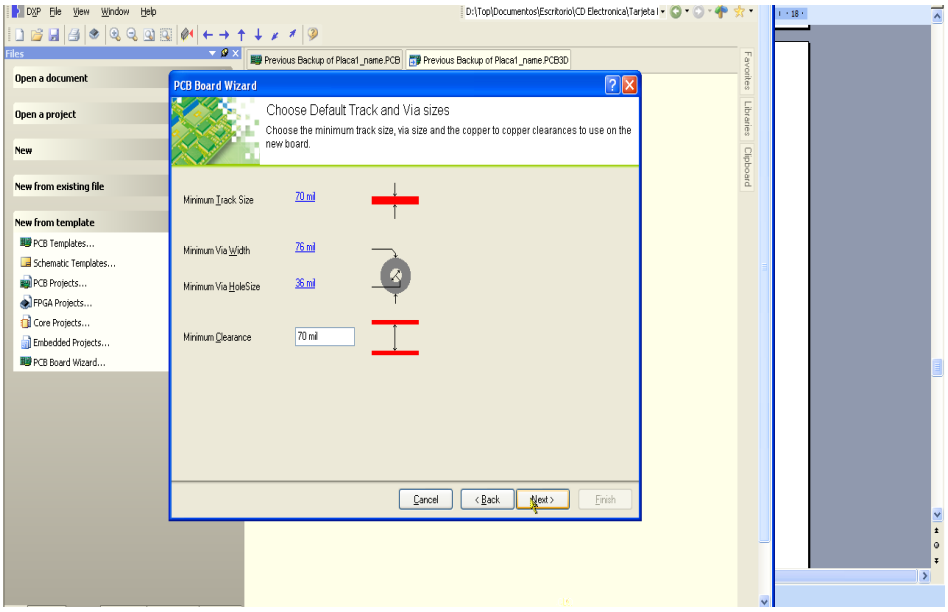



**Figura 3.52** *Distancia de hoyos recomendada*

### 3.12.10 Compilación De Placa Impresa.

- Se arrastra el documento PCB (*.PRJPCB) hacia el Proyecto en la barra de herramientas PROJECT.

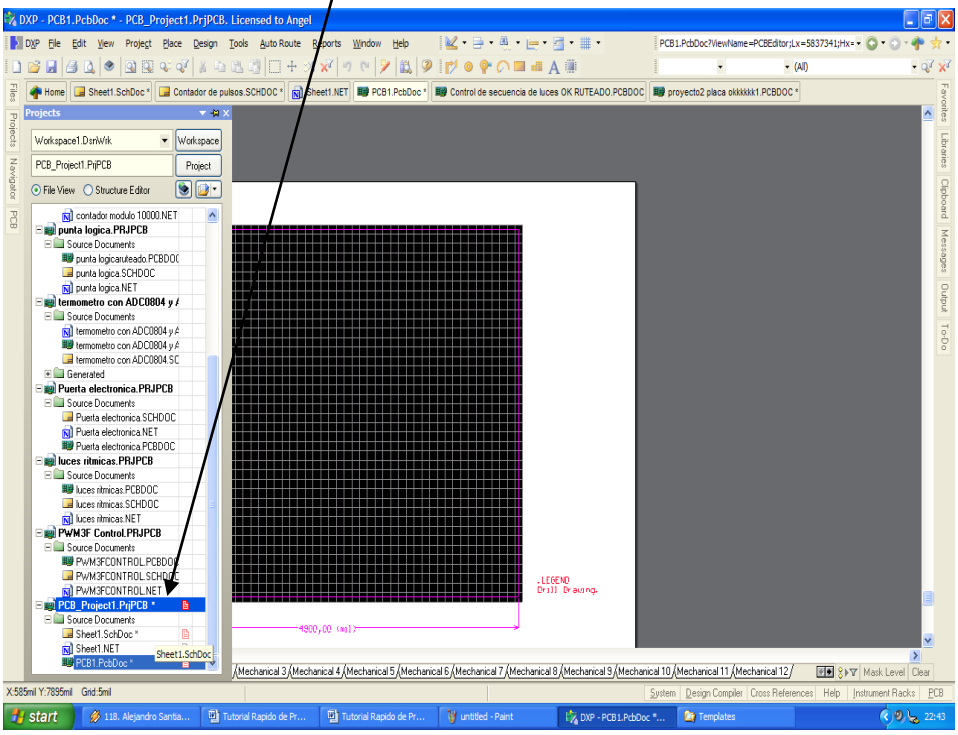


**Figura 3.53** *Barra de herramientas Project*

- Compilar el documento PCB (*.PcbDoc)

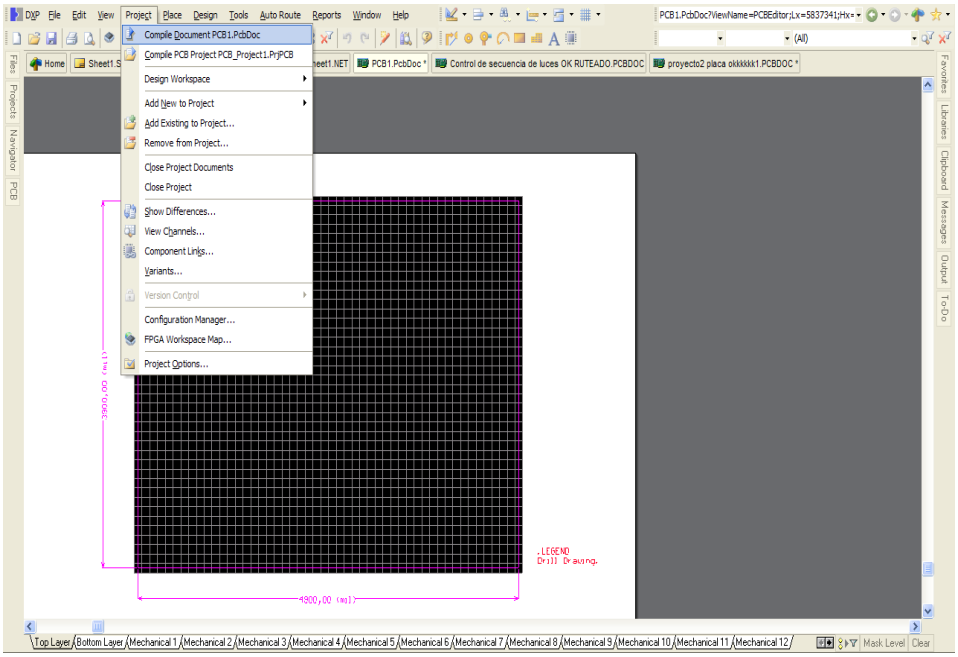



**Figura 3.54** *Compilar el documento*

- Compilar el Proyecto (*.PRJPCB)

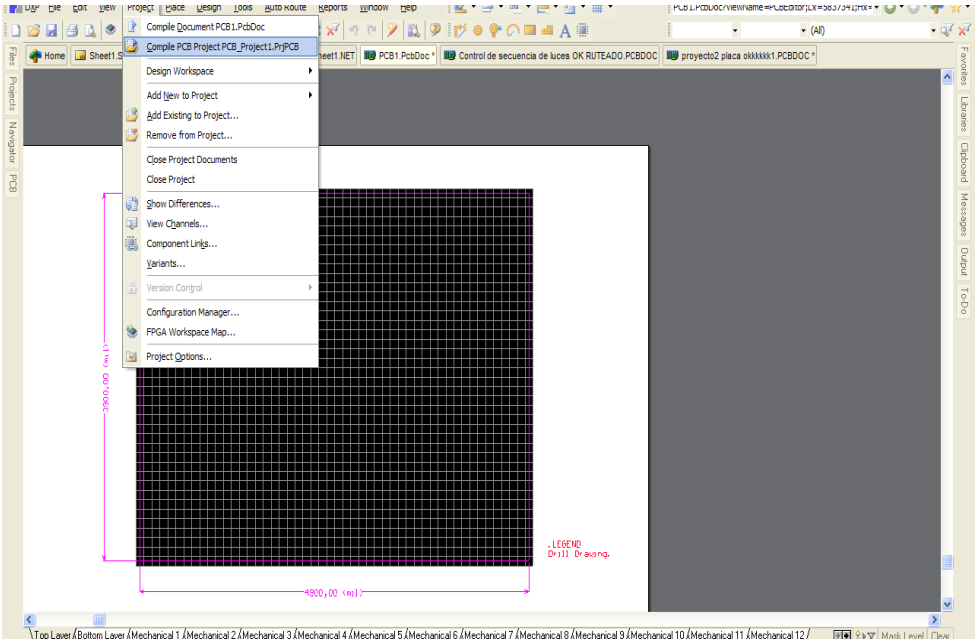



**Figura 3.55** *Herramienta Compilar*

- Actualizar la Lista de Uniones (NetList) en Protel.

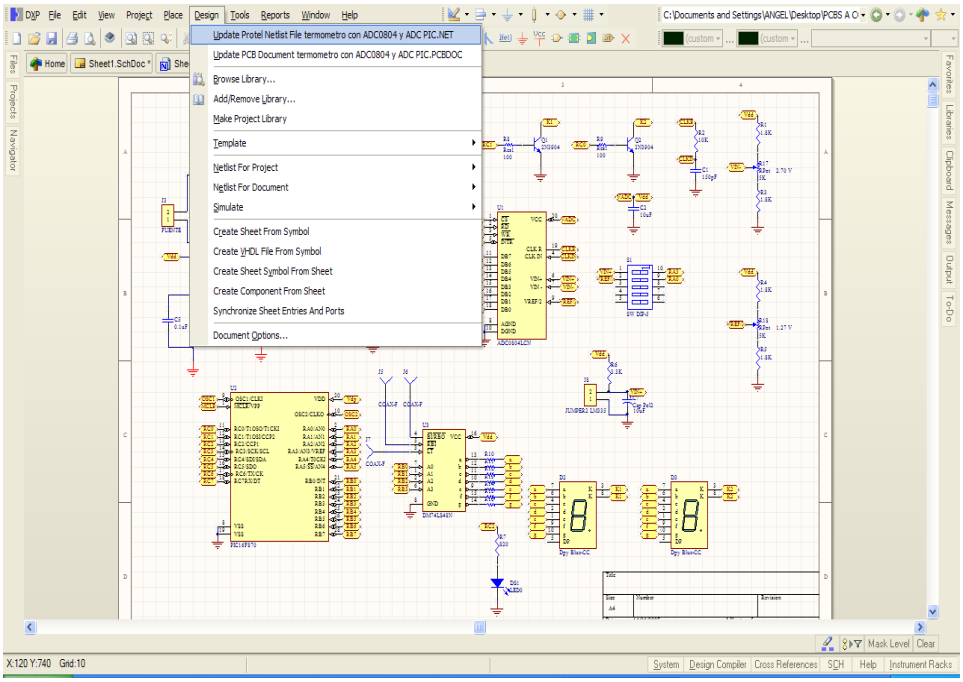



**Figura 3.56** *Actualizar la Lista de Uniones*

- Actualizar el documento PCB en Protel para generar la placa impresa.

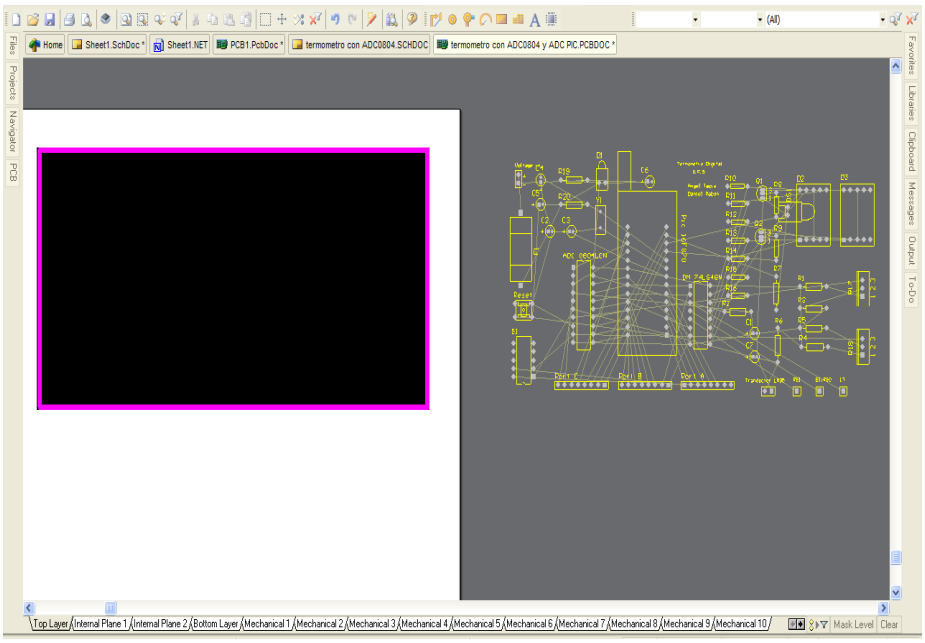


**Figura 3.57** *PCB Compilado*

Observación:

- La línea de precaución (Keep Out Layer) es indispensable para mantener la distancia de ruteo en los bordes de la placa.
- Disponer los componentes dentro de la placa manualmente con la ayuda del puntero (Mouse) y la barra de herramientas Aligned Tools:

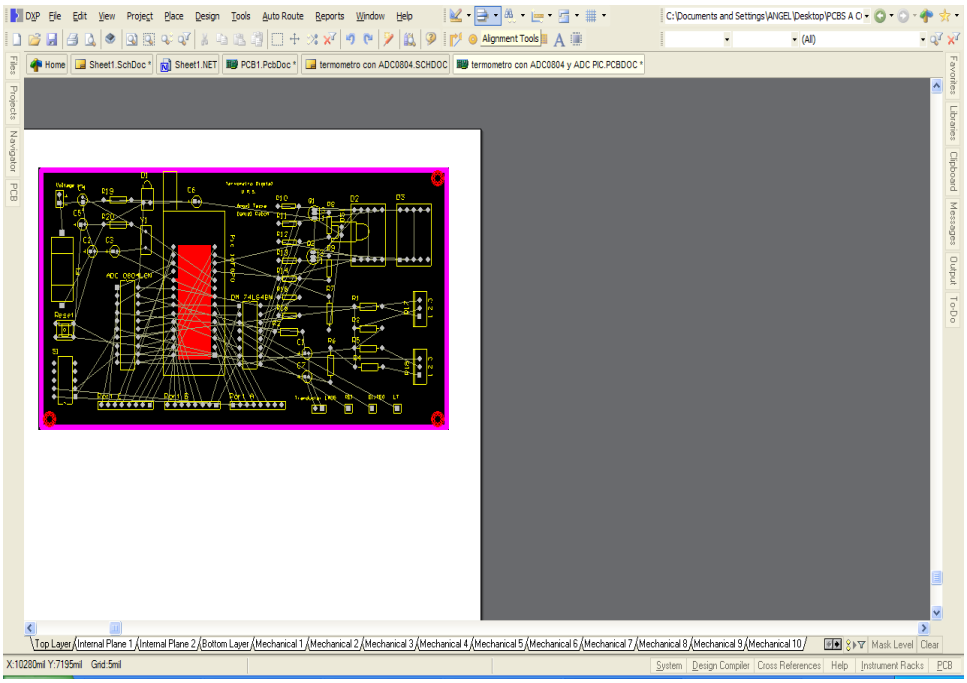


**Figura 3.58** *PCB Ordenado*

- Colocar una Malla de Tierra debajo de los componentes que generen ruido (Microcontroladores) mediante un plano de tipo Polígono (Place Polygon Plane).

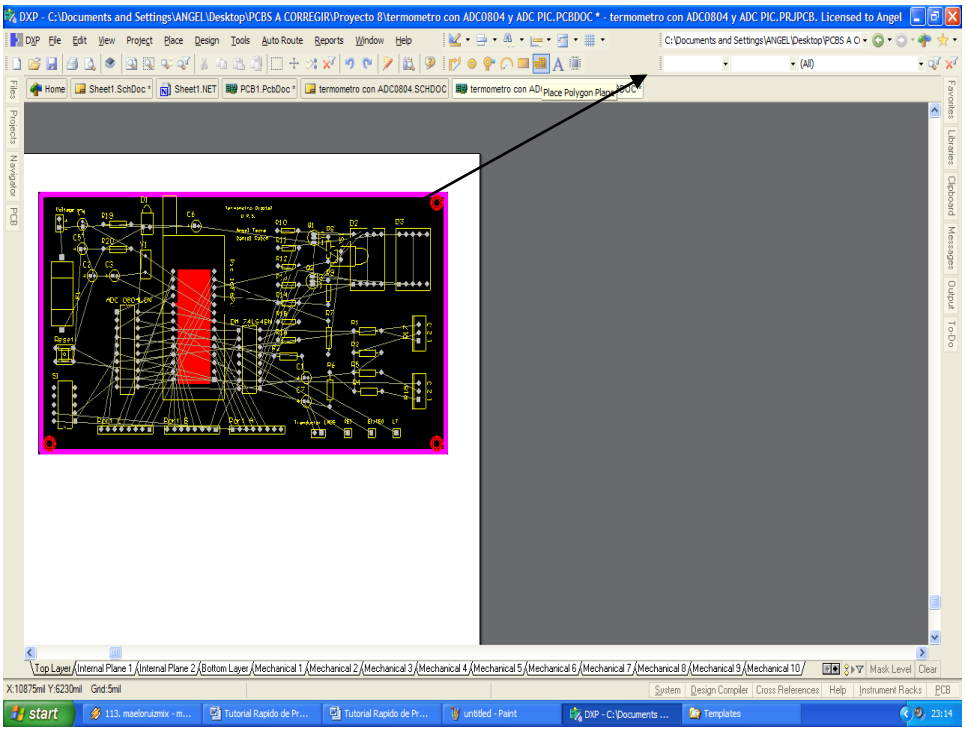


**Figura 3.59** *Colocar malla a tierra*

Colocar las siguientes configuraciones en las propiedades del polígono:

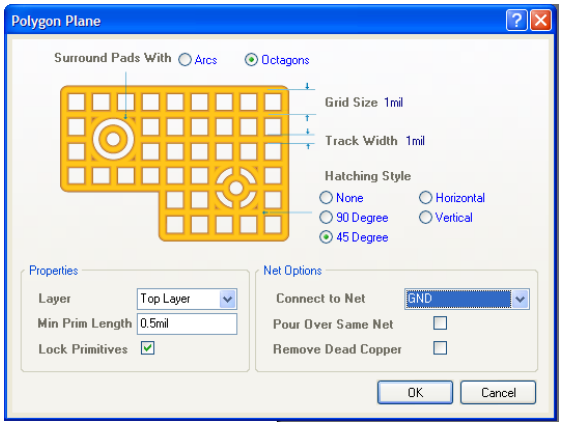



**Figura 3.60** *Opciones de malla Polygon Plane*

- Utilizar la herramienta de Autoruteo y configurar la distancia mínima de distancia del pad a las vías (Clearance) y el espesor de la vía (Width):

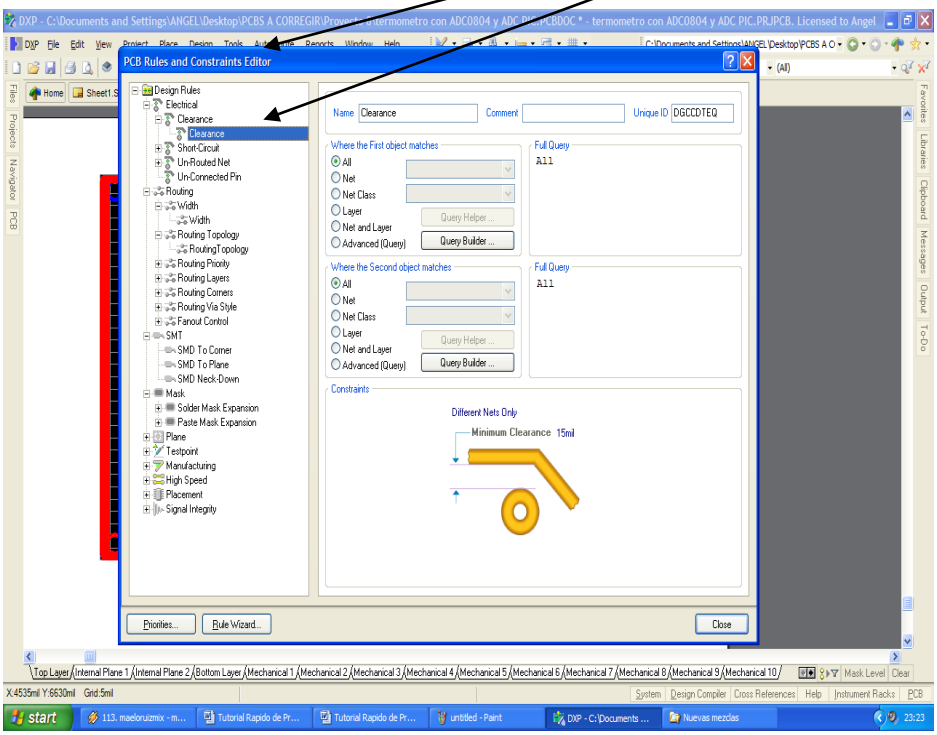



**Figura 3.61** *Opciones de Ruteo*

Espesor y distancia recomendados:

Clearance = 15mil

Width = 37mil

- Rutear toda la placa con las dimensiones establecidas:

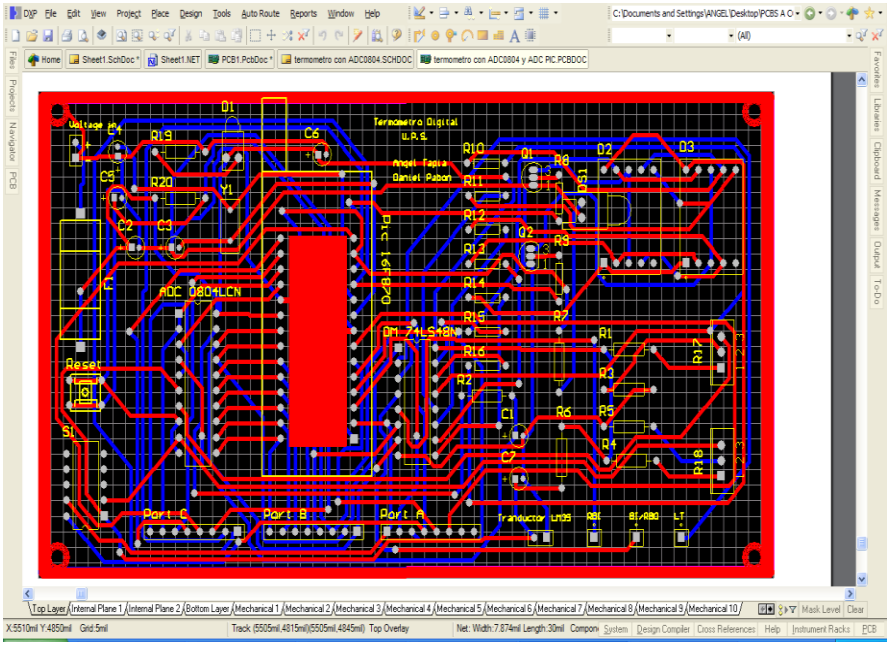


**Figura 3.62** *PCB Ruteado*

### 3.12.11 Lista de materiales.

- Generar la Lista de Materiales.

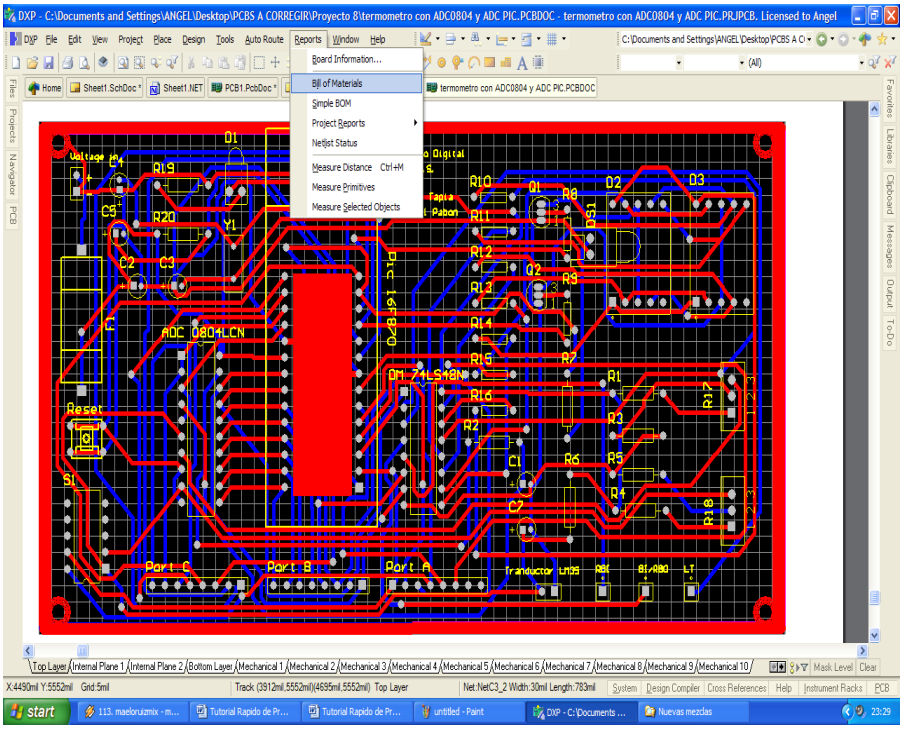



**Figura 3.63** *Generar lista de materiales*

Menú: Reports ⟶ Bill of Materials

- Exportar a la Hoja de Cálculo la Lista de Materiales;

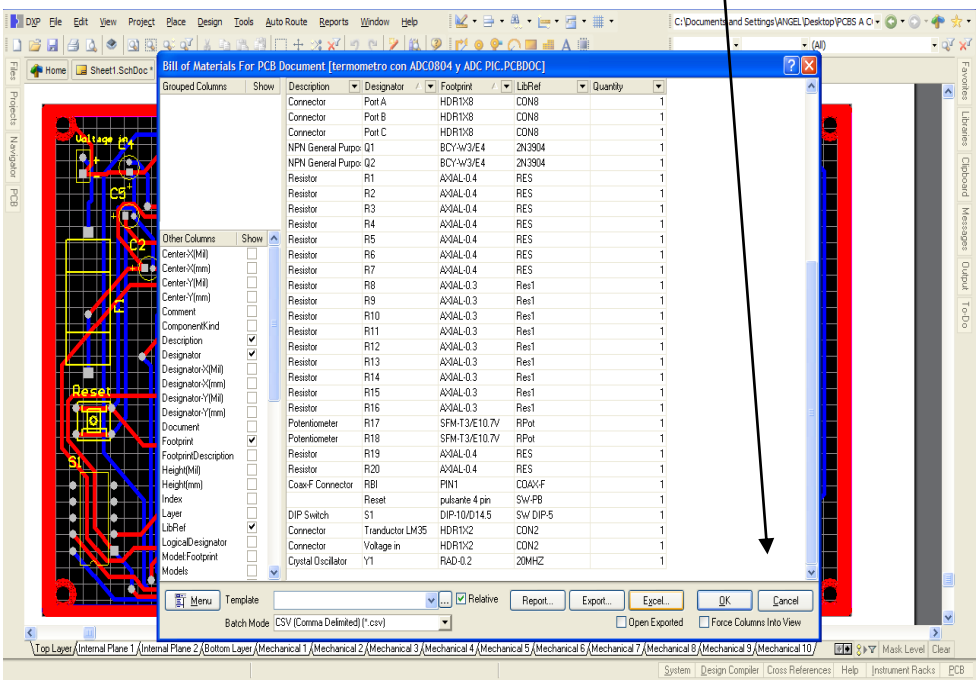



**Figura 3.64** *Exportar la lista de materiales*

## 3.13 Tutorial De Proteus.

### 3.13.1 Introducción

La utilización de programas de software para la simulación de fenómenos físicos es una práctica habitual en el mundo de la ingeniería, sea cual sea su especialidad. El conocimiento preciso del funcionamiento de un diseño real antes de su fabricación es la gran aportación de los ordenadores. Todos estos paquetes de software están basados en modelos matemáticos, siendo la tarea del hombre decidir si los datos introducidos y los resultados obtenidos son satisfactorios o no.

La misión de un simulador de circuitos electrónicos es reproducir lo más exactamente posible el comportamiento de un determinado circuito electrónico, sin necesidad de construirlo físicamente, con el consiguiente ahorro de dinero y tiempo.

En el caso de la simulación del comportamiento de un circuito electrónico con Proteus, los pasos a seguir son los siguientes:

### 3.13.2 Primeros pasos para el diseño

En primer lugar se debe dibujar el esquema electrónico del circuito a simular. Para que la simulación resulte cierta, se debe contar con el correspondiente modelo «spice» de todos los componentes utilizados. Un modelo spice no es más que un fichero que contiene la información necesaria para que el simulador pueda reproducir el comportamiento de dicho componente. Proteus se suministra con una amplia librería de más de 6.000 elementos con su correspondiente modelo spice. Aunque Proteus permite la creación por parte del usuario de nuevos componentes con modelo spice no incluidos en sus librerías estándar...

En segundo lugar debemos colocar en el esquema electrónico aquellos generadores de señal que definamos como entradas de nuestro circuito.

En tercer lugar colocaremos tantas sondas como se consideren necesarias para conocer las señales resultantes que se definan como salidas del circuito.

Proteus permite la utilización de herramientas gráficas para facilitar la generación de las señales y la visualización de los datos resultantes. Sin embargo estas herramientas sólo son utilidades para facilitar esta labor. Los conceptos básicos siempre seguirán siendo la utilización de generadores de señales para simular las entradas y de sondas para visualizar las señales de salida.

En cuarto y último lugar, una vez que se haya dibujado el esquema electrónico con los correspondientes generadores y sondas, se procederá a la simulación del funcionamiento del circuito mediante la utilización del panel de control de animación.

### 3.13.3 Realización de un Circuito de Prueba

Para la primera simulación se realizará un sencillo montaje de un divisor de tensión compuesto por dos resistencias, una de 10k y otra de 100k en serie. En ISIS crearemos el esquema recogido en la **figura 3.65.**

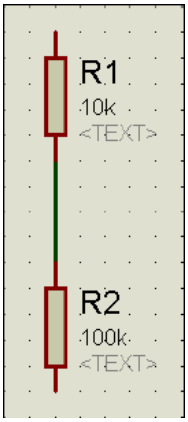



**Figura 3.65** *Conexionado en Proteus*

En segundo lugar se definen dos señales de entrada. Una señal de corriente alterna de 48V y una puesta a tierra. Para ello, en la barra de herramientas, se selecciona la herramienta: Terminales.

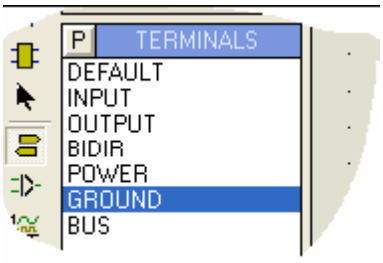



**Figura 3.66** *Barra de herramientas terminales*

Se colocará un terminal tipo «input» en la parte superior y un terminal tipo «ground» en la inferior. El resultado final debe ser como el mostrado en la siguiente figura 3.67:

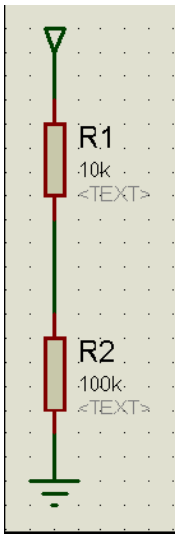



**Figura 3.67** *Conexionado en Proteus*

A continuación se colocará un generador de señal en el terminal de entrada de la parte superior. Para ello en la barra de herramientas, se seleccionará la herramienta generadores.

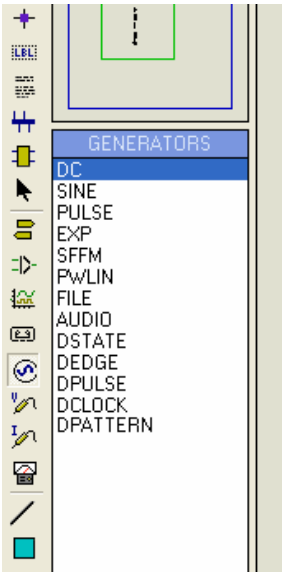



**Figura 3.68** *Barra de herramientas Generators*

En la ventana de generadores, se escoge la opción «sine» y con el ratón nos situamos sobre el «cable» que une el terminal de potencia y la resistencia R1, y se pulsa en el botón izquierdo para colocar un generador en dicho punto.

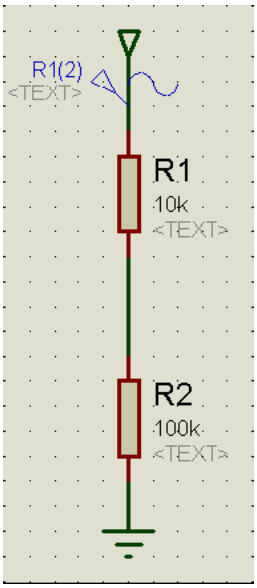



**Figura 3.69** *Conexionado en Proteus terminado*

Se debe situar sobre el generador que acabamos de crear y se pulsa una vez el botón derecho del ratón para seleccionarlo. Al seleccionarlo el generador cambia de color. Entonces se pulsa el botón izquierdo del ratón y se abre la ventana de opciones. En esta ventana se deja las opciones disponibles como se muestran en la imagen inferior Figura 3.70:

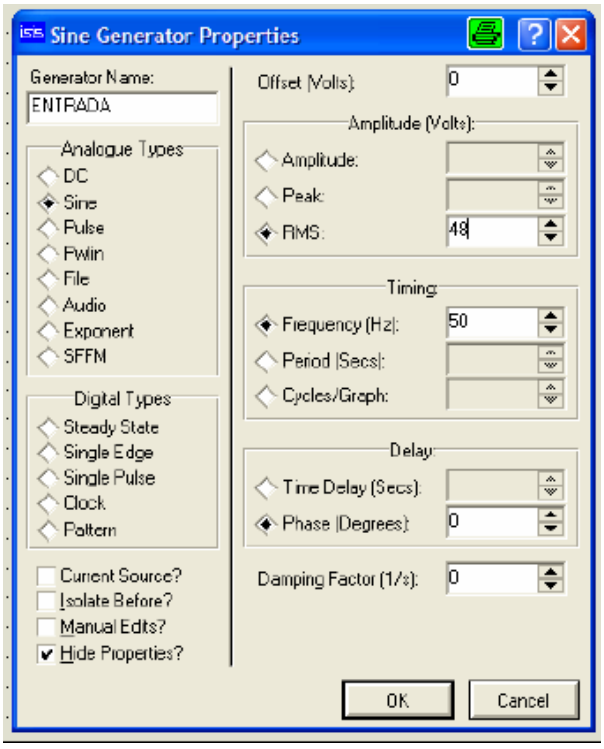



**Figura 3.71** *Propiedades del generador de funciones*

Se debe comprobar que se han introducido los datos correctamente en las casillas Generator name», «Sine», «RMS» y «Frecuency». Con ello se le ha dado el nombre ENTRADA al generador, se le esta diciendo que es una señal de tipo senoidal (alterna) con valor RMS de 48 Voltios y una frecuencia de 50Hz. Pulse en el botón «Ok» para terminar y podrá ver que en el esquema electrónico la sonda ya tiene el nombre ENTRADA asignado.

En tercer lugar se va a colocar una sonda entre las dos resistencias para conocer la tensión de salida resultante de nuestro divisor de tensión. Para ello en la barra de herramientas, se debe elegir la herramienta sonda de tensión.

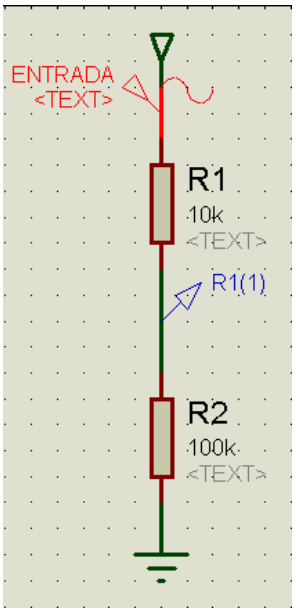



**Figura 3.72** *Conexionado en Proteus con sonda de tensión*

Se pulsa con el botón derecho del ratón sobre la sonda para seleccionarla (cambiará de color) y una vez seleccionada en el botón izquierdo. Nos aparecerá la ventana de opciones.

Solo tenemos que introducir el nombre SALIDA para la sonda, tal y como se muestra en la siguiente imagen.

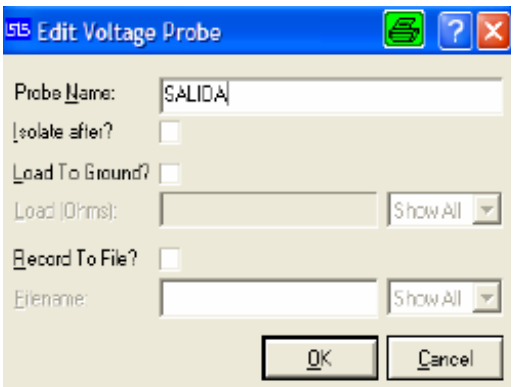



**Figura 3.72** *Ventada de edición de voltaje*

El resultado final debe ser como el mostrado en la siguiente figura:

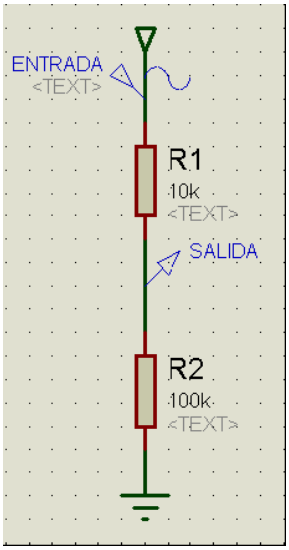



**Figura 3.73** *Conexionado en Proteus final*

En cuarto y último lugar se iniciará la simulación. Para ello se utilizará el control de animación situado en la parte inferior izquierda del área de trabajo.


**Figura 3.74** *Herramientas de simulación*

El control de animación es muy similar a los mandos de cualquier reproductor. Tiene cuatro botones: reproducir, reproducir un salto, hacer una pausa y parar la simulación. En este momento sólo vamos a utilizar los botones reproducir y parar. Para comenzar la simulación pulsaremos sobre el primero de ellos.
Se puede observar que en la sonda que llamamos SALIDA se visualizan los valores medidos. Se podrá comprobar que es muy difícil leer los resultados porque están variando constantemente.

En primer lugar se puede variar la velocidad a la que se realiza la simulación. Proteus nos permite definir cuántos fotogramas vamos a visualizar cada segundo (hasta un máximo de 50) y cuánto tiempo va a durar cada fotograma. Así, por ejemplo, si se dice que cada fotograma dura 50 mseg y

que se va a ejecutar 20 fotogramas por segundo, se tendrá una simulación en tiempo real (20 fot/seg * 0,05 seg = 1 seg).

En cambio, si se va a ejecutar 1 fotograma por segundo y que cada fotograma dura 50 mseg, tendremos una especie de cámara lenta, puesto que cada segundo real de tiempo sólo se ejecutará una simulación de 50 mseg. De la misma forma, si se va a ejecutar 20 fotogramas por segundo y que cada fotograma dura 1 microseg, también se tendrá el efecto de cámara lenta.

Con esta filosofía, Proteus permite simular cualquier proceso por rápido que sea en cualquier ordenador por lento que sea sin perder datos durante la simulación.

Para configurar en Proteus los fotogramas por segundo y el tiempo de cada fotograma, se tendrá que ir a la opción de menú System -> Set animations options… Se abrirá la siguiente ventana de configuración:

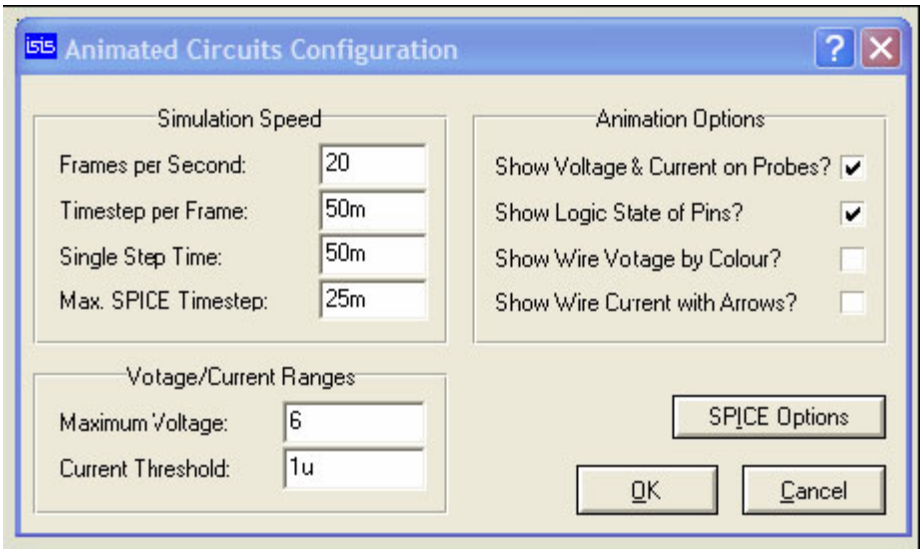



**Figura 3.75** *Ventana Configuración de animación de circuitos*

Con el objeto de facilitar la percepción de la velocidad real del proceso simulado, cuando la simulación está en marcha, Proteus utiliza la barra inferior de mensajes para indicarnos el tiempo real del proceso ejecutado y la carga de la CPU del ordenador donde estamos realizando la simulación. Si la carga es muy cercana al 100%, nuestro ordenador está saturado y deberá

modificarse los parámetros de simulación para corregir este problema y aliviar a la CPU.


**Figura 3.76** *Estado de la barra de herramientas mientras simula de circuito en Proteus*

### 3.13.4 Gráfico al modelo de simulación

Como es lógico, de forma intencionada se ha hecho seleccionar una señal de entrada de corriente alterna de 50 Hz de frecuencia. De esa forma ha podido comprobar que las lecturas resultantes son muy difíciles de leer. En la mayoría de los simuladores SPICE sólo se dispone de un tipo de representación textual como la que acaba de ver. Pero, afortunadamente, Proteus cuenta con unas inmensas capacidades de visualizaciones gráficas para ayudarnos en nuestro trabajo.

En primer lugar se va a incluir en nuestro diseño una gráfica donde se mostrará la evolución de nuestra señal de entrada y los valores medidos por nuestra sonda de salida. Es decir una gráfica valores/tiempo.
Para hacerlo en la barra de herramientas, seleccionaremos la herramienta gráfica.

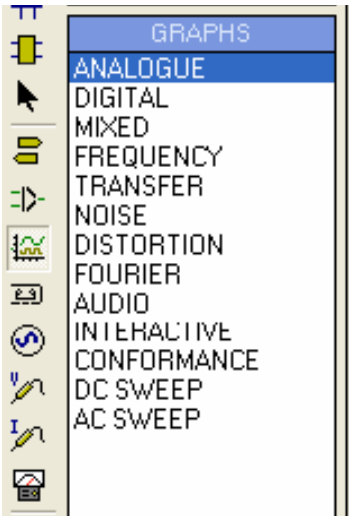



**Figura 3.77** *Barra de herramientas de gráficos*

En la ventana que aparece con los diferentes tipos de gráficas, marcaremos la opción ANALOGUE, puesto que queremos mostrar una gráfica de tipo analógico. Observar que existen tantos tipos de gráficas como análisis SPICE es posible realizar con Proteus. Una vez marcada la opción ANALOGUE coloque el ratón donde quiera que comience la gráfica pulse el botón izquierdo y sin soltarlo señale con el ratón donde quiera que termine la gráfica. El resultado debe ser algo similar a lo mostrado en la figura 3.78:

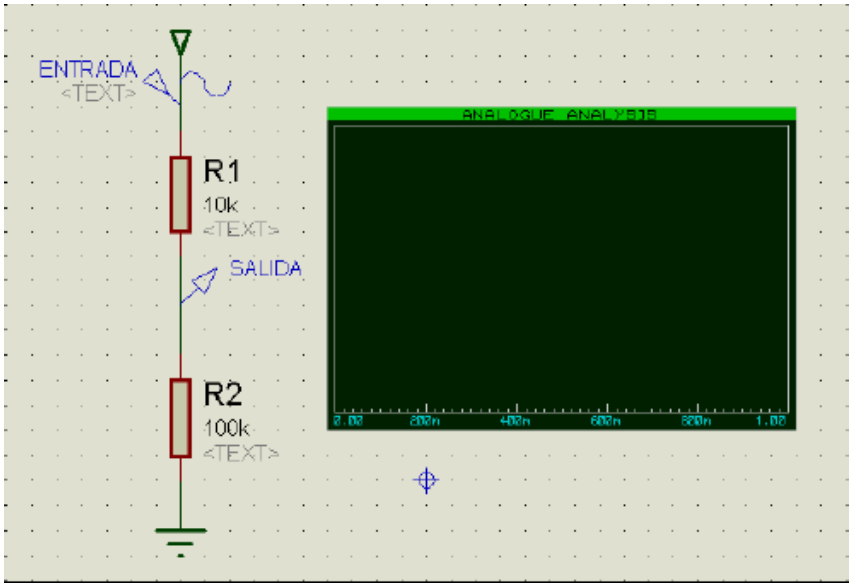



**Figura 3.78** *Ventana para gráficos de Proteus*

Coloque el ratón sobre la gráfica y pulse el botón derecho del ratón para seleccionarla. La gráfica cambiará de color cuando esté seleccionado. A continuación pulse el botón izquierdo y le aparecerá la ventana de opciones.

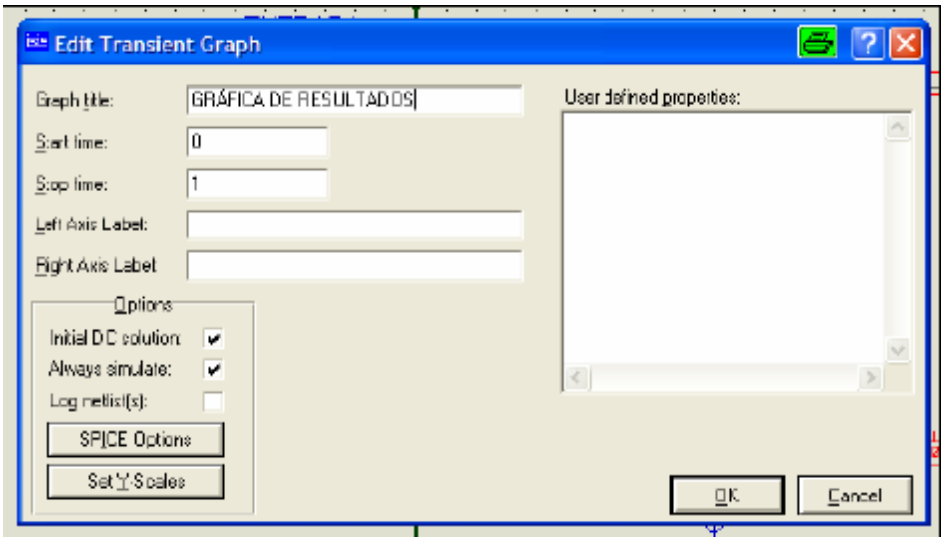



**Figura 3.79** *Ventana de edición de gráficos en Proteus*

Se debe rellenar el campo «Graph title» con el literal GRAFICA DE RESULTADOS y pulsar el botón OK. Coloque el ratón en cualquier parte de la superficie de trabajo que está libre y pulse el botón derecho para deseleccionar la gráfica. Para decir a Proteus que datos debe presentar en la gráfica, realizaremos la siguiente operación. Coloque el ratón sobre el generador «ENTRADA» y pulse el botón derecho para seleccionarlo (cambiará de color).

A continuación pulse el botón izquierdo y desplace el ratón hasta situarlo sobre la gráfica y allí suelte el botón. La gráfica tendrá el siguiente aspecto Figura 3.80:

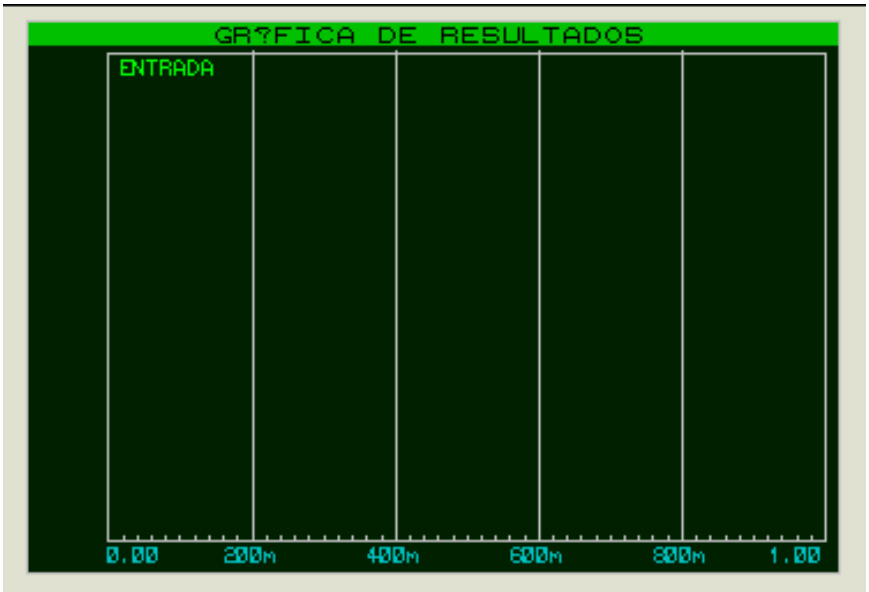



**Figura 3.80** *Grafica de resultados*

Realice la misma operación con la sonda SALIDA. El resultado final debe ser similar al que se representa en la siguiente figura:

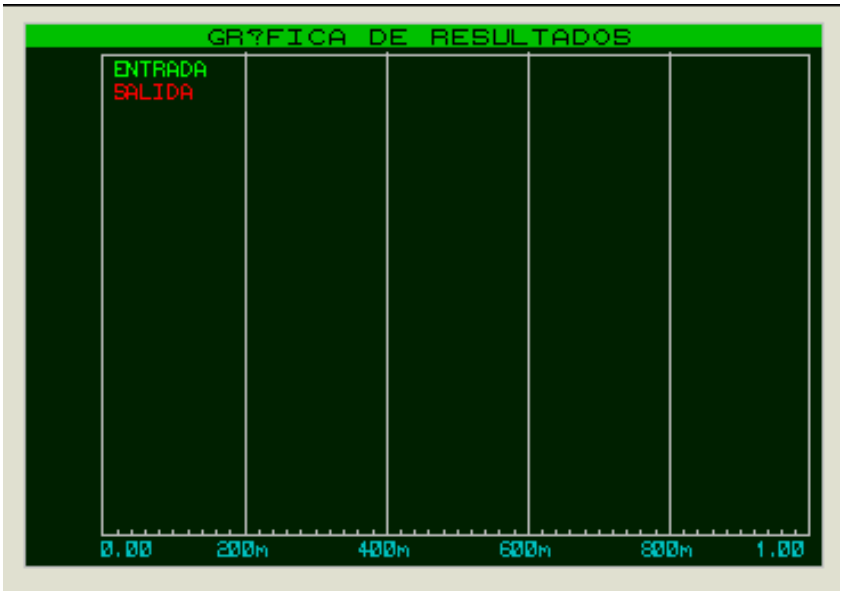
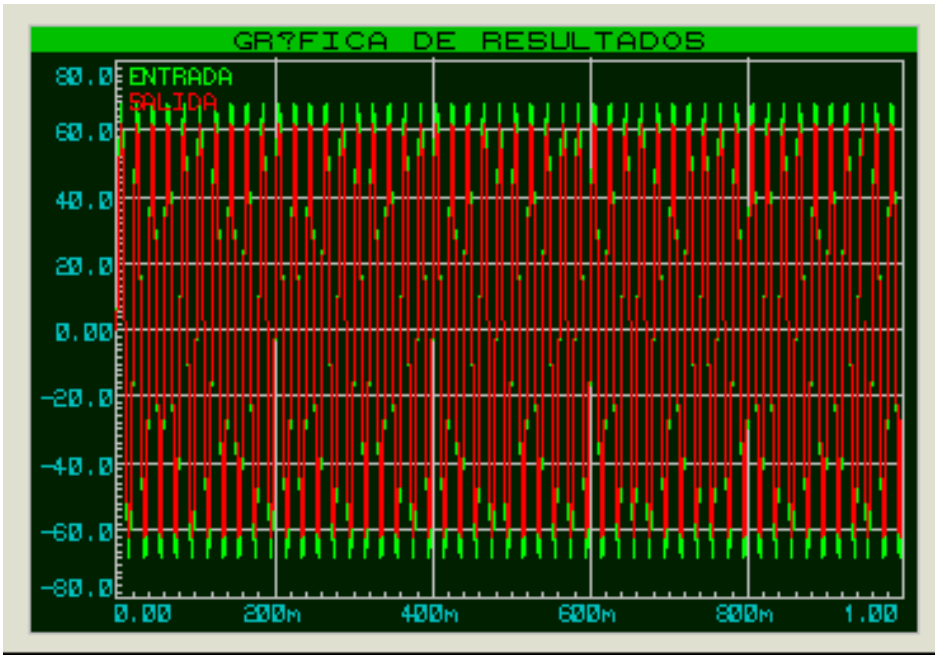



**Figura 3.81** *Gráfica de resultados con salida*

Para rellenar la gráfica con los datos resultantes deberá pulsar la tecla espaciadora.

El resultado que aparece, es el siguiente Figura 3.82:




:

**Figura 3.82** *Gráfico de la simulación en Proteus*

La gráfica visualizada es el resultado obtenido durante un intervalo de un segundo de los valores correspondientes a dos ondas senoidales, una de entrada de color verde y una de salida de color rojo. Al contener tantos datos el resultado se ve mal, pero no se preocupe, trataremos de mejorar el resultado.

Se debe seleccionar la gráfica, pulsando sobre ella con el botón derecho del ratón y a continuación abra la ventana de propiedades pulsando con el botón izquierdo.

En el campo «Stop time» se debe cambiar el valor de 1 que introducimos antes por el valor 20m. Con ello se esta indicando a Proteus que simule los datos que se produzcan en un intervalo de tiempo igual a 20 milisegundos, en lugar del de un segundo que nos aparecía por defecto.

Luego se debe cerrar la ventana pulsando el botón OK y pulse en cualquier lugar libre del área de trabajo para deseleccionar la gráfica. Hay que volver a pulsar la tecla espaciador y el resultado que obtendrá debe ser el siguiente:

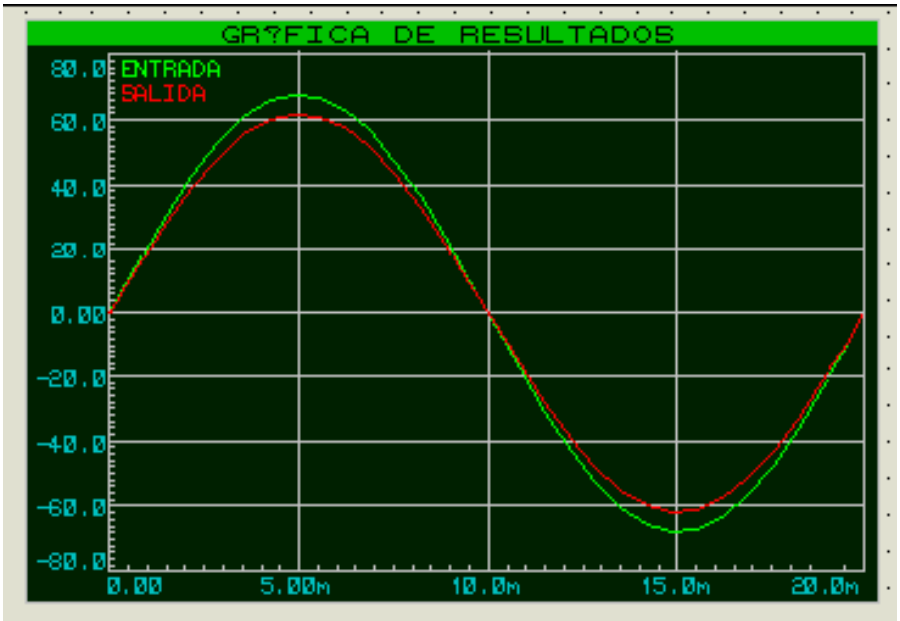



**Figura 3.83** *Gráfica de la simulación extendida*

### 3.13.5 Simulación utilizando el Pic 16F870

El siguiente circuito representa un control de dos semáforos mediante un Pic 16F870 simulado en Proteus ISIS:

Al momento de tener el archivo *.ASM emulado en MPLAB IDE automáticamente se puede ingresar al Pic 16F870 dando doble click sobre él con la herramienta de selección:

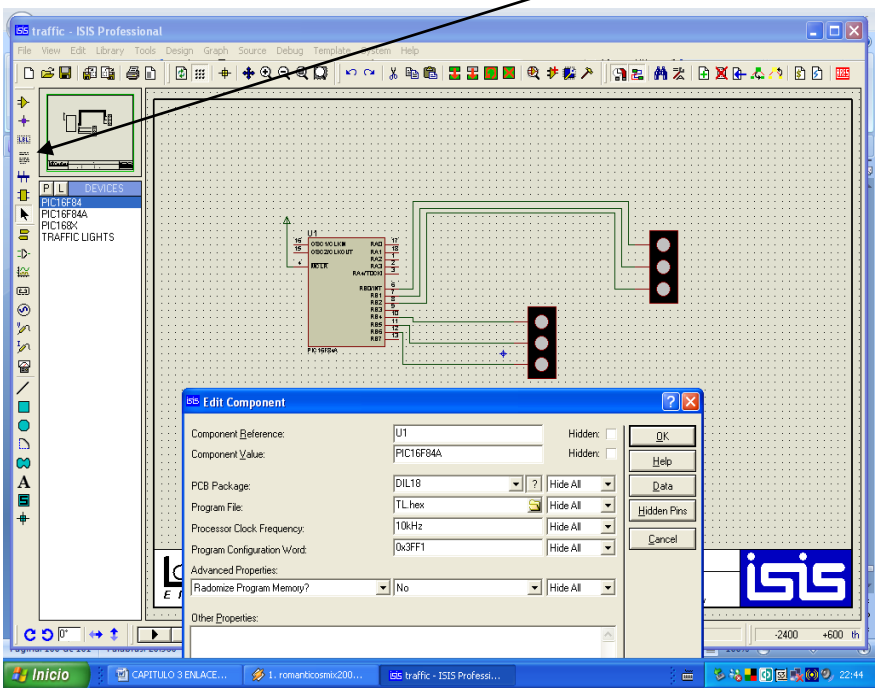



**Figura 3.84** *Circuito para un semáforo*

Luego inmediatamente se elige el archivo *.HEX en el campo Program file.

El programa se puede simular dando click sobre el botón empezar o Play

# ANEXO A
# GUIAS DE PRÁCTICAS

# ANEXO B

# CIRCUITOS PCB

# ANEXO C

# COSTOS

# ANEXO D

# DATA SHEETS

# CONCLUSIONES Y RECOMENDACIONES

## CONCLUSIONES

Los microcontroladores son dispositivos electrónicos que debidamente bien utilizados pueden aplicarse en todos los campos de las ciencias facilitando el entorno hombre-máquina.

La comunicación entre dispositivos electrónicos es fundamental ya que permite interactuar entre el receptor y emisor, mejorar la velocidad transmisión es un requisito que el diseñador debe tener en cuenta para cualquier proyecto.

La importancia del diseño y la experimentación es la base para realizar proyectos que tienen una gran aplicación en todas las áreas relacionadas con el control y la automatización de procesos.

Partir desde la simulación y llegar al uso de periféricos en circuitos con microcontroladores, facilita el camino a desarrollar complejos diseños más rápidos y baratos para su implementación en la Industria.

El estudio de programas que ayuden al diseño de circuitos esquemáticos y circuitos impresos (PCB), es esencial al momento de crear proyectos que implementan microcontroladores en el diseño de hardware.

Los microcontroladores PIC están en auge, utilizándose en proyectos industriales, de investigación y para docencia. No tiene sentido que el diseñador sólo pueda utilizar una única plataforma con PLCs u otro dispositivo de control para desarrollar aplicaciones, ya que el uso de los microcontroladores permite versatilidad, bajos costos y eficiencia en los proyectos industriales pequeños de automatización.

El diseño electrónico constituye una base fundamental para crear tecnología propia de bajo costo y ayuda a crear nuevos horizontes en el campo de los procesos industriales, debido a su versatilidad para el uso de diversos tipos de periféricos análogos y digitales.

La utilización de tecnologías de diseño y monitoreo por software como Labview e Intouch aplicadas al microcontrolador mediante el computador, permiten desarrollar interfaces completas de control que pueden ser creadas por estudiantes o profesionales a bajo costo y con la mejor confiabilidad.

## RECOMENDACIONES

En la programación se debe incluir las directivas correctas para el microcontrolador, de lo contrario el programa ensamblador presentará errores que deberán ser depurados.

En el software grabador ICProg se debe configurar adecuadamente las opciones como el tipo de oscilador usado, la protección de programa y los bits de configuración.

Si al momento de programar o compilar un PIC tenemos una ventana de error (Error de programación en la dirección 0000h!) es porque no se encuentra bien habilitado el Hardware a utilizar o porque el Microcontrolador está averiado y se deberá reemplazarlo.

Los niveles de voltaje y la polaridad de las fuentes de poder deben ser revisados antes de alimentar al módulo Entrenador de Microcontroladores (+5Vcc).

Se debe verificar exhaustivamente el diseño de la placa impresa en el circuito esquemático en Circuit Manager de DXP 2004 antes de fabricar la placa impresa para evitar errores y gastos económicos innecesarios.

Para soldar de los elementos electrónicos en la placa impresa se debe utilizar un cautín de máximo 20W de potencia y pasta de soldar, de lo contrario se pueden producir daños en las pistas o en los hoyos.

Para limpiar el exceso de pasta de soldar o impurezas en las placas impresas se recomienda usar espray limpiador de contactos eléctricos.

Para la colocación de circuitos integrados en la placa impresa se deben utilizar obligatoriamente zócalos para el fácil desmontaje de los mismos.

Se recomienda el uso del software Altium Designer 2006 en el cual se puede trabajar de igual forma que su antecesor DXP 2004 pero permite realizar en el diseño PCB tecnología mixta (plcc+dil+bga) complejas con mucha calidad, versatilidad y profesionalismo.

Las prácticas en un futuro próximo deberán ser orientadas hacia el uso de otros microcontroladores de las fábricas ATMEL o MOTOROLA, en los cuales se pueda trabajar con las mismas bondades del Pic de MICROCHIP.
La utilización de microcontroladores de 16 bits deberá ser indispensable ya que permitirá manejar un volumen de datos (Entradas, Salidas, etc.) más consistente para diversas aplicaciones complejas.

El estudio de la gama alta de Microchip (18FXXX y dsPic) y dispositivos microcontroladores para aplicaciones especiales así como el RFPIC, será fundamental para complementar el estudio de los microcontroladores y sus diversas aplicaciones en la robótica y los automatismos.

La aplicación de tecnologías basadas en microcontroladores se podría complementar de mejor manera al integrar la interfaz con kits como los de la National Instruments (Adquisición de datos, Control e instrumentación GPIB, Visión de máquina y Control de movimiento) en los cuales se pueden simular ejercicios completos de automatización, control y monitoreo manual y remoto con la PC.

# BIBLIOGRAFÍA


[1] ANGULO, J.M., Martinez I.,Microcontroladores PIC Diseño práctico de aplicaciones, Editorial McGraw Hill/Interamericana de España, 2da. Edición, 1999.

[2] Inga Ortega, E. M. (2002). Análisis costo beneficio de la automatización en el sistema de producción de hormigón para Hormiazuay Cía. Ltda. . Cuenca: Universidad Politécnica Salesiana.

[3] BOLTON W., Mecatrónica, Sistemas de Control Electrónico en Ingeniería Mecánica y Eléctrica, 2da. Edición.

[4] CORRALES V. Santiago, Electrónica Práctica con Microcontroladores Pic Electrónica & Computación, Ecuador, 2006.

[5] PALLAS ARENY Ramón, Sensores y Acondicionadores de Señal, 3ra. Edición

[6] STENERSON Jon, Fundamentals of Programmable Logic Controllers Sensors and Comunications, 1ra. Edición

**URL:**

[7] http://www.cerne-tec.com.br

[8] http://www.educa.madrid.org

[9] http://www.elrebujito.com.es

[10] http://www.ieeproteus.com

[11] http://www.microchip.com

[12] http://www.micropic.es

[13] http://www.pic16f84a.com

[14] http://www.picmania.garcia-cuervo.com

[15] http://www.robots-argentina..com.ar

[16] http://www.roboticajoven.mendoza.edu.ar

[17] http://www.superrobotica.com

[18] http://www.todopic.com.ar

[19] http://www.x-robotics.com