ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNOLOGOS

IMPLEMENTACION DE UN PROTOTIPO DE CONTROL COMPUTARIZADO DE TRES CARGAS DE 1000[W] CADA UNO, MEDIANTE EL MICROCONTROLADOR PIC16F870, CON INTERFAZ SERIAL RS – 485.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRONICA Y TELECOMUNICACIONES

DIEGO RENE GUALOTO GUACOLLANTE
EDWIN RAUL PERALTA PILLAJO

DIRECTOR: PABLO LOPEZ M.

QUITO, NOVIEMBRE 2007

DECLARACIÓN

Nosotros, **Diego René Gualoto Guacollante, Edwin Raúl Peralta Pillajo**, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad, intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Diego René Gualoto G.	Edwin Raúl Peralta P.

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por **Diego René Gualoto Guacollante, Edwin Raúl Peralta Pillajo**, bajo mi supervisión.

Ing. Pablo López M.
DIRECTOR DE PROYECTO

PRESENTACIÓN

Este proyecto tiene por objeto presentar la implementación de un prototipo de control computarizado de tres cargas de 1000[W] cada uno, mediante el microcontrolador PIC16F870, con interfaz serial RS – 485, el cual esta desarrollado en tres capítulos con los siguientes contenidos:

El primer capitulo se realiza una descripción de todos los componentes de este proyecto, encontramos así la información sobre las características del PIC como parte central del circuito de control, programación mediante el Microcode, amplificadores, fuentes de poder, comunicación serial de PC a PIC, comunicación serial PIC a PIC con la interfaz RS-485, comunicación serial de Visual Basic y PC.

El segundo capitulo esta encaminado a proporcionar la información teórica y las bases técnicas necesarias para la construcción y ensamblaje del módulo para el control de la red eléctrica a implementar con microcontrolador PIC16F870 que incluye el protocolo RS – 485, su funcionamiento, su programación, el desarrollo del software en Visual Basic 6.0 para control de funcionamiento de la red eléctrica por medio del protocolo RS – 485, así como también el respectivo análisis y pruebas técnicas.

El tercer capitulo contiene las conclusiones y recomendaciones, que deba ser consideradas en el presente proyecto y a futuro.

RESUMEN

El presente documento expone la implementación de un prototipo de control computarizado de tres cargas de 1000[W] cada uno, mediante el microcontrolador PIC16F870, con interfaz serial RS – 485, el cual puede ser utilizado en diversos lugares donde sea necesario llevar un control de las cargas en determinadas áreas.

Tener el control computarizado es una ventaja muy grande para una empresa. Por tal motivo en la actualidad es necesario poder controlar todos los periféricos desde un mismo sitio. En un edificio podemos controlar los sistemas iluminación, sistemas de ventilación, circuito cerrado de televisión, sistemas de cerraduras eléctricas, sistemas de seguridad, etc. (áreas comunales) todo desde la pantalla de un computador.

La implementación es desarrollada aprovechando las características que ofrece el microcontrolador PIC16F870, su flexibilidad de programación y accesibilidad de periféricos de entrada y salida. Y además la utilización de la programación en Visual Basic 6.0 para el control mediante un computador.

Se realiza el análisis técnico y económico, determinando todos los beneficios que nos brinda esta implementación.

Con el presente trabajo se trata de impulsar el desarrollo de proyectos que sean aplicables a nuestra realidad, explotando los conocimientos técnicos adquiridos en la carrera y aprovechando el avance de la tecnología.

AGRADECIMIENTO

Agradezco a Dios Todopoderoso por darme vida, salud y fuerza para afrontar las distintas circunstancias que se presentan en nuestro camino.

Agradezco principalmente a mis padres y hermanos por brindarme su apoyo y confianza incondicional para la consecución de todas mis metas.

Y mi agradecimiento a mis ingenieros, mi eterna gratitud porque con su ejemplo ha sembrado en mi, la semilla del saber y a la Escuela Politécnica Nacional de cuyas aulas llevo los mejores recuerdos.

DIEGO GUALOTO

DEDICATORIA

Este trabajo se lo dedicó a mis padres Sr. Hernán Gualoto y Sra. Martha Guacollante, ya que con su amor y compresión han sabido guiar mi vida por el sendero de la verdad y la justicia a fin de engrandecer a mi Patria y honrar a mi familia. Doy gracias el haberme brindado el fruto de su esfuerzo y sacrificio por ofrecerme un mañana mejor.

A mis hermanos Vinicio y Carolina por el cariño y apoyo que me ofrecieron en todo momento.

DIEGO GUALOTO

AGRADECIMIENTO

Agradezco profundamente a nuestro guía, Dios Todopoderoso, por darme el entendimiento, la sabiduría, el consuelo, la tenacidad de seguir adelante con mis aspiraciones.

Agradezco a mis Padres los cuales me dieron todo su apoyo y confianza para seguir adelante en mi carrera de estudiante.

Agradezco a mis hermanos los cuales ven con gran alegría que he alcanzado un peldaño mas en mi vida, por darme todo su apoyo moral de seguir adelante.

Y al final pero no menos importante agradezco a mis amigos y compañeros de clases, los cuales los llevare en el corazón por su amistad sincera, por haber conocido personas tan maravillosas en estas salas de clase.

RAÚL PERALTA

DEDICATORIA

Dedico este trabajo a Dios el cual me guía,

A Mis Padres: Sr. PABLO PERALTA U.

Sra. DELFINA PILLAJO P.

Por su amor y comprensión.

A Mis Hermanos: GLORIA, RENE, PATRICIA

Por su gran apoyo y esfuerzo.

RAÚL PERALTA

CONTENIDO

TEMA **Pág.**

CAPITULO 1

MARCO TEORICO

1.1 LOS MICROCONTROLADORES[1]

1.1.1 INTRODUCCIÓN A LOS MICROCONTROLADORES

Los microcontroladores hicieron su aparición a principio de los ’80 y se trata de un circuito integrado programable que contiene toda la estructura de una microcomputadora. Es decir que, dentro de un microcontrolador podemos encontrar:

- Una CPU (Unidad Central de Proceso)
- Memoria RAM.
- Memoria ROM
- Memoria EEPROM
- Puertos de Entrada/Salida (Pines de E/S) e incluso muchos modelos de microcontroladores incorporan distintos módulos “periféricos”, como pueden ser; conversores analógico/digital, módulos PWM (control por ancho de pulso), módulos de comunicaciones seriales o en paralelo, ver figura 1.1.

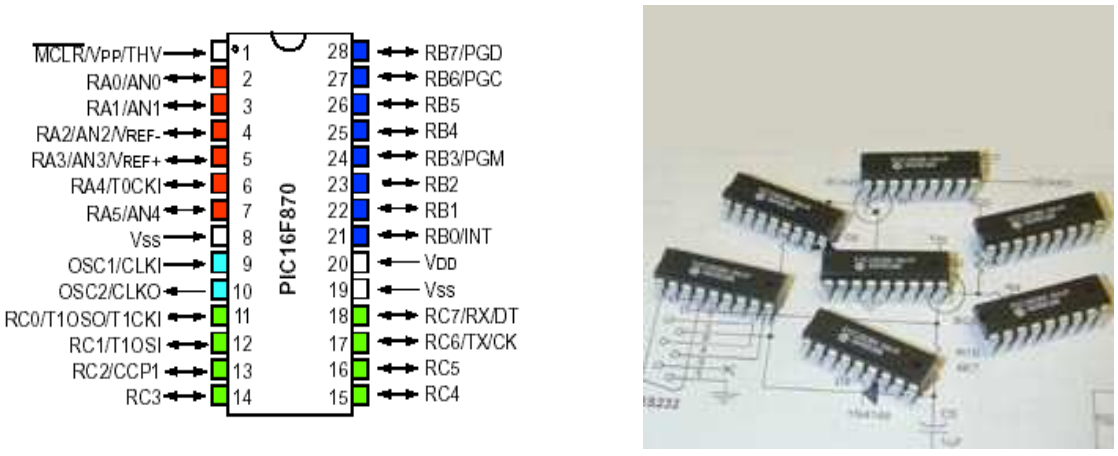


Figura 1.1 Circuito integrado del PIC 16F870

[1] José M Angulo Usategui, Susana Romero Yesa e Ignacio Angulo Martinez “Microcontroladores PIC – Diseño práctico de aplicaciones segunda parte- PIC16F87x”, *ed. Graw Hill* 1ª Edición

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

1.1.2 LOS MICROCONTROLADORES “PIC”

Los microcontroladores denominados “PIC”, corresponden exclusivamente a la marca “Microchip”. PIC significa "**Peripheral Interface Controller**" y fue desarrollado por Microchip a principio de los 80.

Existe una gran cantidad de modelos de microcontroladores cuyas características y prestaciones varían de un modelo a otro. De esta manera los fabricantes pueden seleccionar el modelo que mejor se ajuste a sus necesidades. Los distintos modelos de microcontroladores se agrupan por “familia”. Una familia puede estar formada por un conjunto de modelos cuyas características y prestaciones son bastante similares.

Cuando compramos un microcontrolador, la memoria del mismo se encuentra “vacía” y para que funcione es necesario que sea “programado”.

1.1.3 ESTRUCTURA DE UN MICROCONTROLADOR[2]

Básicamente, un microcontrolador esta compuesto por los siguientes componentes:

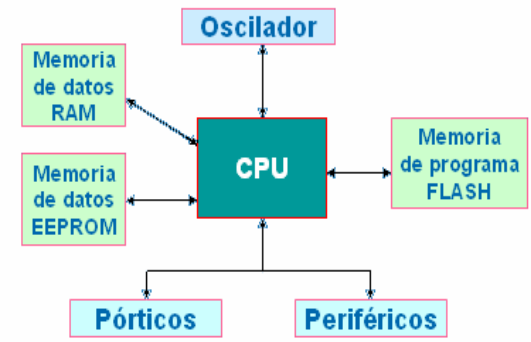


Figura 1.2 Partes de un microcontrolador

[2] Microchip, *http:// www.microchip.com*

- Procesador o CPU (Unidad Central de proceso).
- Memoria para el programa tipo ROM.
- Memoria RAM para contener los datos.
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, Puertas Serie y Paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.).

En la figura 1.3 podemos ver la estructura de un microcontrolador PIC, en este caso de la familia 16F87X.

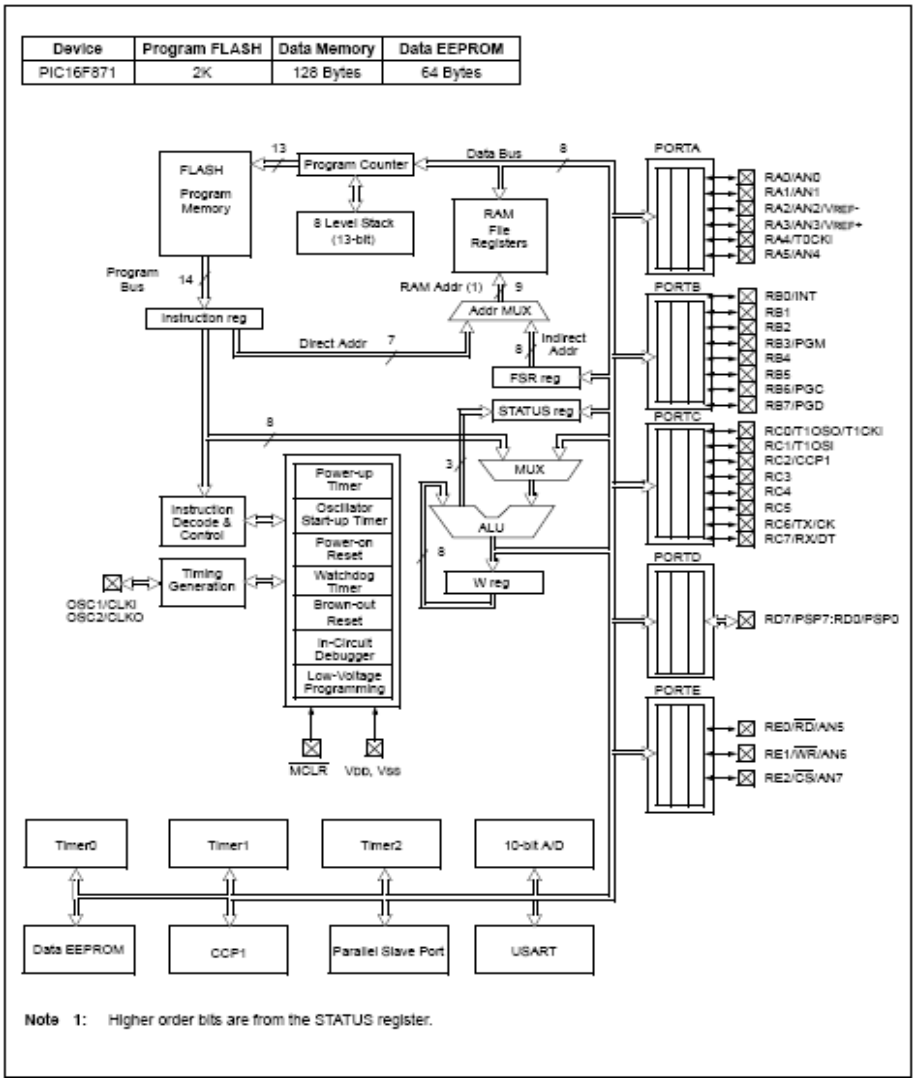


Figura 1.3 Estructura interna del PIC

1.1.3.1 El Procesador o CPU

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. La CPU (Unidad Central de Proceso) se encarga la decodificación y ejecución del programa.

Actualmente, existen 3 tipos de arquitectura de procesadores:

- **CISC** (Juego de Instrucciones Complejo para Computadores): Disponen de más de 80 instrucciones 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 instrucciones complejas que actúan como macros.

- **RISC** (Juego de Instrucciones Reducido para Computadores): En estos procesadores el repertorio de instrucciones es muy reducido y las instrucciones son simples y generalmente se ejecutan en un ciclo maquina. La ventaja de éstos es que la sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

- **SISC** (Juego de Instrucciones Específico para Computadores): 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.

1.1.3.2 Memoria ROM

La memoria ROM es una memoria no volátil, es decir, que no se pierden los datos al desconectar el equipo y se destina a contener el programa de instrucciones que gobierna la aplicación. Los microcontroladores disponen de capacidades de ROM comprendidas entre 512 bytes y 8 k bytes.

Existen distintos tipos de memorias ROM, la cual determinará la aplicación del microcontrolador.

- **ROM con máscara:** Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip.

- **OTP:** El microcontrolador contiene una memoria no volátil de sólo lectura "programable una sola vez" por el usuario. OTP (One Time Programmable).

- **EPROM:** Los microcontroladores que disponen de memoria EPROM (Erasable Programmable Read Only Memory) pueden borrarse y grabarse muchas veces. Para borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico.

- **EEPROM:** Es una memoria de sólo lectura, las cuales se puede escribir y borrar eléctricamente. EEPROM (Electrical Erasable Programmable Read Only Memory). Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. El número de veces que puede grabarse y borrarse una memoria EEPROM es finito.

- **FLASH:** Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos energía y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM. Es más veloz y tolera más ciclos de escritura y borrado.

1.1.3.3 Memoria RAM

La memoria RAM es una memoria volátil, es decir, que se pierden los datos al desconectar la energía eléctrica y se destina a guardar las variables y los datos. Los microcontroladores disponen de capacidades de RAM comprendidas entre 20 y 512 bytes.

1.1.3.4 Registros y Bits

Un registro es una posición de memoria en la cual se puede almacenar un dato. Es decir que la memoria esta dividida en pequeñas partes llamadas "**Registros**". Dentro de la memoria, cada registro se identifica mediante un número, el cual se denomina "**Dirección de memoria**" y generalmente está expresado en formato Hexadecimal. El primer registro de una memoria corresponde a la dirección 00h.

Dado a que cada registro es identificado mediante un número hexadecimal puede resultar muy complejo a la hora de diseñar el programa, existe la posibilidad de asignar un "nombre" a una dirección de registro. En general, este nombre está directamente relacionado con la función que cada registro cumple dentro del sistema.

Entonces podemos decir que un Registro esta formado por un conjunto de 8 bits.

Los sistemas digitales representan la información en forma de bits porque sus circuitos sólo pueden tener dos estados: encendido o apagado. En general podemos decir que:

- 1 = Encendido = Verdadero = "SI" = +5V
- 0 = Apagado = Falso = "NO" = 0V

Cada Bit se identifica por la posición que ocupa dentro del registro, siendo el primer Bit el número 0, que es el que se encuentra en el extremo derecho del registro, ver figura 1.4.

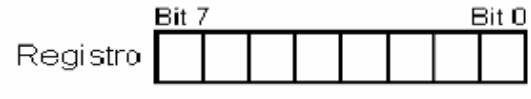


Figura 1.4 Estructura de un registro

En un registro se puede almacenar una combinación 8 ceros y/o unos. Esto nos da una cantidad de 2^8 combinaciones, es decir, 256 posibles combinaciones de ceros y unos. Esto significa que un registro puede procesar valores entre 0 y 255.

El siguiente ejemplo muestra el desarrollo de un cálculo de conversión de base de sistema binario (base 2) a sistema decimal (base 10):

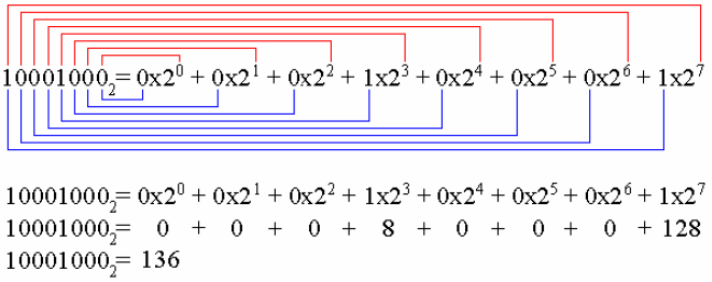


Figura 1.5 Cambio de Sistema binario a sistema decimal

Se llama "**Peso Binario**" al valor que representa un Bit según la posición que ocupa dentro del registro. El Bit que está ubicado más hacia la derecha del registro, es el Bit menos significativo (LSB) y tiene un peso de $2^0 = 1$. El Bit del extremo izquierdo del registro es el Bit más significativo (MSB) y tiene un peso de $2^7 = 128$.

Los pesos binarios crecen de derecha a izquierda en potencias de 2.

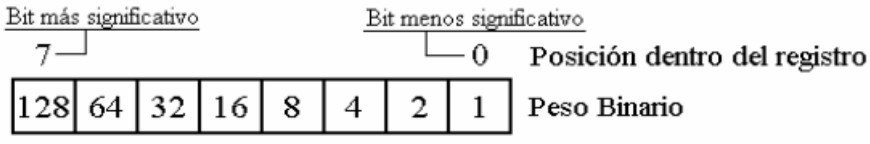


Figura 1.6 Sistema de pesos en el sistema binario.

La manera de simplificar la conversión de binario a decimal, es directamente sumar los valores de los pesos binarios de los bits cuyo valor sea 1.

Figura 1.7 Simplificación del cálculo de conversión.

El sistema hexadecimal es un sistema en base 16 y consta de 16 dígitos diferentes que son: del 0 al 9 y luego de la letra "**A**" a la "**F**", es decir, 10 dígitos numéricos y seis caracteres alfabéticos. El sistema hexadecimal se usa como forma simplificada de representación de números binarios y debidos a que 16 es una potencia de 2 ($2^4 = 16$), resulta muy sencilla la conversión de los números del sistema binario al hexadecimal y viceversa.

Mediante el sistema hexadecimal podemos representar un valor numérico de 8 bits utilizando sólo 2 dígitos. De ésta manera estamos dividiendo el registro de 8 bits en dos partes de 4 bits cada una llamada **Nibble**.

Al nibble correspondiente a los 4 bits menos significativos, se lo denomina "**Nibble Bajo**" y al nibble correspondiente a los 4 bits más significativos se lo denomina "**Nibble Alto**".

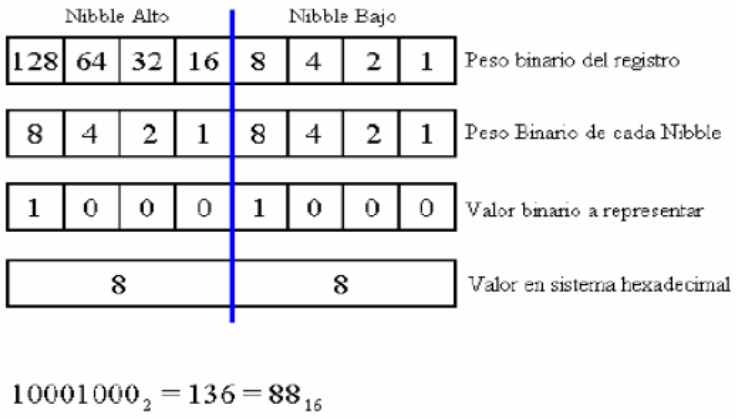


Figura 1.8 Separación entre Nibbles

El sistema hexadecimal es utilizado para identificar las direcciones de registros de las memorias, en sistemas digitales porque permite representar el valor de un **Nibble** con sólo un digito, ya que:

$$1111_2 \rightarrow (8+4+2+1) = 15 = F_{16}$$

Figura 1.9 Sistema Hexadecimal

Esto permite representar números grandes utilizando unos pocos dígitos.
Por ejemplo:

$$FF\ FF_{16} = 11111111\ 11111111_2 = 65535$$

Figura 1.10 Representación de 4 dígitos hexadecimales

En la programación de microcontroladores, es habitual utilizar los tres sistemas de numeración (Binario, Decimal y Hexadecimal), dependiendo del proceso que necesitemos realizar. Por eso es fundamental tener claros estos conceptos.

1.1.3.5 Líneas de Entrada/Salida (E/S), (Puertos)

Los microcontroladores cuentan con una serie de pines destinados a entrada y salida de datos o señales digitales. A estos pines se les denomina "**Puerto**".

Un puerto si puede estar formado por menos de 8 pines. Un microcontrolador puede contener varios puertos dependiendo del modelo. A cada puerto se lo identifica con una letra. Por ejemplo; "Puerto A", "Puerto B", etc. Para poder utilizar un puerto, el mismo debe ser configurado. Cada pin de un puerto puede ser configurado como entrada o salida independientemente del resto de los pines del mismo puerto.

1.1.3.6 Módulos Temporizadores Internos (TMRS)

Un temporizador interno (TMR), es un módulo de hardware incluido en el mismo microcontrolador, el cual está especialmente diseñado para incrementar

automáticamente el valor de un registro asociado al TMR cada vez que el módulo TMR recibe un pulso. A este pulso se lo llama "**señal de reloj**".

El módulo TMR siempre incrementa el valor del registro asociado, nunca decrementa dicho valor. Algunos microcontroladores pueden incluir más de un módulo TMR y la señal de reloj de cada uno de éstos puede ser de origen interno o externo. Si el origen de la señal de reloj está configurado como externo, el módulo temporizador puede ser utilizado como un contador de eventos externos, incrementando el TMR con cada pulso recibido mediante el pin correspondiente.

Si el origen de la señal de reloj es interno, el TMR incrementa con cada ciclo del oscilador. Esto permite utilizar el temporizador como "contador de ciclos de programa", donde, un ciclo corresponde al tiempo de ejecución de una instrucción, lo cual se puede calcular con la siguiente fórmula:

$$\frac{1}{Frec.Osc./4}$$

Donde "Frec. Osc." es la frecuencia del oscilador utilizado.

Dado que la velocidad de ejecución del microcontrolador corresponde a ¼ de la velocidad del cristal utilizado, cada ciclo de programa se ejecuta en un tiempo determinado según el cristal que estemos utilizando. Por ejemplo; con un cristal de 4Mhz la velocidad real de procesamiento del microcontrolador es de 1 MHz. Aplicando la siguiente fórmula:

$$T = \frac{1}{4000000/4} = \frac{1}{1000000} = 0.01ms$$
$$\Rightarrow T = 1\mu s(microsegundos).$$

Esto significa que cada ciclo de programa se ejecuta a **[1/1000000] = 1*u*s** y dado que cada incremento del TMR corresponde a un ciclo de programa, si contamos los incrementos de un TMR, indirectamente podremos calcular el tiempo transcurrido.

1.1.4 UTILIZANDO UN MICROCONTROLADOR

- **Lenguaje de "Alto Nivel":** permite que los algoritmos se expresen en un nivel y estilo de escritura fácilmente legible y comprensible por el hombre. En la actualidad se trata de lenguajes de tipo visual.
- **Lenguaje de "Bajo Nivel":** el usuario se acerca un poco más al lenguaje de maquina. Permiten un acceso más amplio al control físico de la maquina (hardware).

- **Lenguaje Ensamblador:** Podríamos considerarlo el lenguaje de más bajo nivel. El usuario escribe código en el mismo "idioma" del procesador. Se tiene control total del sistema. El lenguaje de programación es muy especifico para cada modelo de procesador, incluso puede variar de un modelo a otro de procesador dentro de un mismo fabricante.

Podemos decir que los lenguajes de alto Nivel se asemejan más al lenguaje humano y que los lenguajes de bajo Nivel se asemejan más al lenguaje de máquina y en el lenguaje ensamblador el usuario debe programar en el propio "idioma del procesador".

Figura 1.11 Niveles de lenguajes de programación

El microcontrolador sólo entiende de números, es decir que, el código Assembler (texto) no puede ser procesado directamente por el microcontrolador. Para poder grabar el programa en el microcontrolador, primero debemos convertir el texto del código Assembler a números, en general, en formato hexadecimal. A este proceso se le llama "**Compilación**".

Una vez desarrollado el código Assembler, el mismo debe ser "compilado". Esto se realiza con un "software compilador".

El archivo compilado tiene una extensión .hex. Por último, después de compilado, el programa está listo para ser grabado al microcontrolador. Esto realiza mediante una placa programadora. A ésta placa programadora, comúnmente se la llama "**programador**".

Existen distintos tipos de programadores, los cuales pueden ser conectados a la PC mediante el puerto Serie (COM) o Paralelo (LPT).

A continuación se presenta un resumen del proceso de desarrollo del código y grabación de un microcontrolador:

- Escribir el código Assembler: Se genera un archivo con extensión ASM.
- Compilar el código Assembler: Se genera un archivo con extensión HEX.
- Grabar (transferir) el programa desde la PC al microcontrolador mediante un programador.

1.1.5 CARACTERÍSTICAS TÉCNICAS DE LOS PIC 16F87X

Inicialmente todos los microcontroladores incorporaron la arquitectura de Von Neumann, la cual se caracteriza por disponer de una sola memoria principal donde se almacenan los datos y las instrucciones. A esta memoria se accede a través de un sistema de buses único (direcciones, datos y control).

Los microcontroladores de la familia 16F87X, al igual que el resto de los microcontroladores de la actualidad, están diseñados con la arquitectura **Harvard**. La arquitectura Harvard dispone de dos memorias independientes; una que contiene sólo instrucciones, y otra donde se almacenan los datos, ver figura 1.12.

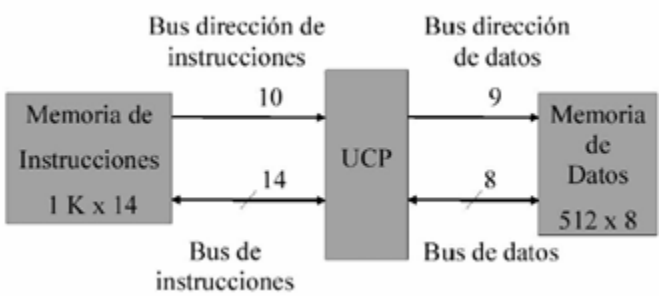


Figura 1.12 Arquitectura Harvard

Ambas memorias cuentan con sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias. Para que la CPU funcione, debe contar con un generador de impulsos de reloj externo que sincroniza el funcionamiento. Como origen de la señal de reloj externa utilizamos un cristal de cuarzo. En general, un ciclo de programa corresponde a una línea de código Assembler.

1.1.5.1 Los Puertos del PIC 16F87X

Los microcontroladores de la familia 16F87x disponen de 3 a 5 puertos según el modelo de microcontrolador: Estructura interna y especificaciones técnicas en el (ANEXO 1).

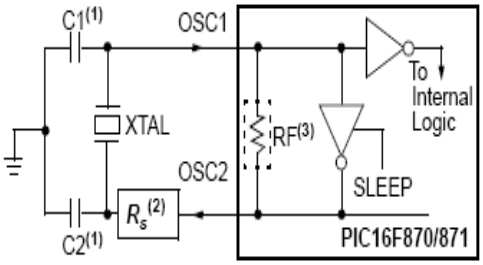
- Puerto A = 6 pines (5 pines A/D)
- Puerto B = 8 pines
- Puerto C = 8 pines
- Puerto D = 8 pines
- Puerto E = 3 pines (3 pines A/D)

1.1.5.2 Oscilador externo 16F87X

Para la programación del microcontrolador PIC 16F870 se debe tener en cuenta toda su estructura externa y elementos que serán conectados a él, una buena operación en el tiempo de maquina se debe colocar los siguientes esquemas:

- Polarización: V_{DD} = 5V; V_{SS} = 0V
- OSC1/CLKIN: entrada del circuito oscilador externo
- OSC2/CLKOUT: Auxiliar del circuito oscilador

Configuración de cristal resonante (HS, XT o LP).



Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

These values are for design guidance only.
See notes following this table.

Figura 1.13 Configuración de cristal resonante (HS, XT o LP)

Oscilador de modo RC

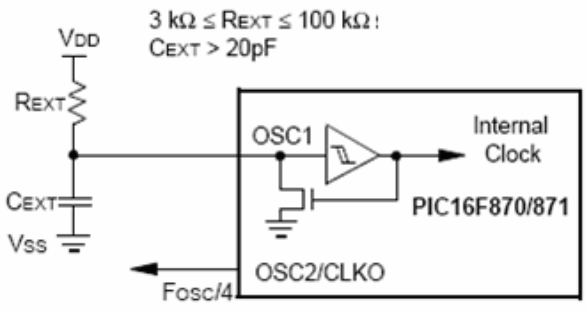


Figura 1.14 Oscilador de modo RC

1.2 VISUAL BASIC[3]

1.2.1 CARACTERÍSTICAS GENERALES

Visual Basic (VB).- Es un lenguaje de programación desarrollado por Alan Cooper para Microsoft. El lenguaje de programación es un dialecto de BASIC. Su primera versión fue presentada en 1991 con la intención de amplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y en cierta medida también la programación misma.

Es un lenguaje de fácil aprendizaje pensado tanto para programadores principiantes como expertos y centrado en un motor de formularios que facilita el rápido desarrollo de aplicaciones gráficas.

Es conocida la inhabilidad de VB, para que de manera directa pueda tomar el control directo de los periféricos de una PC, pero VB es muy versátil para la creación de programas en ambientes Windows, así como la incorporación de nuevos elementos multimedia y gráficos. Por esta razón se utiliza este lenguaje como plataforma de desarrollo del control electrónico, que sí puede tomar control de los periféricos de una PC y aprovechar la facilidad de VB para crear interfaces de usuario fácil y rápidamente.

Dicho mecanismo toma la forma de una DLL, misma que se programa en otro lenguaje diferente de VB. Una vez que se cuenta con dicha librería se utiliza como cualquier otra disponible en Windows, que a través de teclas definidas en el tablero de una PC (que cubra los requisitos estipulados para la instalación y operación), de manera gráfica en el monitor se observan las diferentes opciones del programa haciendo el control fácil y versátil.

El compilador de Microsoft genera ejecutables que requieren una DLL para que funcionen, en algunos casos llamada MSVBVMxy.DLL (acrónimo de "Microsoft Visual Basic Virtual Machine x.y", siendo x.y la versión), que provee todas las funciones implementadas en el lenguaje. Además existen un gran número de

[3] http://es.wikipedia.org/wiki/Lenguaje_de_programación%C3%83n

bibliotecas (DLL) que facilitan el acceso a muchas funciones del sistema operativo y la integración con otras aplicaciones.

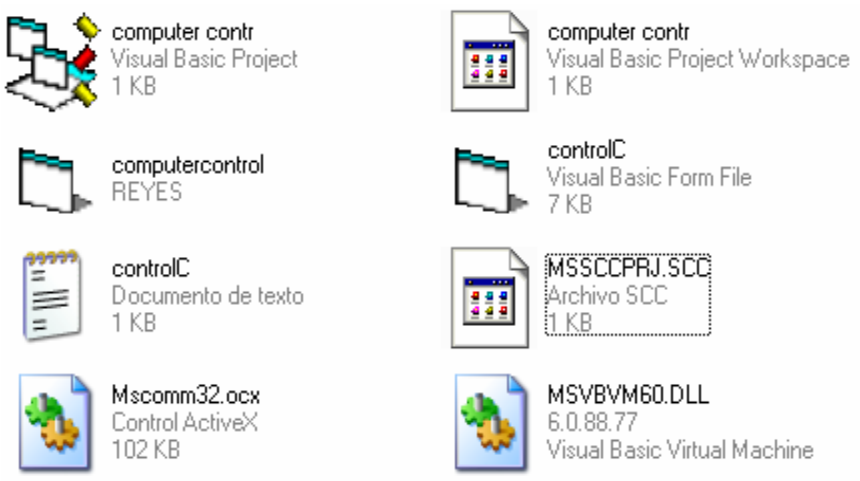


Figura 1.15 Iconos generados por el programa Visual Basic

Sin embargo esto sólo es una limitación en sistemas obsoletos, ya que las bibliotecas necesarias para ejecutar programas en Visual Basic vienen de serie en todas las versiones de Windows, desde Windows 2000. La versión 6.0 continúa utilizándose masivamente porque soporta características propias de los lenguajes orientados a objetos.

1.2.2 CARACTERÍSTICAS DE VISUAL BASIC.

Diseñador de entorno de datos: Es posible generar, de manera automática, conectividad entre controles y datos mediante la acción de arrastrar y colocar sobre formularios o informes.

Los Objetos Activos son una nueva tecnología de acceso a datos mediante la acción de arrastrar y colocar sobre formularios o informes.

Asistente para formularios: Sirve para generar de manera automática formularios que administran registros de tablas o consultas pertenecientes a una base de datos, hoja de cálculo u objeto.

Asistente para barras de herramientas: Es factible incluir barra de herramientas personalizada, donde el usuario selecciona los botones que desea visualizar durante la ejecución.

1.2.3 DESCRIPCION DE BARRAS, VENTANAS Y HERRAMIENTAS EN VISUAL BASIC.[4]

Barra de titulo: muestra el nombre del proyecto y del formulario que se está diseñando actualmente.

Figura 1.16 Barra de titulo del Programa VB.

Barra de menús: agrupa los menús despegables que contienes todas las operaciones que pueden llevarse a cabo con Visual Basic 6.0.

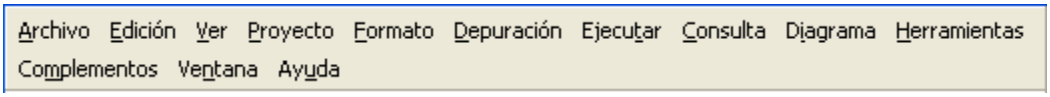


Figura 1.17 Barra de menú del programa VB.

Barra de herramientas estándar: contienen los botones que se utilizan con mayor frecuencia cuando se trabaja con un proyecto. Simplifica la elección de opciones de los menús Archivo, Edición, Ver y Ejecutar; además, en el área derecha presenta la ubicación (coordenadas) y el tamaño del objeto seleccionado.

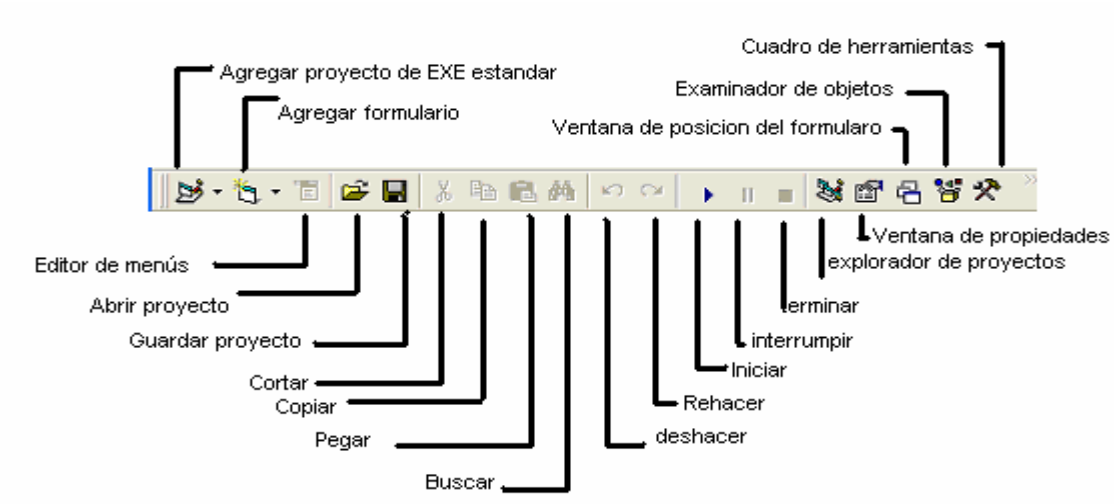


Figura 1.18 Barra de herramientas del programa VB.

[4] *La ruta practica a VISUAL BASIC*, Editorial Macro ERIL, 1ra. Edición; Perú 2003, Pág. 10-24.

Ventana de formulario: es el área donde se diseña la interfaz gráfica, es decir, es donde se inserta los gráficos seleccionados, como botones, imágenes, casilla de verificación, cuadros de listas, etc.

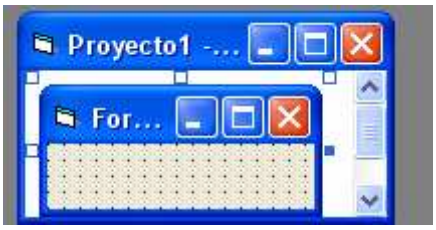


Figura 1.19 Ventana de formulario del programa VB.

Cuadro de herramientas: presenta todos los controles necesarios para diseñar una aplicación, como cuadros de texto, etiquetas, cuadros de listas, botones de comandos, etc.

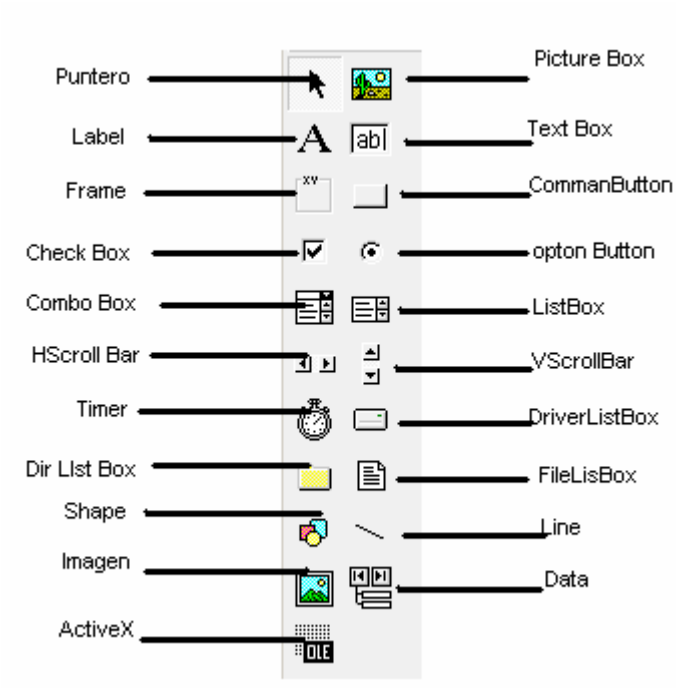


Figura 1.20 Cuadro de herramientas del programa VB.

Ventana de proyecto: muestra los elementos involucrados en el proyecto, como formularios, módulos, controles, etc. Cada elemento puede seleccionarse en forma independiente para su edición.

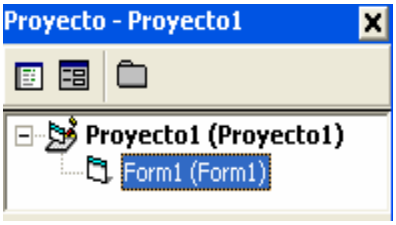


Figura 1.21Ventana de proyecto del programa VB.

Ventana de posición del formulario: muestra la ubicación que tendrá el formulario en la pantalla, cuando ejecute la aplicación. Esta ubicación puede cambiarse si se hace clic con el botón izquierdo del mouse.

Figura 1.22 Ventana de posicionamiento de formulario del programa VB.

La Ventana propiedades: muestra todas las propiedades del control actualmente seleccionado, en este caso muestra las propiedades del Form1, luego podemos ver que abajo dice "Form1 Form", lo que está en negrita es el nombre del objeto, y lo que le sigue es el tipo de objeto, en este caso es un Formulario (Form).

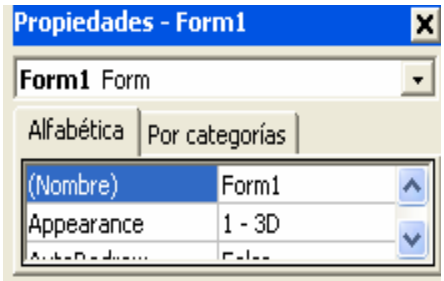


Figura 1.23 Ventana de propiedades del programa VB.

1.2.3.1 Descripción de la barra de herramientas

Las herramientas en VB son de fácil comprensión por su carácter grafico:

TextBox: Mediante este control podremos realizar tanto la entrada como la salida de datos en nuestras aplicaciones. No hace falta que indiquemos las coordenadas de la situación del formulario en pantalla, simplemente tendremos que marcar sobre el control de la caja de herramientas y dibujarlo con el tamaño que queramos en nuestro formulario.

Figura 1.24 Icono de acceso a TextBox

Label: Este control es también uno de los más utilizados, aunque su utilidad queda restringida a la visualización de datos en el mismo, no permitiendo la introducción de datos por parte del usuario.

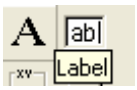


Figura 1.25 Icono de acceso a Label

CommandButton: Este control es el típico botón que aparece en todas las aplicaciones y que al hacer clic sobre él nos permite realizar alguna operación concreta, normalmente Aceptar o Cancelar. Aunque según el código que le asociemos podremos realizar las operaciones que queramos

Figura 1.26 Icono de acceso a CommandButton.

OptionButton: Este control nos permite elegir una opción entre varias de las que se nos plantean. Cada opción será un control OptionButton diferente.

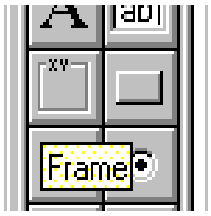
Figura 1.27Icono de acceso a OptionButton.

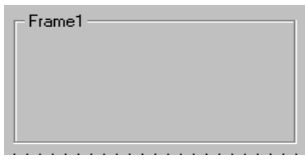
1.2.3.2 Bloquear los Controles

Cuando estén situados los controles en el formulario se pueden bloquear para que no puedan moverse de forma accidental. Para esto deberemos pulsar en la barra de herramientas.

Cuando actives este botón y mientras no desbloquees los controles utilizando la misma opción no se podrán mover ninguno de los controles del formulario activo. Sin embargo si abres otro formulario que no tenga los controles bloqueados si se podrán mover. Si añades más controles a un formulario bloqueado estos quedan bloqueados automáticamente.

Frame: proporciona un agrupamiento identificable para controles. También puede utilizar un Frame para subdividir un formulario funcionalmente por ejemplo, para separar grupos de controles OptionButton.





Tiene la siguiente forma:

Figura 1.28 Icono de acceso a Frame.

CheckBox y OptionButton (botones de elección y opción): El control *CheckBox*, o casilla de verificación, permite elegir una opción (activada / desactivada, True/False) que el usuario puede establecer o anular haciendo clic. Cada casilla de verificación es independiente de las demás que puedan existir en el formulario, pudiendo tomar cada una de ellas el valor True o False, a voluntad del operador.

Un control *OptionButton* muestra una opción que se puede activar o desactivar, pero con dependencia del estado de otros controles OptionButton que existan en el formulario.

Los controles OptionButton se agrupan dibujándolos dentro de un contenedor como un control Frame, un control PictureBox o un formulario. Para agrupar

controles OptionButton en un Frame o PictureBox, dibuje en primer lugar el Frame o PictureBox y, a continuación, dibuje dentro los controles OptionButton. Todos los controles OptionButton que están dentro del mismo contenedor actúan como un solo grupo, e independientes de los controles OptionButton de otros grupos distintos.

Figura 1.29 Icono de acceso a Check Button y Optio Button.

ListBox y ComboBox: Un control ListBox muestra una lista de elementos en la que el usuario puede seleccionar uno o más. Si el número de elementos supera el número que puede mostrarse, se agregará automáticamente una barra de desplazamiento al control ListBox.

Un control ComboBox combina las características de un control TextBox y un control ListBox. Los usuarios pueden introducir información en la parte del cuadro de texto y seleccionar un elemento en la parte de cuadro de lista del control. En resumen, un ComboBox es la combinación de un ListBox, que se comporta como si de un ListBox se tratase.

Figura 1.30 Icono de acceso a ListBox y ComboBox

Controles HScrollBar y VScrollBar: Son dos controles similares, para introducir un dato cuasi-analógico en una aplicación. Se toman directamente de la caja de herramientas, y tienen un aspecto parecido al de un control de volumen de un equipo de música. El HScrollBar está en posición horizontal, y el

VScrollBar en posición vertical, mediante estos controles se pueden introducir datos variando la posición del cursor.

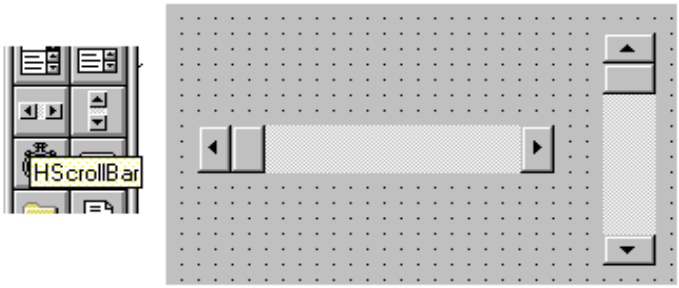


Figura 1.31 Icono de acceso a Controles: HScrollBar y VScrollBar.

Timer: Este objeto permite establecer temporizaciones. Presenta una novedad respecto a los controles estudiados hasta ahora. El control Timer solamente se ve durante el tiempo de diseño. En tiempo de ejecución, el control permanece invisible.

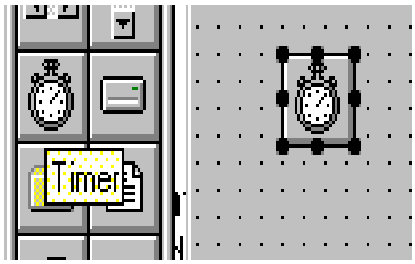


Figura 1.32 Icono de acceso a Timer.

La temporización producida por el Timer es independiente de la velocidad de trabajo del ordenador.

Shape: es un control gráfico que se muestra como un rectángulo, un cuadrado, una elipse, un círculo.

Utilice controles Shape en tiempo de diseño en lugar o además de invocar los métodos Circle y Line en tiempo de ejecución. Puede dibujar un control Shape en un contenedor, pero no puede actuar como contenedor. (Esto quiere decir que un control Shape nunca le servirá, por ejemplo, para albergar varios OptionButton y pretender que sean independientes de otros controles OptionButton que se encuentren fuera del control Shape.

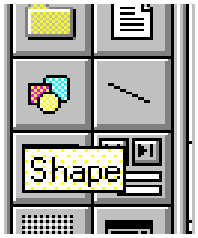


Figura 1.33 Icono de acceso a Shape.

Line: es un control gráfico que solamente sirve para poner una línea en un formulario. Del mismo modo, no tiene procedimientos, por lo que no sirve para aportar código al programa. Solo sirve para aportar una característica gráfica, es un adorno.

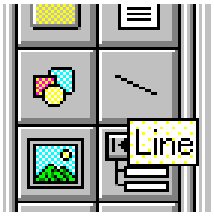


Figura 1.34 Icono de acceso a Line.

Control Gauge: este control presenta una información numérica de forma gráfica, bien como un display lineal (típico por ejemplo en ecualizadores de audio), o como una aguja. No está normalmente en la caja de herramientas, por lo que hay que traerla desde los Controles Personalizados (Menú desplegable de Herramientas) Se denomina MicroHelp Gauge Control. El archivo que lo contiene se denomina GAUGE16.OCX, 16 bits

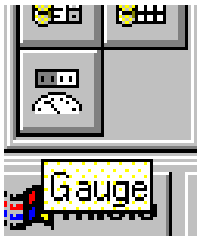


Figura 1.35 Icono de acceso a Control Gauge.

Mediante este control, podemos presentar una magnitud numérica de una forma cuasi-analógica. Podríamos decir que es un control similar al HScrollBar, que en vez de meter información a la aplicación, la presenta.

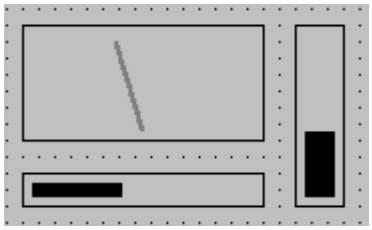


Figura 1.36 Imagen de presentación de la opción Gauge.

En la figura puede verse un Gauge de aguja, uno de barra horizontal y otro de barra vertical. Para mejorar la presentación, el Gauge permite poner un gráfico como fondo, cambiar el color de la barra, color de fondo, etc.

Nota para la distribución cuando cree y distribuya aplicaciones con controles Gauge, tendrá que instalar el archivo apropiado en el subdirectorio SYSTEM de Windows del cliente.

CommonDialog: este control no se presenta en tiempo de diseño más que con un simple icono.

Figura 1.37 Icono de acceso a CommonDialog.

El cuadro de diálogo, CommonDialog se utiliza para varias funciones:

- Abrir Ficheros
- Guardar Ficheros
- Elegir colores
- Seleccionar Impresora
- Seleccionar Fuentes

En realidad el cuadro de diálogo permite conocer datos con los cuales, y mediante el código adecuado, abriremos o guardaremos ficheros, elegiremos colores o seleccionaremos fuentes. Es decir, el CommonDialog “NO” realiza

más funciones que mostrar ficheros existentes, fuentes disponibles, colores, para que, mediante código, abramos esos ficheros o usemos una determinada fuente.

1.2.4 FUNDAMENTOS DE PROGRAMACION EN VB.

1.2.4.1 Comentarios y Variables

Comentarios: Visual Basic interpreta que todo lo que está a la derecha del carácter (**‘**) en una línea cualquiera de programa es un comentario y no ejecuta acción alguna. Por ejemplo:

```
‘Suma de números pares
Suma Pares = 0
‘Se inicializa la variable Suma Pares al valor 0
```

Los comentarios son de mucha utilidad para poder entender el código del programa utilizado.

Las variables: se utilizan valores temporalmente durante la ejecución del programa.

```
Nventas = 5500
```

Asigna el valor 5500 a la variable **Nventas**. El valor de una variable puede modificarse a lo largo de la ejecución del programa.

Una constante: es un valor que no cambia durante la ejecución del programa. Para declarar un dato como constante se utiliza la palabra **Const** en la declaración de la variable. Ejemplos:

- Const incremento = 2.5 ‘Las constante son privadas por defecto.
- Public Const saludo = “Bienvenido” ‘Declaración de una constante pública.
- Private Const altura as Integer ‘Declaración de un entero constante.
- Const año = “1999”, radio As Double = 4.45 ‘Multiples constantes.

Declaración de variables: Para ello se utiliza la sentencia **Dim**. Esta reserva espacio de memoria para la variable y permite a Visual Basic saber qué tipos de datos deberá guardar en dicha variable. Por ejemplo:

```
Dim nLongitud As integer
```

Después del nombre de la variable especifique el tipo de la misma. Por ejemplo, la variable nLongitud ha sido de tipo Integer (entero).

Tipo de datos: Visual Basic dispone de varios tipos de datos, aplicables tanto para constantes como para variables. La siguiente tabla muestra los tipos de datos disponibles en Visual Basic.

Tipo	Descripción	Carácter	Rango
Boolean	Binario		False o true
Byte	Entero corto		0 a 255
Integer	Entero (2 byte)	%	-32768 a 32767
Long	Entero largo(4 byte)	&	-2147483648 a 2147483647
Single	Real simple precisión (4bytes)	!	-3.40E +38 a 3.40E +38
Double	Real doble Precisión (8 bytes)	#	-1.79D + 308 a 1.79D +308
Currency	Número con punto decimal fijo(8bytes)	@	0 a 65500 caracteres

Tabla 1.1 Tipo de datos en Visual Basic.

Declaración explicita: para evitar errores se utiliza la sentencia **Option Explicit** en la sección de declaración del formulario del modulo. Option Explicit genera un mensaje de error si encuentra una variable no declarada explícitamente.

Option Explicit opera sólo en el formulario o en el módulo donde se haya puesto. Para tener esta opción activa para todo el código de una aplicación, haga clic en el comando opciones del menú herramientas.

Ámbito de las variables: Se entiende por ámbito de las variables al espacio de la aplicación donde la variable es reconocida y por lo tanto se puede utilizar.

Variables Locales: es reconocida solamente en el procedimiento en el que ha sido declarada. Fuera de ese procedimiento, la variable no es reconocida. Utilicé la sentencia Dim o Static para declarar una variable local a un procedimiento.

Variables Estadísticas: esta variable conserva su valor entre llamadas en el procedimiento y se destruirá solo cuando en el programa termine.

Variable a nivel del formulario: una variable declarada a nivel formulario puede ser compartida por todos los procedimientos de ese formulario. Para declarar una variable a nivel de formulario, haga doble clic sobre uno de los objetos contenidos en el formulario y, en la ventana de código, seleccione “(General)” del cuadro de lista Objetos y “(Declaraciones)” del cuadro de lista procedimientos.

Variable a nivel del módulo: se utiliza para compartir una variable entre los formularios y procedimientos contenidos en un proyecto, necesita declararla en un módulo de dicho proyecto.

Variables Públicas: esta variable puede ser utilizada desde cualquier parte de la aplicación, sin importar el módulo en el que esté el procedimiento que accede a ella. Hay que declarar en un módulo de la aplicación, en la sección de declaraciones generales, utilizando la palabra clave Public.

1.2.4.2 Operadores

Un operador en un símbolo o palabra que ejecuta una operación o maneja la información.

La siguiente tabla muestrea le conjunto de operadores que soporta Visual Basic.

Tipo	Operación	Operador
Aritmético	Potenciación	ˆ
	Cambio de signo	-
	Multiplicación y división	*,/
	División entera	\
	Resto de una división entera	+, -
Concatenación	Concatenar o enlazar	&
Relacional	Igual, distinto, menor, mayor, menor, igual, mayor o igual	=, < > , <, >, <=,>0
Lógico	Negación	Not
	And	And
	Or inclusiva	Or
	Or exclusiva	Xor
	Equivalencia	Eqv

Tabla 1.2 Operadores

1.3 MICROCODE STUDIO

1.3.1 CARACTERÍSTICAS GENERALES

Microcode es un programa editor de texto como un bloc de notas, pero con la diferencia que está hecho exclusivamente para facilitar la programación de los microcontroladores PIC.

Los procedimientos para programar son muy sencillos, los cuales deben estar acorde con el modelo de microcontrolador a usarse, con esto se debe elegir el microcontrolador en este caso es el PIC 16F870 el cual es uso de esta tesis, este programa de escritura no trabaja solo necesita de un compilador, ensamblador y programador para que se encuentre completo.

1.3.1.1 Configuración de Microcode[5]

Para poder utilizar de forma fácil y aprovechar las ventajas de este editor de texto programador se debe acoplar diferentes programas, como son el compilador y ensamblador que es el programa PicBasicPro y un programador como lo es IC-Prog 1.05E.

Los cuales son sumamente necesarios para una correcta generación de código hexadecimal, en cual va hacer utilizado por el microcontrolador PIC.

La instalación de programa Microcode es similar a la de cualquier programa la diferencia radica en que se debe intercalar los dos programas restantes, una vez instalado el programa se genera una carpeta con el nombre de **Mecanique** en la unidad C de memoria, es muy importante encontrar esta carpeta porque es la que va a contener los dos programas restantes.

Una vez generada la carpeta de contención se colocan los dos programas restantes en la misma carpeta esto se lo realiza para que el programa encuentre más fácilmente y rápido su compilador, ensamblador y programador quedando de la siguiente forma vista por el Explorador de Windows:

---

[5] REYES, Carlos; *Microcontroladores PIC Programación en Basic 16F62X, 16F8XX, 16F87X*, Editorial Rispergraf C.A, Segunda edición; Ecuador 2006

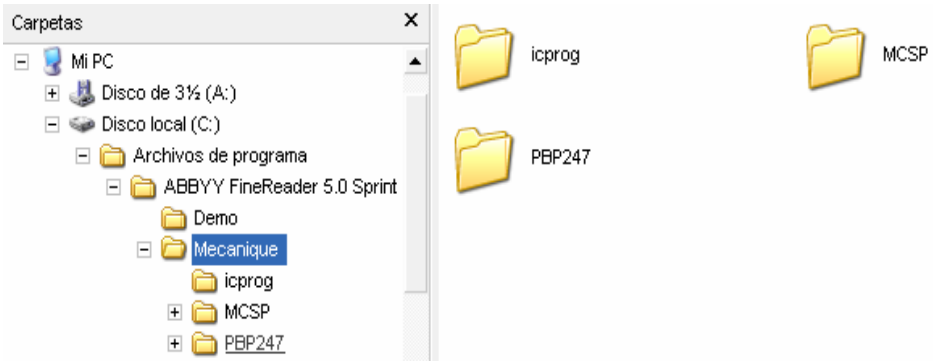


Figura 1.38 Imagen obtenida del explorador de la estructura jerárquica del programa.

Colocados los dos programas en la carpeta **Mecanique** se prosigue a la configuración de cada uno, para lo cual se abre el programa Microcode Studio y en la pantalla se busca la opción **View** y damos un clic en **PicBasic Options,** aparece una nueva ventana mas pequeña en donde se encuentra en compilador ensamblador y programador, el compilador y ensamblador debe pertenecer a la carpeta C:\archivos del programa\mecanique\pbp 247 y si no existen como se muestra en la ventana se debe dar clic en la opción *Find Manually* (Encontrar Manualmente) y encontrarlos por el explorador hasta poder colocarlos sino es así no funcionara el programa.

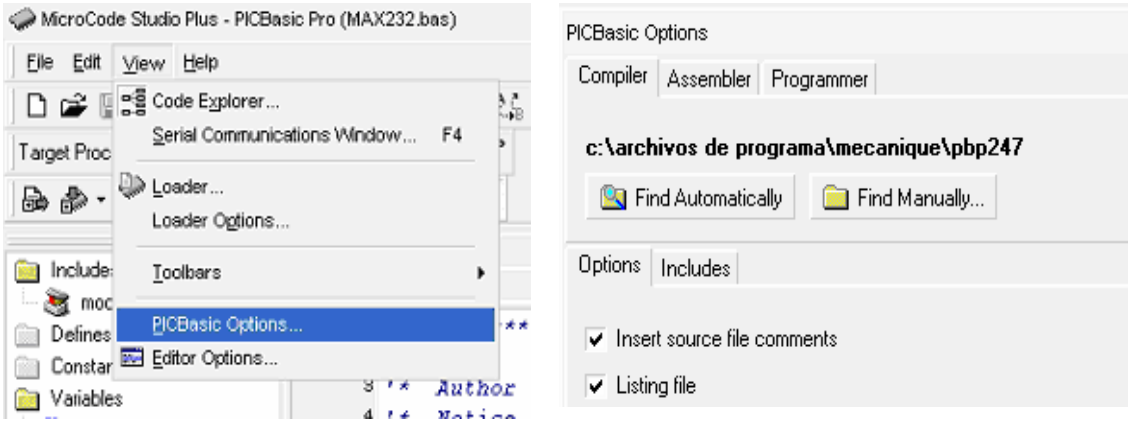


Figura 1.39 Ventanas de configuración del Microcode Studio

Corregido y encontrado el compilador y ensamblador con el PicBasicPro damos clic en la pestaña programador (programmer) pero no aparece.

El aspecto del programador lo podemos observar en la figura 1.40.

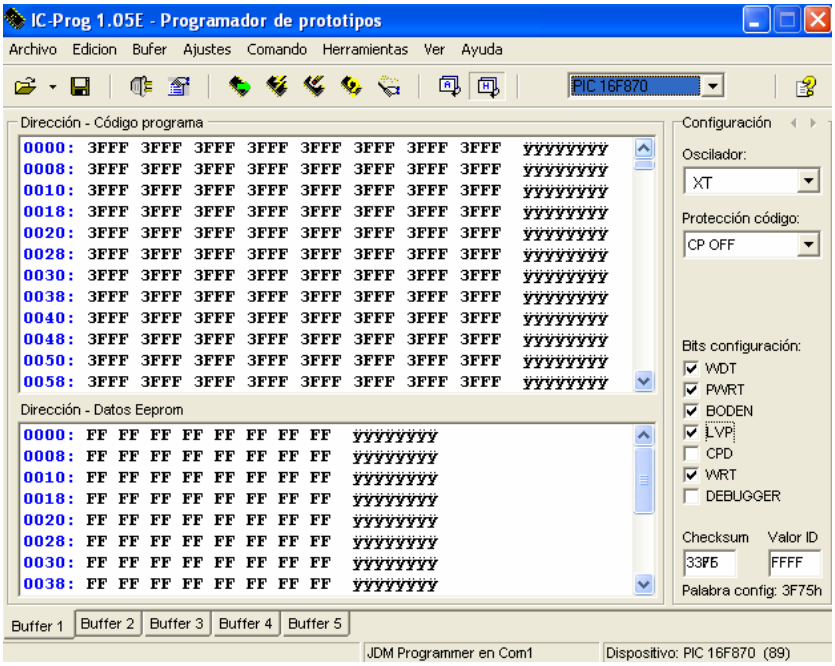


Figura 1.40 Ventana principal del programador IC-Prog 1.05E

Para que el programador **IC-Prog 1.05E,** funcione correctamente debemos tener instalado el driver. Caso contrario observará una serie de errores en la ejecución del programa.

Para instalarlo primero necesita encontrar el archivo **Icprog_drive.zip**, este archivo puede ser descargado por Internet completamente gratis, una vez descargado se procede a descomprimirle, y aparece la carpeta **icprog_driver,** en el cual contiene el archivo **icprog.sys**. Este archivo se debe mover junto al archivo ejecutable **icprog.exe** sólo de esta manera se lo podrá activar.

Una vez instalado el driver es importante habilitar al check box “Habilitar Driver NT/2000/XP” en caso de usar Windows NT, 2000 ó XP.

Para habilitar el **Driver**, vaya a la ventana principal del programador IC-Prog 1.05E se da clic en **Ajuste** y luego de un clic en **Opciones,** esta vez de un clic en **miscelánea** para habilitar driver señalar **Normal** y **Habilitar Driver NT/2000/XP** y luego clic en **OK**, ver figura 1.41.

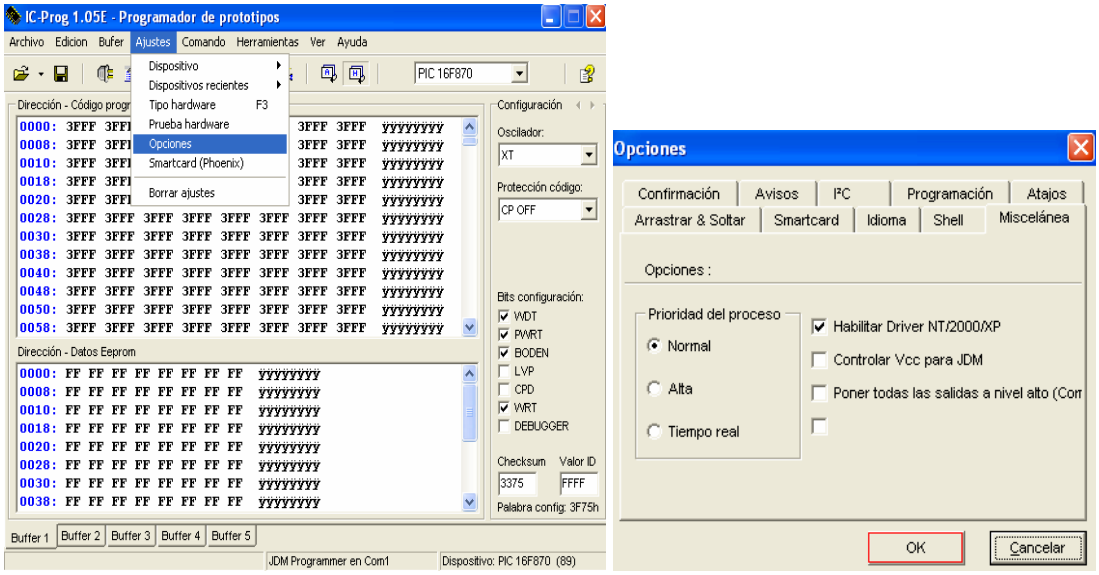


Figura. .141 Ventana para habilitar el Driver NT/2000/XP

El IC-Prog1.05E, así que debe crearlo con la finalidad de que a futuro pueda acceder desde el Microcode y programar directamente, para lo cual en la ventana anterior figura 1.39 hacemos clic en la pestaña Programmer y colocamos **Add New Programmer,** inmediatamente aparece otra ventana figura 1.42 en la cual damos clic en **create a custom programmer entry,** y luego en **next**.

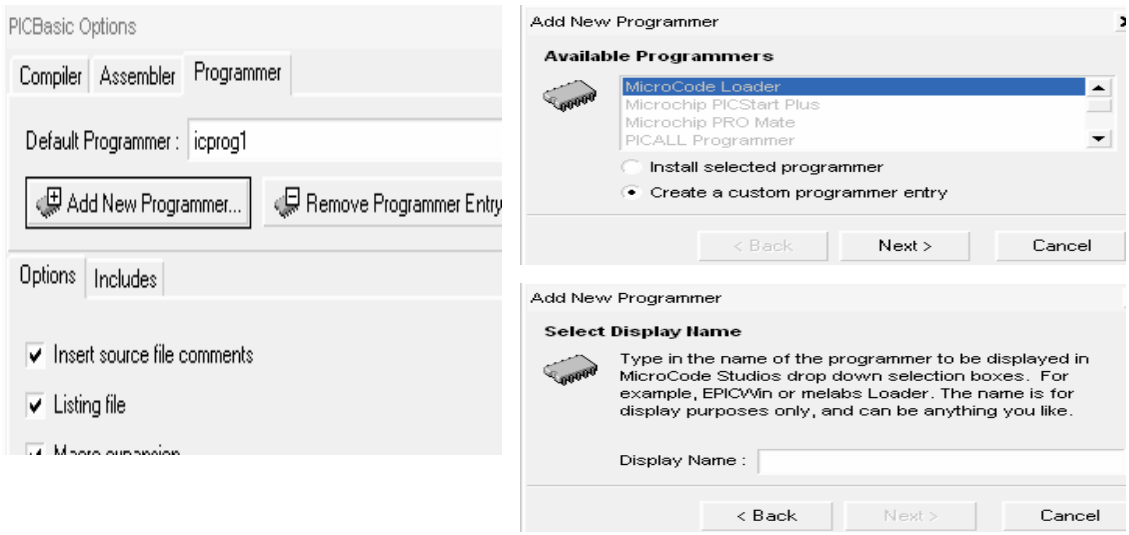


Figura 1.42 Ventanas de configuración del programador.

En la siguiente ventana colocamos el nombre del programador y siguiente, luego en **Programmer Filename:** Aquí colocamos el ejecutable del programa que es **icprog.exe** y presionamos encontrar automáticamente, una vez encontrado nos pide los parámetros pero no se debe colocar nada y dar un clic en **finalizar**.

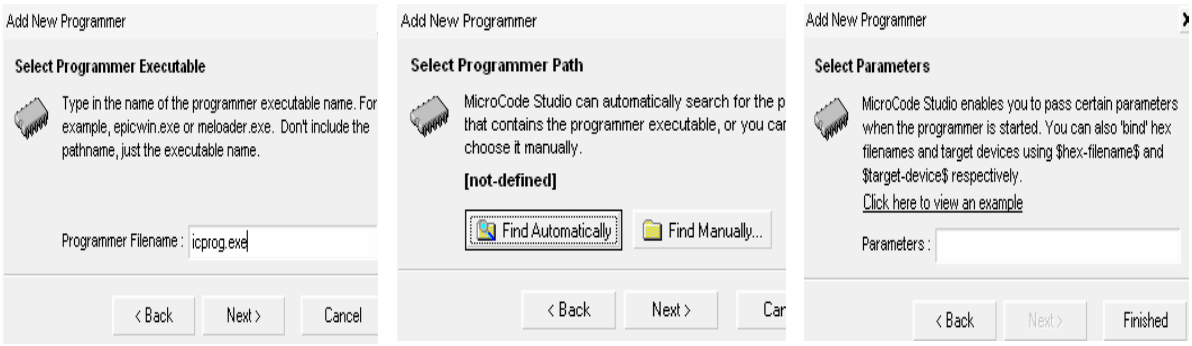


Figura 1.43 Ventanas de localización del programador utilizado por Microcode Studio.

Ya configurado todo como lo hemos hecho esta listo para ser usado Microcode Studio, es muy fácil de utilizar si se conoce cada una de las herramientas que nos ofrece para la programación de microcontroladores PIC, en la siguiente imagen damos a conocer cada una de las partes de la pantalla que conforma Microcode Studio.

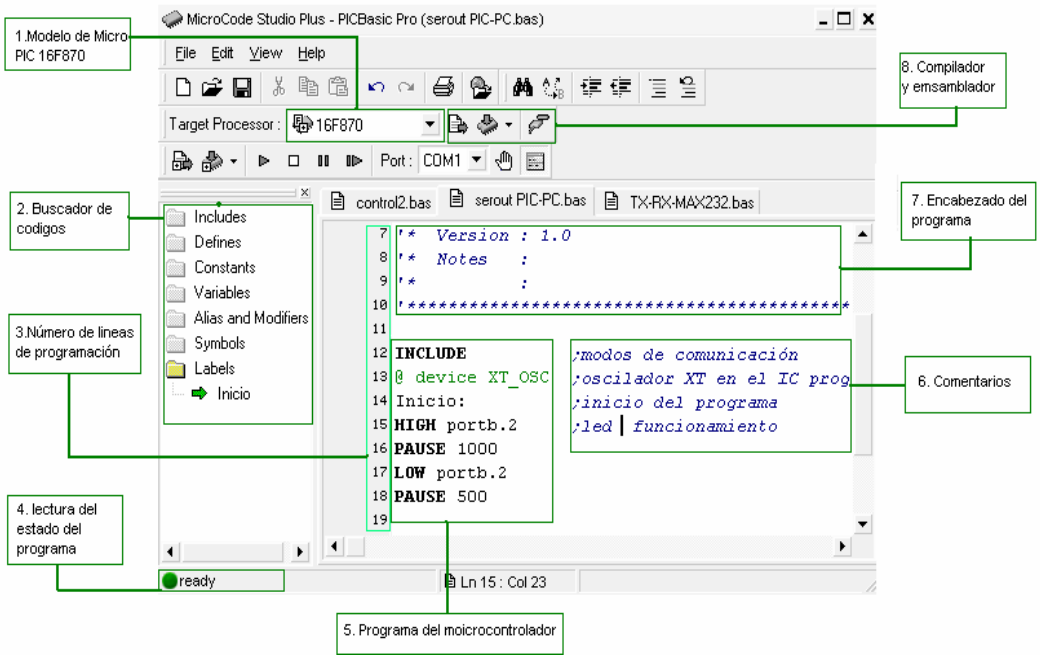


Figura 1.44 Partes de la pantalla del programa Microcode.

1.3.2 UTILIZACIÓN DE SET DE INSTRUCCIONES Y PROGRAMACIÓN[6]

Para la programación en Microcode se utiliza el set de instrucciones proporcionado por el programa PicBasicPro, el cual es un lenguaje de alto nivel cuyo objeto es realizar las líneas de programación para el microcontrolador.

Cada instrucción tiene una tarea especifica, dando, así a constituirse en las instrucciones que debe seguir el PIC en el cual va ha ser grabado, estas instrucciones no van directamente a colocarse en el PIC sino que se lo compila y ensambla para cambiarlo a datos en hexadecimal los cuales estos son grabados en los microcontroladores.

El set de instrucciones leído e interpretado por Microcode no puede ser tomado como datos o variables por lo cual se coloca en negrilla y mayúsculas, en la tabla siguiente se encuentra las instrucciones con su respectiva acción en la programación, ver tabla 1.3.

PicBasicPro 2.47 Set de Instrucciones	
@	Inserta una línea de código ensamblador.
ADCIN	Lee el conversor análogo.
ASM..ENDASM	Inserta una sección de código ensamblador.
BRANCH	GOTO computado (equiv. to ON…GOTO).
BRANCHL	BRANCH fuera de pagina (BRANCH largo).
BUTTON	anti-rebote y auto_ repetición de entrada en el pin
CALL	Llamada a subrutina de ensamblador.
CLEAR	Hacer cero en todas las variables.
CLEARWDT	Hace cero el contador del Watchdog Timer.
COUNT	Cuenta el número de los pulsos en un pin.
DATA	Define en contenido inicial en un chip EEPROM.
DEBUG	Señal asincrónica de salida en un pin y baud.
DEBUGIN	Señal asincrónica de entrada en un pin fijo y baud.
DISABLE	Deshabilita el procesamiento de ON INTERRUP
DISABLE DEBUG	Deshabilita el procesamiento de ON DEBUG.
DISABLE INTERRUPT	Deshabilita el procesamiento de ON INTERRUPT.

[6] http://picbasic.com/products/pbpis.htm

DTMFOUT	Produce tonos telefónicos en un pin.
EEPROM	Define contenido inicial en un chip EEPROM.
ENABLE	Habilita en procesamiento de ON INTERRUPT.
ENABLE DEBUG	Habilita en procesamiento ON DEBUG.
ENABLE INTERRUPT	Habilita el procesamiento ON INTERRUPT.
END	Detiene la ejecución e ingresa en modo de baja potencia.
ERASECODE	Borrar bloque de códigos de la memoria
FOR...NEXT	Ejecuta declaraciones en forma repetitiva.
FREQOUT	Produce hasta 2 frecuencias en un pin.
GOSUB	Llama a una subrutina BASIC en la línea especifica.
GOTO	Continua la ejecución de la línea especifica.
HIGH	Saca un 1 lógico
HPWM	Salida de hardware con ancho de pulsos modulador.
HSERIN	Entrada serial asincrónica (hardware).
HSERIN2	Entrada serial asincrónica en segundo puerto.
HSEROUT	Salida serial asincrónica (hardware).
HSEROUT2	Entrada serial asincrónica en segundo puerto.
I2CREAD	Leer bytes de dispositivos I^2C
I2CWRITE	Graba bytes de dispositivos I^2C
IF..THEN..ELSE..ENDIF	Ejecute declaraciones condicionales.
INPUT	Convierte un pin en entrada.
LCDIN	Lee características desde una RAM de un LCD.
LCDOUT	Muestra características en un LCD
{LET}	Asigna el resultado de una expresión a una variable.
LOOKDOWN	Busca un valor constante 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 cero lógico a un pin especifico..
NAP	Apaga el procesador por un corto tiempo.
ON DEBUG	Ejecuta un DEBUG en Basic.
ON INTERRUPT	Ejecuta una subrutina BASIC en una interrupción.
OWIN	Entrada de dispositivos es un alambre.
OWOUT	Salida a dispositivos es unalambre.
OUTPUT	Convierte un pin en salida.
PAUSE	Demora una resolución 1 milisegundo.
PAUSEUS	Demora una resolución 1 microsegundo.
PEEK	Lee bytes desde un registro.
PEEKCODE	Lee bytes desde un espacio de código

POKE	Escribe bytes en un registro.
POKECOD	Escribe bytes en espacio de código programando en tiempo.
POT	Lee potenciómetros especificando el pin.
PULSIN	Mide el ancho de pulso en un pin.
PULSOUT	Genera un pulso hacia un pin.
PWM	Salida modulada en ancho de pulso por un pin especifico.
RANDOM	Genera un número seudo aleatorio.
RCTIME	Mide el ancho de pulso de un pin.
READ	Lee bytes del un chip de la EEPROM.
READCODE	Lee palabras desde un código de memoria.
REPEAT...UNTIL	Ejecuta declaraciones con condiciones de verdad.
RESUME	Continua ejecutando después de una interrupción.
RETURN	Continua en la declaración que sigue al último GOSUB.
REVERSE	Convierte un pin en salida en entrada.
SELECT CASE	Compara una variable con diferentes valores.
SERIN	Entrada señal asincrónica tipo (BS1).
SERIN2	Entrada señal asincrónica tipo (BS2).
SEROUT	Salida señal asincrónica (tipo BASIC stamp 1).
SEROUT2	Salida señal asincrónica (tipo BASIC stamp 2).
SHIFTIN	Entrad de señal sincrónica.
SHIFTOUT	Salida de señal sincrónica.
SLEEP	Apagar en procesador por un tiempo.
SOUND	Generar un tono o ruido blanco en un pin.
STOP	Detiene el programa de ejecución
SWAP	Intercambia los valores de dos variables.
TOGGLE	Hace salida a un pin y cambia su estado.
USBINIT	Inicializar USB
USBOUT	Salida USB.
WHILE..WEND	Ejecuta declaraciones mientras la condición sea cierta.
WRITE	Graba bytes en un chip EEPROM.
WRITECODE	Escribe palabras en código de memoria.
XIN	Entrada X -10.
XOUT	Salida X -10.

Tabla 1.3 Set de instrucciones PicBasicPro

Especificación detallada de cada instrucción utilizada en PicBasicPro en el (ANEXO 2).

1.3.2.1 Funciones y operaciones:

Todas las operaciones matemáticas son designadas y registradas con 16 bit de precisión. Las operaciones que soporta son:

Operaciones Matemáticas	Descripción
+	Suma
-	Resta
*	Multiplicación
**	Multiplicación tope 16 bits
*/	Multiplicaron media 16 bits
/	División
//	Remanente (Modulo)
<<	Ir a la izquierda
>>	Ir a la derecha
ABS	Valor absoluto
COS	Coseno
DIG	Digito
MAX	Máximo*
MIN	Mínimo*
NCD	de código
REV	Invertir Bits
SIN	Sino
SQR	Cuadro or
&	AND
|	OR
^	Exclusive OR
~	NOT
&/	NOT AND
|/	NOT OR
^/	NOT Exclusive OR
=	Operador de asignación

Operadores de Comparación	Descripción
= o = =	Igual
<> o !=	No Igual
<	Menor
>	Mayor
<=	Menor o Igual
>=	Mayor o Igual

Operadores Lógicos	Descripción
AND o &&	AND Lógico
OR o ||	OR Lógico
XOR o ^ ^	OR Exclusivo
NOT AND	NAND lógico
NOT OR	NOR lógico
NOT XOR	NXOR lógico

Tabla 1.4 Operaciones aritméticas, comparaciones y lógicas

1.3.2.2 Declaraciones en programación de Microcode

Tipos de Variables:

```
A    var  byte
B    var  bit
C    var  Word
D    var  w0.byte0; cuatro es el primer byte de Word 0
E    var  uno.0;    cinco es el bit 0 de uno
F    var  portb.0;  seis  es un alias del bit 0 del puerto B
```

- Bit (un bit de longitud, almacena 0 o 1 únicamente)
- Byte (un byte de longitud, almacena números enteros entre 0 y 255)

- Word (dos bytes de longitud, almacena números enteros entre 0 y 65,535).

Creación de Constantes:

binario con %100 ; definición de una constante binaria.
hexadecimal con $100 ; definición de una constante hexadecimal.

Llamada a archivos externos:

Incluye “bs1defs.bas”
Definición de la Velocidad del Reloj:
DEFINE OSC 4; define la velocidad del reloj a 4Mhz.

Arreglos o Vectores:

Vector1 **var** bytes[10] ; vector1 typo bytes y tiene 10 elementos.
Vector2 **var** bit[8] ; vector2 typo bit y tiene 8 elementos.

La primera posición de un vector es la posición cero. Los límites para el número de elementos que puede tener un vector es:

BIT 128, BYTE 64, WORD 32

Manejo de Puertos:

led **var** portb.0 ; da el nombre de led al pin 0 del puerto b
led **var** portb.1 ; da el nombre de led al pin 1 del puerto b
led **var** portb.2 ; da el nombre de led al pin 2 del puerto b
portb = %01010101 ; asigna un valor en binario al puerto b

Para indicar si el puerto es de salida o entrada se utiliza la instrucción TRIS ejemplo:

- TRISB = 0 ; indica que el puerto B sea de salida.
- TRISA = 1 ; indica que el puerto A sea de entrada.
- TRISB.0 = 0 ; indica que el pin 0 del puerto B sea de salida.
- TRISA.0 = 1 ; indica que el pin 0 del puerto A sea de entrada.

1.3.2.3 Utilización y programación de Microcode

Una vez colocada y detallada cada instrucción a utilizarse y teniendo en claro la tarea que ejecutará el microcontrolador PIC se prosigue a la programación, para esto se debe tener en cuenta el hardware que será conectado a esté, la programación debe seguir una estructura como cualquier programación habitual, en otros simuladores o programadores de PIC como es el PIC Simulator, MLAP, Visual Basic, colocando primero las variables y constantes a ser utilizadas, luego las instrucciones a ejecutarse y por ultimo cerrando instrucciones con lazos.

Cada instrucción debe estar bien definida y ser lógica para que no existan errores en la parte de ejecución, pues el PIC solo realiza lo que le hemos programado quedando así toda la responsabilidad de ejecución al programador.

Aquí podemos ver un programa ya escrito y ejecutado, todo programa puede tener errores de sintaxis o mal escritura en las declaraciones de ejecución por la cual al ser compilado, el corrector del programa explica y señala la línea de error, siendo de gran ayuda para este tipo de correcciones.

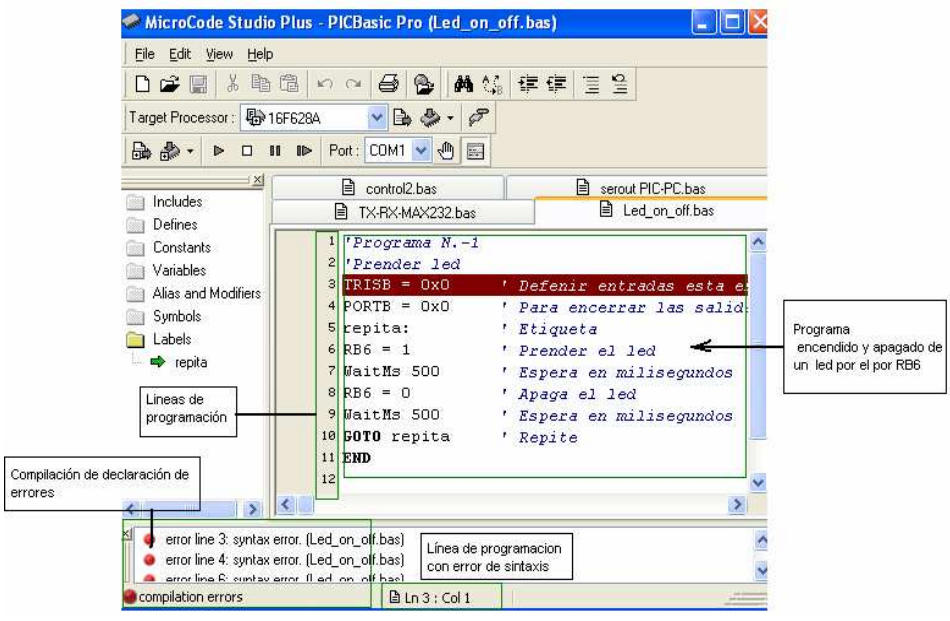


Figura 1.45. Microcode ventana de visualización de errores de programación.

En cada paso para la realización del programa de ejecución debemos tener en cuenta las líneas de almacenaje del PIC a utilizarse esto es importante pues si no se esta bien dimensionado no funciona, en estos caso debe cambiar de PIC y listo, el programa Microcode al compilar y ensamblar nos da el espacio que esté va a ocupar el PIC.

Una vez corregido los errores de programación en la parte inferir izquierda tenemos el tamaño de memoria que ocupara en el PIC.

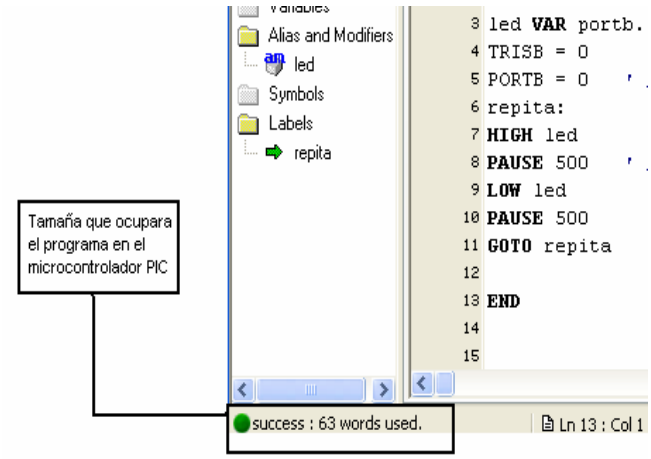


Figura 1.46. Imagen del tamaño que ocupara en el PIC.

Si todo esta bien compilado y ensamblado se crea el código hexadecimal el cual va ha servir para programar el microcontrolador PIC con el programa IC-Prog 1.05E, tomando de la carpeta de almacenaje de programas, el programa con extensión **.HEX** y colocándole en el programador.

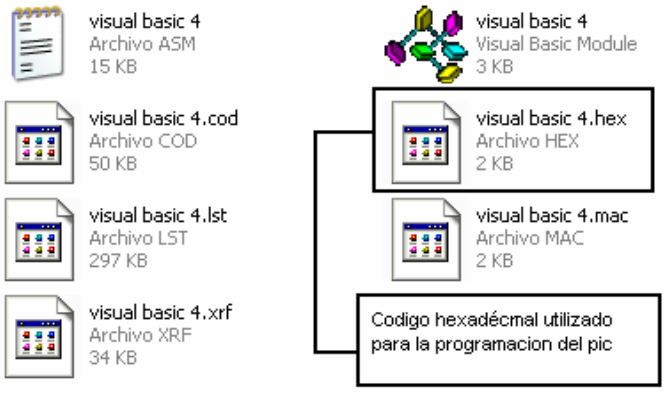


Figura 1.47 Imagen de archivos creados por Microcode Studio.

Para utilizar el código hexadecimal en el programador solo se lo debe abrir en la opción archivo, abrir archivo y buscarlo con el Explorador en la carpeta que esta almacenado y listo, el código se carga automáticamente, es importante seleccionar el tipo de PIC que se va a programar.

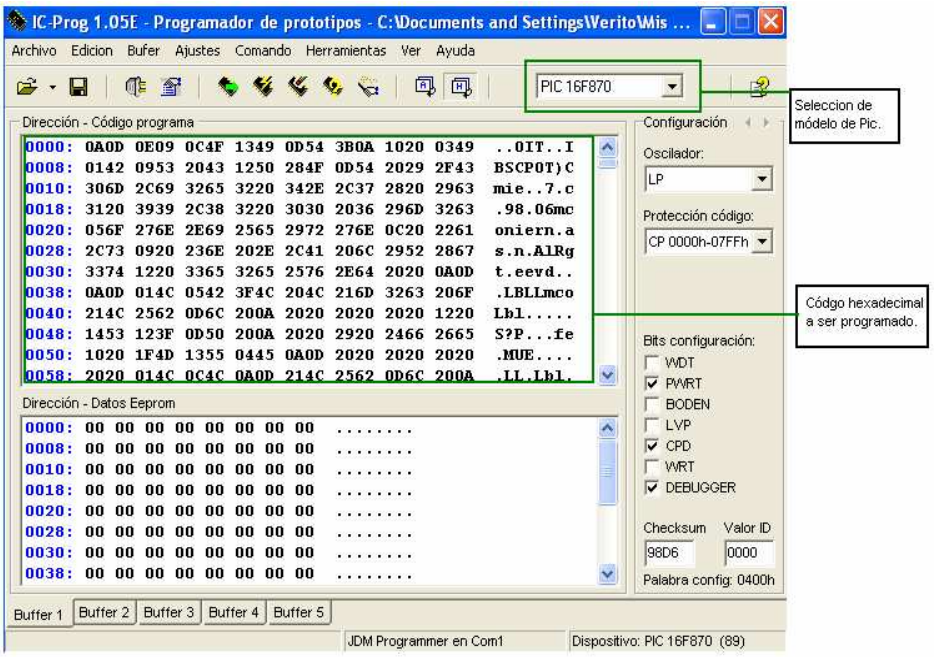


Figura 1.48 Imagen del programador IC-Prog 1.05E

Después de abrir el archivo .HEX (no antes) configure el oscilador en la ventana oscilador en la parte derecha de la ventana, intr. I/O (Oscilador interno resistencia condensador pin de I/O los A6 y A7), en el MCLR (reset externo debe estar deshabilitado, la protección de código apagada, esto depende de la utilización que va realizar el PIC.

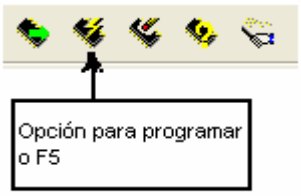


Figura 1.49 Imagen de controles de programación del Programa IC-Prog1.05E

Si ya está colocado el PIC en el grabador de microcontroladores y cargado el programa ha ser grabado, coloque el puntero en la opción con un relámpago

(grabar) y de un clic o presionando F5 debe aparecer una venta de información que esta verificando la dirección y el estado del PIC.

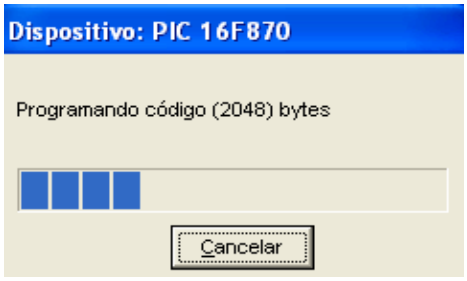


Figura 1.50 Imagen de comprobación y programado del PIC.

Si todo esta bien el PIC queda programado con la información en hexadecimal, pero si existe algún error después de ejecutado aparece una ventana de error, esta ventana de error puede aparecer cuando el programador físico no esta conectado al puerto correcto, no se a insertado bien el microcontrolador PIC, cuando se graba un PIC con protección de código o ya el PIC por mal uso se encuentra dañado.

Figura 1.51 Imagen de visualización de error en el momento de programación.

El microcontrolador no solo se puede programar de esta forma existen el mercado muchos programadores y simuladores de programación pero las herramientas que hemos utilizado son de bajo costo y para tener mejor control al momento de programar ya que un programador especializado tiene su costo adicional y por lo tanto también sus ventajas.

Con todo lo anterior el PIC esta listo para realizar su tarea y armar su circuito electrónico.

1.4 TRANSMISIÓN DE DATOS

1.4.1 CARACTERÍSTICAS GENERALES

1.4.1.1 Métodos de comunicación.

1.4.1.1.1 Comunicación símplex

Una comunicación es símplex si están perfectamente definidas las funciones del emisor, del receptor y la transmisión de los datos siempre se realiza en una sola dirección. La transmisión de señales por medio de la televisión es el ejemplo más claro de comunicación símplex.

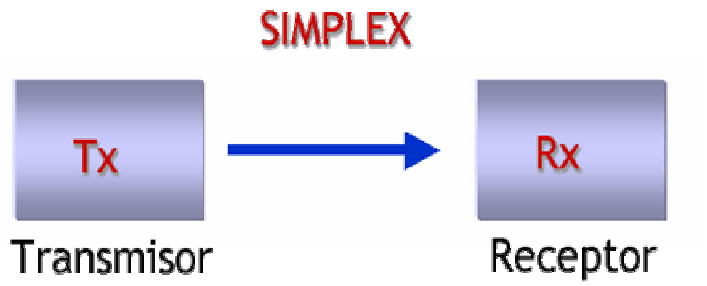


Figura 1.52 Transmisión simplex

1.4.1.1.2. Comunicación semidúplex(half-duplex)

En las comunicaciones semidúplex puede ser bididireccional, esto es, emisor y receptor pueden intercambiarse los papeles. Sin embargo, la bidireccionalidad no puede ser simultánea. Cuando el emisor transmite, el receptor necesariamente recibe. Puede ocurrir lo contrario siempre y cuando el antiguo emisor se convierta en el nuevo receptor.

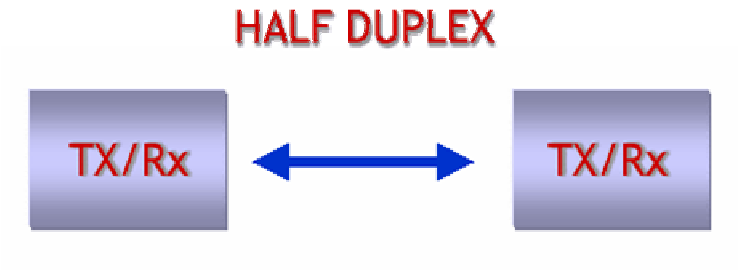


Figura 1.53 Transmisión half-duplex

1.4.1.1.3. Comunicación dúplex o full dúplex

En este tipo de comunicación es bidireccional y simultánea. Por ejemplo el teléfono. En ella el emisor y el receptor no están perfectamente definidos. Ambos actúan como emisor y como receptor indistintamente. En una comunicación dúplex se dice que hay un canal físico y dos canales lógicos.

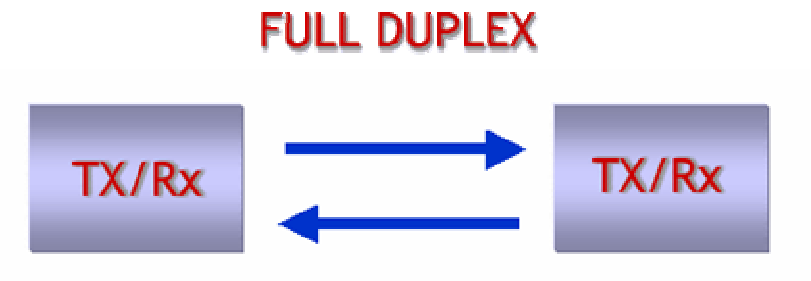


Figura 1.54 Transmisión full-duplex

1.4.1.2 Tipos de conexiones directas.

Conexiones Punto a Punto

Enlace de comunicación entre dos objetos finales o sistemas de computadoras en una red informática. El enlace puede ser dedicado (permanente), o temporal.

Conexiones Multipunto

En estas conexiones se conectan múltiples dispositivos al enlace que se ramifican desde un único punto. Generalmente, el dispositivo que proporciona la conexión es un controlador inteligente, que manejan el flujo de información de los múltiples dispositivos unidos a ella.

1.4.2 TIPOS DE TRANSMISION

1.4.2.1 Características Serie y Paralelo

Los dos tipos de transmisión que se pueden considerar son *serie* y *paralelo*. Para transmisión serial los bits que comprenden un carácter son transmitidos secuencialmente sobre una línea; mientras que en la transmisión en paralelo

los bits que representan el carácter son transmitidos serialmente. Si un carácter consiste de ocho bits, entonces la transmisión en paralelo requerirá de un mínimo de ocho líneas. Aunque la transmisión en paralelo se usa extensamente en trasmisiones de computadora a periféricos, no se usa aparte que en transmisiones dedicadas por el costo que implica el uso de circuitos adicionales.

1.4.2.2 La transmisión serial.

La transmisión serial es más lenta que la paralela puesto que se envía un bit a la vez. Una ventaja significativa de la transmisión serial en relación a la paralela es un menor costo del cableado puesto que se necesita un solo cable se tiene un octavo del costo que se ocuparía para transmisión paralela. Este ahorro en costo se vuelve más significativo conforme sean mayores las distancias requeridas para la comunicación.

Como ya se dijo, la información intercambiada entre computadora y otros sistemas digitales está constituida por paquetes de bit, denominados caracteres, de extensión fija (típicamente 8 bit). La elaboración de la información se efectúa normalmente considerando el entero carácter, es decir, elaborando contemporáneamente ***en paralelo*** todos los bits del carácter mismo.

La transmisión serial reduce drásticamente el número de las líneas necesarias, y puede realizarse también en líneas físicas con solamente 2 cables. La transmisión serial es la utilizada en el mundo de la transmisión de datos.

Dentro de las computadoras, los datos se elaboran en paralelo, y antes de la transmisión tendrán que convertirse en un formato serial. Esto lo proporcionan los dispositivos tipo UART, USART (Universal Synchronous/Asynchronous Receiver Transmitter), SIO (Serial Input Output), o aparatos más complejos (Control Unit, Front End Processor) en grado de gestionar más de una línea de comunicación serial.

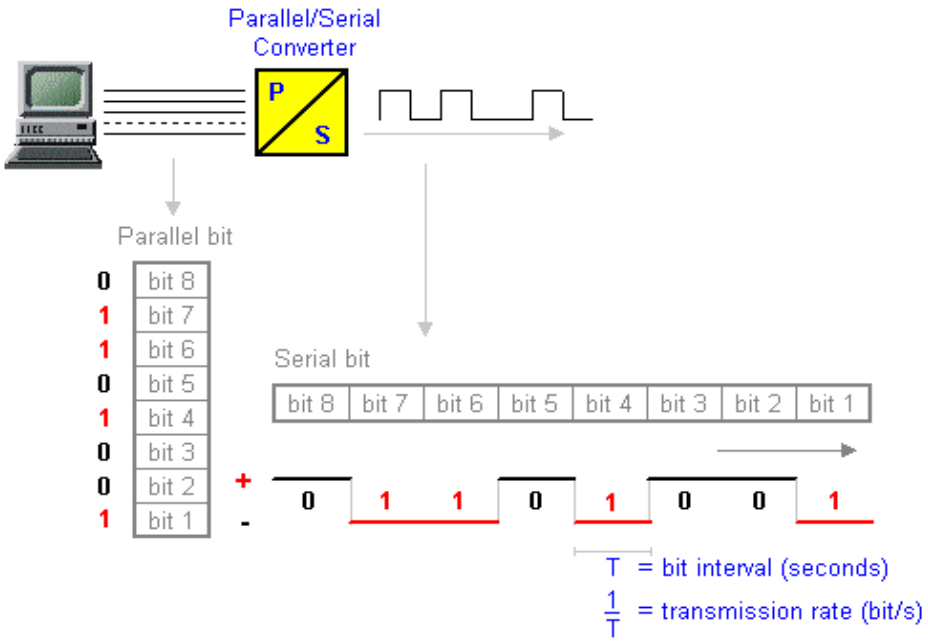


Figura 1.55 Transmisión serial de una trama de bits.

Otra ventaja importante de la transmisión serial es la habilidad de transmitir a través de líneas telefónicas convencionales a mucha distancia, mientras que la transmisión en paralelo esta limitada en distancia en un rango de metros.

1.4.2.3 La transmisión paralelo

La transmisión en paralelo todos los bits del carácter son enviados simultáneamente, un ejemplo de transmisión paralela es la que se tiene entre Computadora e Impresora mediante la llamada puerta paralela. Este tipo de transmisión requiere una línea para cada bit, además de algunas líneas para las señales de control. Se utiliza cuando la distancia es limitada, dentro del radio de algunas decenas de metros.

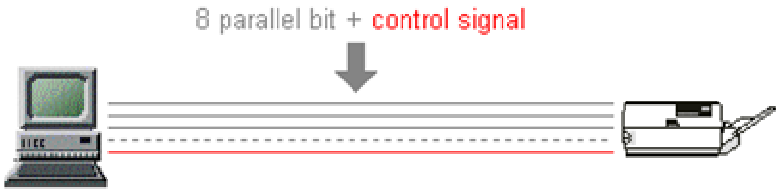


Figura 1.56 Transmisión de datos en paralelo

1.4.3. TÉCNICAS DE TRANSMISIÓN

1.4.3.1 Transmisión asíncrona

La transmisión asíncrona es aquella que se transmite o se recibe un carácter, bit por bit añadiéndole *bits de inicio*, y bits que indican el término de un paquete de datos, para separar así los paquetes que se van enviando/recibiendo para sincronizar el receptor con el transmisor. El bit de inicio le indica al dispositivo receptor que sigue un carácter de datos; similarmente el bit de término indica que el carácter o paquete ha sido completado.

1.4.3.2 Comunicaciones serie asíncronas

Los datos serie se encuentran encapsulados en tramas de la forma:

Figura 1.57 Trama de comunicación asincrónica

Primero se envía un bit de **start,** a continuación los bits de datos (primero el bit de mayor peso) y finalmente los bits de STOP.

El número de bits de datos y de bits de Stop es uno de los parámetros configurables, así como el criterio de paridad par o impar para la detección de errores. Normalmente, las comunicaciones serie tienen los siguientes parámetros: **1 bit de Start, 8 bits de Datos, 1 bit de Stop y sin paridad.**

En esta figura se puede ver un ejemplo de la transmisión del dato binario 10011010. La línea en reposo está a nivel alto:

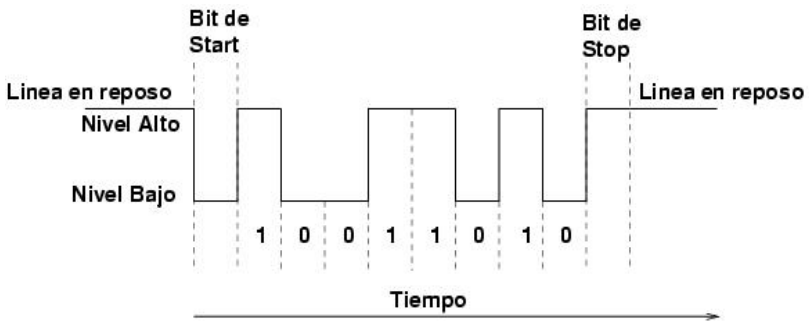


Figura 1.58 Imagen de transmisión serial asincrónica de byte.

1.4.3.3 Transmisión síncrona.

Este tipo de transmisión el envío de un grupo de caracteres en un flujo continúo de bits. Para lograr la sincronización de ambos dispositivos (receptor y transmisor) ambos dispositivos proveen una señal de reloj que se usa para establecer la velocidad de transmisión de datos y para habilitar los dispositivos conectados a los módems para identificar los caracteres apropiados mientras estos son transmitidos o recibidos. Antes de iniciar la comunicación ambos dispositivos deben de establecer una sincronización entre ellos. Para esto, antes de enviar los datos se envían un grupo de caracteres especiales de sincronía. Una vez que se logra la sincronía, se pueden empezar a transmitir datos.

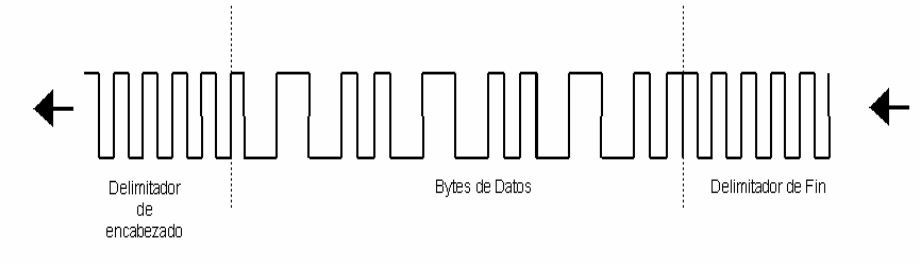


Figura 1.59 Imagen de transmisión serial sincrónica de byte.

Por lo general los dispositivos que transmiten en forma síncrona son más caros que los asíncronos. Debido a que son más sofisticados en el hardware. A nivel mundial son más empleados los dispositivos asíncronos ya que facilitan mejor la comunicación.

1.4.3.4 Nivel enlace

El nivel de enlace tiene las siguientes funciones:

- Entramado.
- Códigos detectores de errores (poner el código).
- Control de errores.
- Control de flujo.

La recepción es más lenta que la transmisión, por lo tanto los buffers se pueden llenar y perder la información, el control de flujo hace reducir la velocidad de transmisión hasta la velocidad de la recepción.

1.4.3.4.1 Entramado

La información que le llega al enlace se empaqueta y se le añade una cabecera y una cola formando así la trama.

Figura 1.60 Trama de transmisión de byte.

1.4.3.4.2 Código detector de errores

Generalmente la cola es el **código detector de errores** y opcionalmente puede estar el sincronismo de trama. En la cabecera suele estar el sincronismo de trama y la información de control.

Sincronismo de trama: tenemos que diferenciar cuando empieza y acaba una trama, el nivel físico no ve tramas sino caracteres. Según cual sea sincronización del nivel físico podemos tener dos tipos de distinción de tramas:

Dentro de la trama es orientado a carácter: para diferenciar el comienzo y final de trama utilizaremos caracteres especiales.

STX ≡ Start of Text ≡ Principio de trama.

ETX ≡ End of Text ≡ Final de trama.

Figura 1.61 Principio y Fin de una Trama de byte.

Si dentro de la trama hay un ETX hay que diferenciarlo, para ello existe la técnica Character Stuffing (transparencia de la información) que utiliza el carácter DLE (Data Link Escape) el cual se añade al STX y ETX de principio y final de trama.

Figura 1.62 Visualización de componentes de detección de errores en una Trama.

Pero, también nos podemos encontrar la secuencia DLE+ETX dentro de los datos de la trama, por lo tanto se pueden producir errores, para solucionarlo añadimos un DLE si se encuentra otro carácter DLE

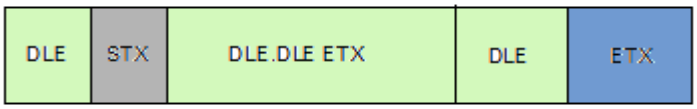


Figura 1.63 Técnica de detección de errores

Ejemplo: datos → A DLE ETX

Trama → DLE + STX | A + ~~DLE~~ + DLE + ETX | DLE + ETX

Otra técnica para delimitar tramas seria indicar la longitud del campo de datos.

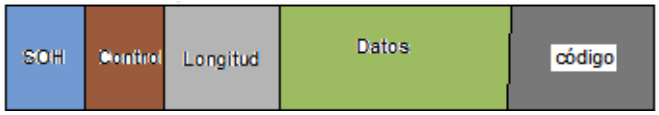


Figura 1.64 Imagen de control de flujo de la trama.

SOH → Start Of Head, indicación de principio de cabecera.

Orientado a BIT: podemos utilizar un campo para indicar la longitud o utilizar flags de apertura y cierre. Un flag de apertura y cierre es el siguiente

01111110 → seis unos

Dentro de los datos puede haber una secuencia como el del flag de apertura y cierre. Por lo tanto hay que utilizar otra vez la transparencia de la información (BIT Stuffing). Esta técnica lo que hace es que justo después de cinco unos inserta un cero y continua.

Ejemplo: 01111110 | 100101011111~~0~~10011 | 01111110

01111100 →011111~~0~~00

1.4.4 CARACTERISTICAS DE TRANSMISIÓN

1.4.4.1Velocidad de transmisión *(baud rate)*

Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*). Por ejemplo, 300 baudios representan 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800Hz.

1.4.4.2 Bits de datos

Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usara para referirse a todos los casos.

1.4.4.3 Bits de parada

Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo la transmisión será más lenta.

1.4.4.4 Paridad

Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par.

Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

1.5 COMUNICACIÓN SERIAL RS232[7]

1.5.1 NORMA RS 232

Los datos lógicos en los ordenadores están representados por bits (**bi**nary digi**ts**). El bit es una construcción intelectual, representada en el ordenador por un voltaje determinado. Cuando los bits deben enviarse dentro del propio ordenador o hacia el exterior, se transmiten a través de cables como cualquier voltaje.

Los bits se agrupan en unidades determinadas que proporcionan un esquema lógico mayor. Por ejemplo, un byte (octeto) está formado por una serie de ocho bits. Estos ocho bits pueden ser unos o ceros indistintamente, son $2^8 = 256$ combinaciones posibles.

[7] CAMPBELL, J. 1988. El libro del RS232. Anaya

1.5.1.1 RS-232-C (RS 232, RS-232)

En 1969 la **EIA** (Asociación de Industrias Electrónicas), conjuntamente con los Laboratorios Bell y los fabricantes de equipos de comunicaciones, formularon el EIA RS-232-C. El propósito inicial fue la conexión entre un Equipo Terminal de Datos (DTE) y un Equipo de Comunicación de Datos (DCE), empleando un intercambio de datos binarios en serie.

Actualmente, la conexión RS-232-C es el medio principal mediante el cual se pueden conectarse equipos auxiliares a los ordenadores personales, a pesar de que este modelo fue proyectado para resolver únicamente el problema de conexión entre módems (DCE) y ordenadores (DTE). La mayoría de dificultades con este modelo provienen de su utilización para tareas diferentes para las que fue diseñado.

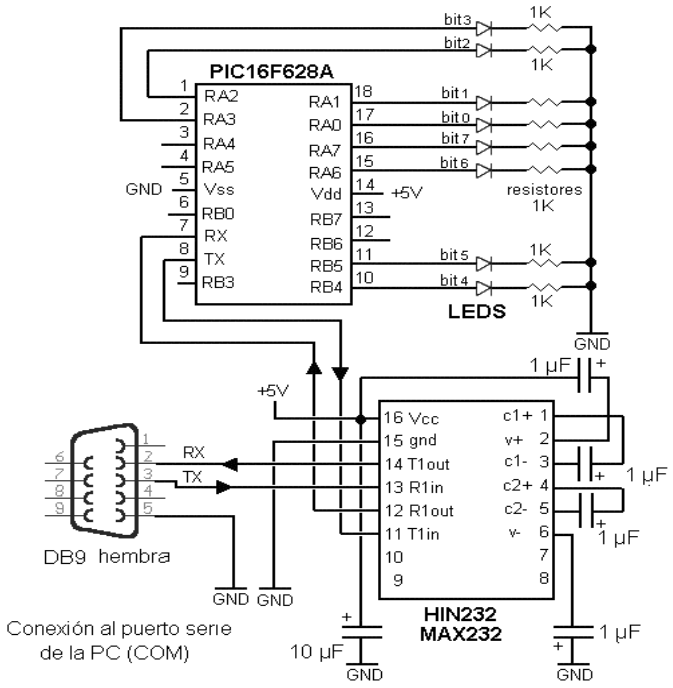


Figura 1.65 Comunicación con Norma RS 232

El documento que establecía el estándar constaba de cuatro secciones:

- *Características de la señal eléctrica:* Definición de los voltajes que representan los ceros y unos lógicos.

- *Características mecánicas de la conexión:* Establece que el DTE dispondrá de un conector macho y el DCE un conector hembra. También se especifican la asignación de números a las patillas. El tipo y las medidas del conector son establecidas por la organización internacional de estándares (ISO). Los más utilizados son los de 9 pines (DB-9) y los de 25 (DB-25).
- *Descripción funcional de los circuitos de intercambio:* En esta sección del documento se define y da nombre a las señales que se utilizarán.
- *Interfaces para configuraciones seleccionadas de sistemas de comunicación:* Son ejemplos de tipos comunes de conexión entre ordenador y módem.

Los tres circuitos principales utilizados para la comunicación son los siguientes:

- **Línea 2 (TXD):** Salida de datos del DTE.
- **Línea 3 (RXD):** Entrada de datos al DTE.
- **Línea 7 (común):** Circuito común, referencia para determinar la polaridad y voltaje de las otras líneas.

El término salida se refiere a la transferencia de datos desde un ordenador a un dispositivo externo. Recíprocamente, la transferencia de datos desde un dispositivo externo al ordenador se conoce como entrada. Estos procesos reciben el nombre genérico de entrada/salida (E/S).

Hay que considerar el sentido físico correspondiente a los conceptos de entrada y salida. La salida de datos se realiza cambiando la diferencia de potencial entre la línea 2 y la 7. Si disponemos de dos cables conectados respectivamente a las patillas 2 y 7 del conector, esta diferencia de potencial se transmitirá a largo de ellos, ya que se trata de materiales conductores. La entrada de datos corresponde al proceso inverso, generación por una fuente externa de una serie de diferencias de potencial y detección de dichas diferencias entre las patillas 3 y 7 del conector.

Los voltajes correspondientes a los niveles lógicos existentes en la conexión RS-232 se esquematizan en la figura siguiente.

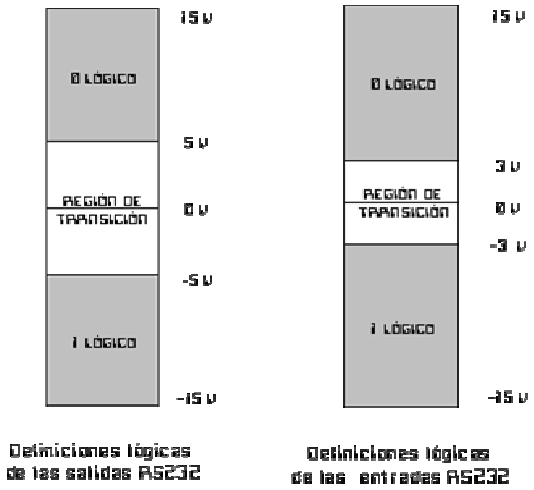


Figura 1.66 Definición de los voltajes que representan los niveles lógicos en el RS-232

Como puede observarse, la conexión RS-232 no opera con la misma fuente de alimentación de 5V de otros circuitos electrónicos integrados en el ordenador. Sus voltajes pueden oscilar entre +15 y -15 voltios. Además, los datos son transmitidos al contrario de las convenciones lógicas de uso corriente: *un voltaje positivo en la conexión representa un 0, mientras que un voltaje negativo representa un 1.*

La única diferencia entre la definición de salida y de entrada es el ancho de la región de transición, de -3 a +3 V en la entrada y de -5 a +5 V en la salida. Esta diferencia entre las definiciones de voltajes mínimos permisibles se conoce como el *margen de ruidos del circuito*. Este margen de seguridad es de gran utilidad cuando los cables deben pasar por zonas cercanas a elementos que generan interferencias eléctricas: motores, transformadores, reguladores, equipos de comunicación. Estos elementos, unidos a la longitud del cable pueden hacer disminuir la señal hasta en voltios, sin que se afecte adversamente al nivel lógico de la entrada.

Si aumentamos la velocidad de transmisión, las señales de datos se vuelven susceptibles a pérdidas de voltaje causadas por la capacidad, resistencia e inductancia del cable. Estas pérdidas son conocidas como *efectos de alta frecuencia*, y aumentan con la longitud del cable. El ancho de la zona de transición (-3V a +3V en la entrada) determina el margen de ruidos, que limita directamente la velocidad máxima a la que se pueden transmitir datos sin

degradación. Entre dos equipos RS-232 esta velocidad es de 19200 bits por segundo, para longitudes de cable inferiores a 15 metros, pero disminuyendo la velocidad pueden utilizarse longitudes mayores de cable.

El acoplamiento por software existe cuando un dispositivo controla al otro por medio del contenido de los datos. Por ejemplo, si debemos mandar información a otro elemento podemos incluir caracteres de control al inicio y final de la comunicación, para indicar la longitud del mensaje y un chequeo de todo el conjunto de datos enviados.

Teniendo en cuenta que para cada señal transmitida puede ser necesario un mínimo de un acoplamiento, y que los dispositivos pueden transmitir y recibir, podemos observar que serán necesarios más de los tres circuitos básicos antes mencionados (común, entrada y salida).

Los nombres dados en el modelo oficial RS-232-C para las señales de datos y acoplamiento, así como su asignación a las diferentes patillas (pines) del conector, aparecen en la tabla 1.4.

PIN DB 25	Pin DB 9	Nombre	Función
2	3	TXD	TRANSMISIÓN DE DATOS (SALIDA)
3	2	RXD	RECEPCIÓN DE DATOS (ENTRADA)
4	7	RTS	PETICIÓN DE ENVÍO (SALIDA)
5	8	CTS	DISPUESTO PARA ENVIAR (ENTRADA)
6	6	DSR	DATOS PREPARADOS (ENTRADA)
7	5	COMÚN	COMÚN (REFERENCIA)
8	1	DCD	DETECCIÓN DE PORTADORA DE DATOS (ENTRADA)
20	4	DTR	TERMINAL DE DATOS LISTO (SALIDA)
22	9	RI	INDICADOR DE LLAMADA (ENTRADA)

Tabla 1.4 Pines para DB9 y DB25

1.5.1.2 Definición de los circuitos más comunes.

- **1 CG*, Chassis Ground*, Tierra del chasis.**

Este circuito es un mecanismo para asegurar que los chasis de los dos dispositivos estén al mismo potencial, y para impedir una descarga eléctrica al operador. Es la tierra de seguridad del sistema.

- **2 TD*, Transmit Data*, Datos de transmisión.**

Este circuito es la trayectoria por medio del cual los datos se envían desde el DTE al DCE. Este circuito debe estar presente si los datos deben viajar en esa dirección en cualquier momento.

- **3 RD*, Receive Data*, Datos de recepción.**

Esta línea es el recorrido por medio del cual los datos se envían desde el DCE al DTE. Esta línea debe estar presente si los datos deben viajar en esa dirección en un momento dado.

- **4 RTS*, Request To Send*, Petición de envío.**

Este circuito es la señal que indica que el DTE desea enviar datos al DCE, (ninguna otra línea está disponible para la dirección opuesta, de aquí en adelante el DTE debe estar siempre listo para aceptar datos).

- **5 CTS*, Clear To Send*, Limpieza de envío.**

Esta línea es la señal que indica que el DCE está preparado para aceptar datos desde el DTE. En operación normal, la línea CTS estará en la condición OFF. Cuando el DTE confirma RTS, el DCE hará lo que sea necesario para permitir que los datos sean enviados.

- **6 DSR*, Data Set Ready*, Datos preparados.**

Esta línea es la señal que informa al DTE que el DCE está vivo y bien. Es normalmente puesta a ON por el DCE al encenderse este. Un DTE típico deberá tener un DSR entrante a fin de desempeñarse normalmente. Si no existe esta línea, debe ser traída desde el DCE, o provista por un corto localmente al DTE. Sobre el DCE, esta señal está casi siempre presente, y

puede volverse atrás (a DTR o RTS) para satisfacer las señales requeridas cuya función no está implementada.

- **7 SG, *Signal Ground*, Señal de tierra.**

Este circuito es de tierra al que todos los otros voltajes están referenciados. Debe estar presente en cualquier interfaz RS-232.

- **8 DCD, *Data Carrier Detect*, Portadora de datos detectada.**

Ésta es la señal por medio del cual el DCE informa al DTE que tiene una portadora entrante. Puede ser usado por el DTE para determinar si el canal está desocupado, y que el DTE pueda pedir un RTS.

- **15 TC, *Transmit Clock*, Reloj de transmisión.**

Este pin provee el reloj para la sección de transmisor de un DTE sincrónico. Debe estar presente sobre las interfaces sincrónicas. Puede o no correr al mismo rango que corre en el receptor.

Para realizar una transmisión solo seria necesario:

- 1 para enviar datos (TXD).
- 1 para recibir datos (RXD).
- 1 común a todos los circuitos.
- 4 señales acoplamiento para poder enviar datos (CTS, DSR, DCD, RI).
- 2 señales de acoplamiento para poder recibir datos (RTS, DTR).

Algunas de las señales (DCD, RI) provienen de características necesarias para poder detectar el estado de un módem, pero no suelen ser necesarias para aplicaciones normales.

Por ejemplo, para conectar dos ordenadores personales (dispositivos DTE) con señales de acoplamiento, sería necesario efectuar las conexiones descritas en la figura 1.76. En estos esquemas, la dirección de las flechas indica realmente el sentido en que se mueve la información, es decir, el emisor y el receptor de la señal.

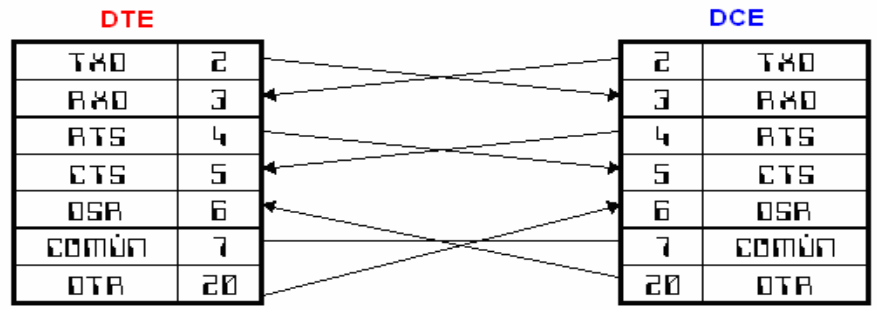


Figura 1.67. Conexión estándar entre dos equipos RS-232-C DTE.

En el caso de no desear utilizar estas señales de acoplamiento, puede optarse por proporcionarlas por un medio físico, pues algunos programas de comunicación pueden requerir su presencia. Un posible esquema para esta conexión, puede ser el indicado en la figura 1.68. Se trata de un esquema más sencillo, pero puede funcionar en una gran parte de equipos, siempre que no se desee trabajar al límite de la capacidad de los dispositivos.

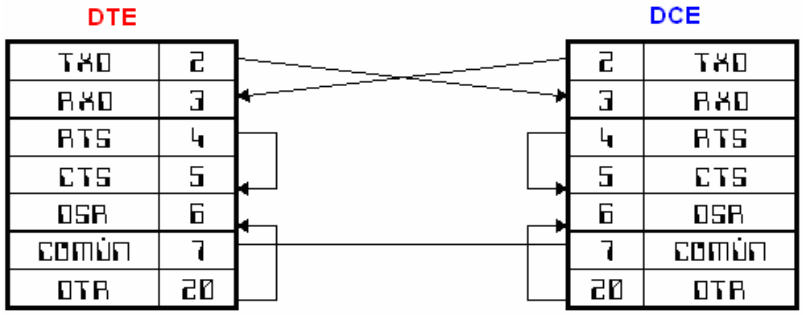


Figura 1.68. Conexión entre dos equipos RS-232-C DTE sin utilizar acoplamientos.

Con lo expuesto hasta el momento es posible realizar la conexión física entre la mayor parte de dispositivos RS-232-C, aunque en ocasiones existen algunas excepciones y particularidades que impiden la correcta conexión.

Es conveniente remarcar que la comunicación RS-232-C fue originalmente diseñada para establecer una comunicación punto a punto, es decir, entre dos únicos elementos. Al ser solamente dos elementos era posible efectuar los acoplamientos necesarios. En el momento que eliminamos los acoplamientos por hardware, estamos abriendo la posibilidad a conectar varios equipos simultáneamente al mismo canal RS-232-C.

1.5.2 EL C.I. MAX 232

El circuito integrado MAX232 cambia los niveles TTL a los del estándar RS-232 cuando se hace una transmisión, y cambia los niveles RS-232 a TTL cuando se tiene una recepción. Sólo es necesario este chip y 4 condensadores electrolíticos de 22 microfaradios. El esquema es el siguiente, ver figura 1.69.

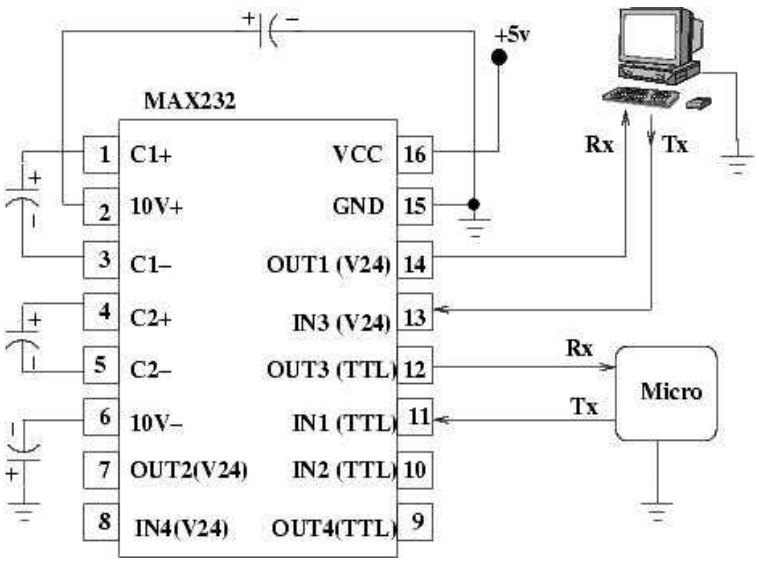
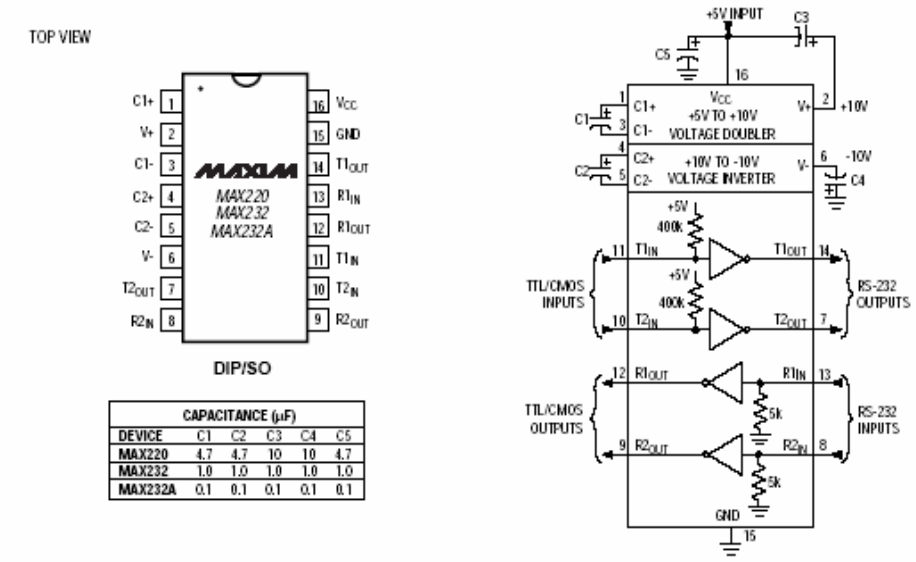


Figura 1.69 Comunicación entre el PIC y un PC por medio del CI MAX232.

El MAX232 soluciona la conexión necesaria para lograr comunicación entre el puerto serie de una PC y cualquier otro circuito con funcionamiento en base a señales de nivel TTL/CMOS. Cambia los niveles TTL a los del estándar RS-232 cuando se hace una transmisión, y cambia los niveles RS-232 a TTL cuando se tiene una recepción, es decir, es un circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa. Lo interesante es que sólo necesita una alimentación de 5V, ya que genera, internamente, algunas tensiones que son necesarias para el estándar RS232. Otros integrados que manejan las líneas RS232 requieren dos voltajes, +12V y -12V.

Dispone internamente de 4 conversores de niveles TTL al bus standard RS232 y viceversa, para comunicación serie como los usados en los ordenadores y que ahora están en desuso, el Com1 y Com2. El circuito integrado lleva internamente 2 conversores de nivel TTL a RS232 y otros 2 de RS232 a TTL con lo que en total podremos manejar 4 señales del puerto serie de la PC.

Por lo general, las más usadas son; TX, RX, RTS, CTS, estas dos ultimas son las usadas para el protocolo handshaking. Ver mas detalles en (ANEXO 3).



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figura 1.70 Distribución de pines y estructura interna de CI MAX232

1.5.3 LIMITACIONES DE LA RS-232 C

La RS-232 C tiene una limitación de distancia máxima de 15 metros. Si bien no es una desventaja considerable cuando los equipos a conectar se encuentran cerca, sí es un inconveniente cuando la RS-232 se utiliza para conectar directamente terminales o impresoras que puedan estar lejanas.

Cuando una señal cambia de una condición a otra, la especificación limita el tiempo que puede permanecer en la región de transición. Este requerimiento determina el máximo de capacidad distribuida admisible en el cable, porque la capacidad limita el tiempo de transición de la señal. La norma RS-232 especifica que la capacidad en la línea no debe superar los 2.500 picofaradios. Los cables que se suelen utilizar tienen una capacidad de 120 a 150 picofaradios por metro de longitud, por lo que la RS-232 tiene como límite de 15 m de distancia.

Una segunda limitación de la RS-232 es su método de toma de tierra o retorno común. Este método, llamado transmisión no balanceada, funciona bien la

mayor parte del tiempo. Sin embargo, si hay diferencia de potencial entre los dos extremos del cable (lo cual es bastante probable en recorridos largos), se reduce la región de transición entre marca y espacio. Cuando ocurre esto, existe la posibilidad que no se interpreten bien los distintos estados de la señal.

Otra dificultad es su máximo de 20 KB/s para la velocidad de transmisión. Si bien en el momento de aparición del estándar era suficiente, en la actualidad, comparando con las velocidades alcanzadas por las redes de área local, 10 y 100 Mbps y las exigencias de ancho de banda que las aplicaciones requieren, la RS-232-C en algunos casos está disminuyendo su aplicación.

A partir de la RS-232 se desarrollaron nuevas interfaces que pretenden transmitir a mayor velocidad alcanzando mayor distancia. Estas nuevas interfaces como la RS-422 y la RS-485 eliminan algunas de las restricciones de la RS-232, por ejemplo, la de poseer un retorno común para todas las señales.

1.6 CONEXIÓN DE UN MICROCONTROLADOR AL PUERTO SERIE DEL PC

Para conectar el PC a un microcontrolador por el puerto serie se utilizan las señales TX, RX y GND. El PC utiliza la norma RS232, por lo que los niveles de tensión de los pines están comprendidos entre +15 y -15 voltios. Los microcontroladores normalmente trabajan con niveles TTL (0-5V). Es necesario por tanto intercalar un circuito que adapte los niveles:

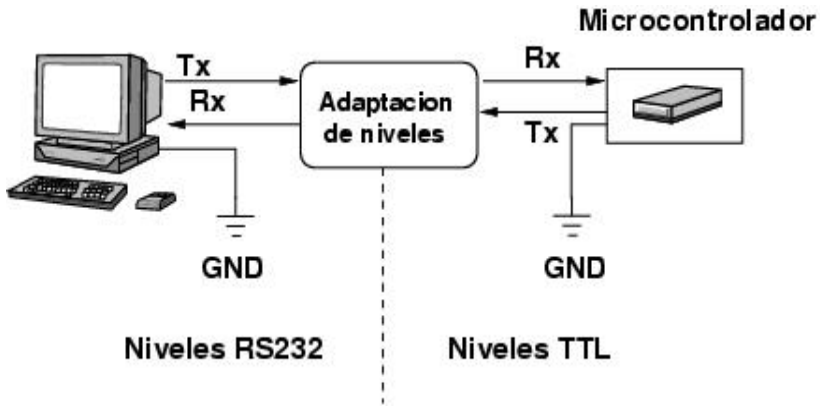


Figura 1.71 Diagrama general de comunicación PC y un Microcontrolador PIC

Para poder acoplar los niveles dichos se utiliza el circuito integrado **MAX232**.

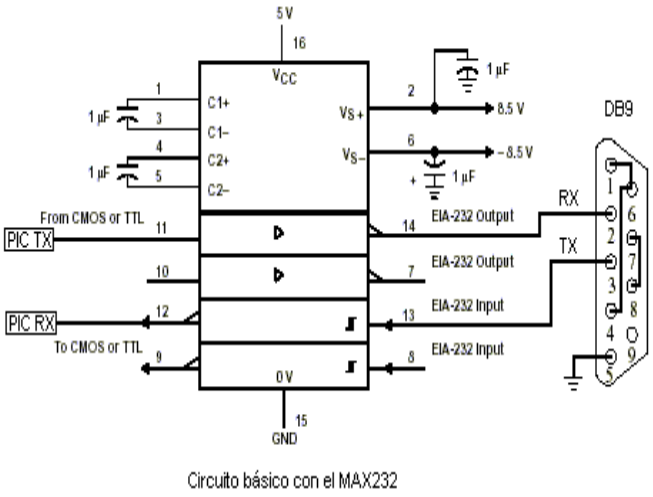


Figura 1.72 Diagrama de conexión del CI MAX232 y terminal DB9

Circuito de conexión básico para la comunicación PIC a PC por medio de CI MAX232, este circuito es colocado en la parte correspondiente a adaptación de niveles, ver figura 1.71.

1.6.1 PRUEBA DE CONEXIÓN SERIE ENTRE UN PIC Y UNA PC

Presento aquí la prueba de laboratorio de un circuito realizado específicamente para experimentar, de manera básica, con la comunicación entre un microcontrolador PIC 16F628A y el puerto serie de una PC (estándar RS-232). El circuito tiene, además del microcontrolador, un integrado convertidor de niveles RS-232 a TTL que es reemplazo del conocido MAX232.

El microcontrolador se comunica utilizando su puerto serie. Es programado este puerto a una velocidad de 9600 baudios, un formato de dato de 8 bits, sin paridad, un bit de parada, y sin ningún control de flujo. El programa en el microcontrolador se inicia enviando un mensaje a la PC. Luego la rutina principal de este pequeño programa espera a recibir un carácter, lo devuelve como eco hacia la PC, y luego lo exhibe a través de sus puertos, donde tenemos conectados indicadores de recepción. Esto nos permite comprobar la recepción de los caracteres ASCII desde la PC, cuyo código veremos sobre los indicadores.

1.6.1.1 Uso del programa "Hyper Terminal" para la comunicación desde la PC

Para establecer la comunicación serie desde la PC, se debe utilizar el programa Hyper Terminal que viene con el sistema operativo Windows.

Aquí se puede ver una explicación de Microsoft sobre cómo utilizar el programa Hyper Terminal.

En la configuración de la conexión, usted debe indicar que se va a conectar "Directo a COM", indicando cuál es el puerto COM de la PC que va a utilizar en esta comunicación (generalmente es el COM1 o COM2).

Esto se define en la sección "Configuración de puerto".

- Bits por segundo = 9600
- Bits de datos = 8
- Paridad = Ninguna
- Bits de parada = 1
- Control de flujo = Ninguno

Las ventanas en la parte inferior muestran las distintas configuraciones que se las realizan para poder acceder a la ventana de comunicación Hyper terminal.

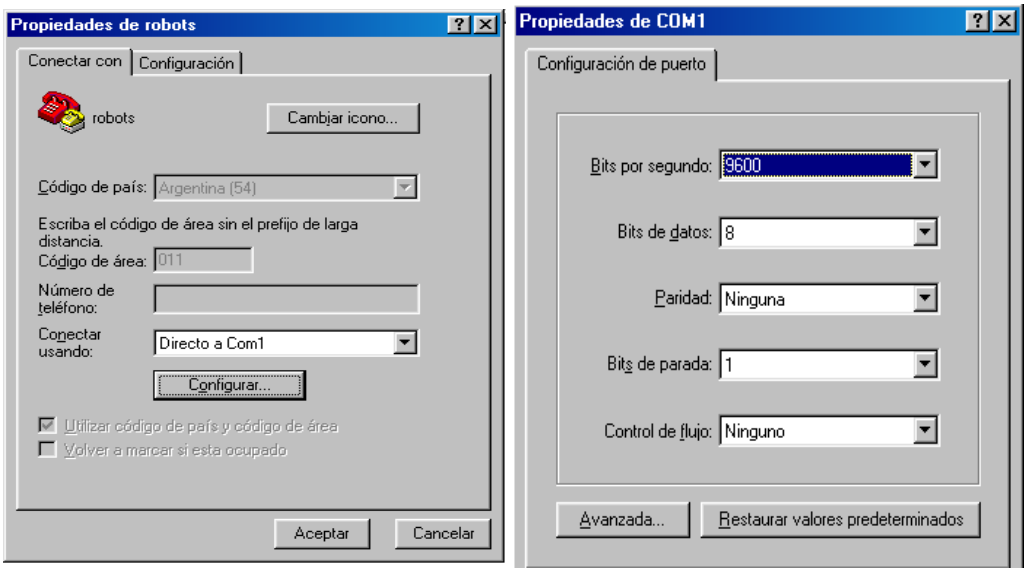


Figura 1.73 Ventanas de configuración para la comunicación serial.

Las ventanas proporcionadas por Windows son para configurar el puerto COM a utilizarse los PCs tienen un puerto serie definido que es el puerto COM1.

En la ventana de la derecha podemos configurar las opciones de transmisión de bits en este caso son 9600 baudios, bits de paridad en este caso ninguno, bits de datos (8) y control de flujo esto se lo realiza para poder visualizar los datos a las velocidades adecuadas, en el caso que no este bien configurado tendremos como resultado símbolos inentendibles.

1.6.2 EL CONECTOR DB9 DEL PC

En los PCs hay conectores DB9 macho, de 9 pines, por el que se conectan los dispositivos al puerto serie. Los conectores hembra que se enchufan tienen una colocación de pines diferente, de manera que se conectan el pin 1 del macho con el pin 1 del hembra, el pin2 con el 2, etc.

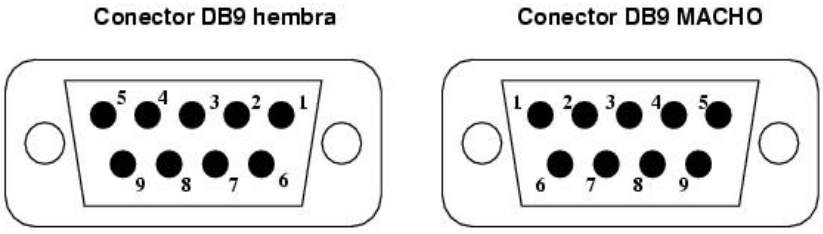


Figura 1.74 Conectores macho y hembra de 9 pines (DB9)

La información asociada a cada uno de los pines es la siguiente:

Número de pin	Señal
1	**DCD** (Data Carrier Detect)
2	**RX**
3	**TX**
4	**DTR** (Data Terminal Ready)
5	**GND**
6	**DSR** (Data Sheet Ready)
7	**RTS** (Request To Send)
8	**CTS** (Clear To Send)
9	**RI** (Ring Indicator)

1.6.3 CABLE DE CONEXIÓN

Para realizar la conexión entre el PC y nuestro circuito podemos usar diferentes alternativas. Una manera es utilizar un **cable serie macho-hembra no cruzado**, y en el circuito un conector hembra DB9 para circuito impreso.

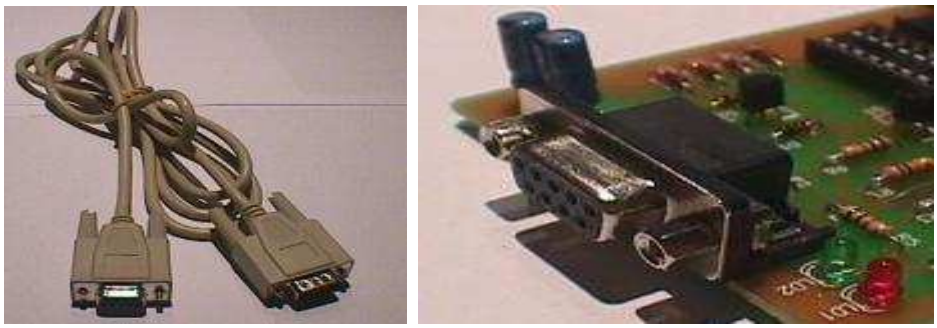


Figura 1.75 Cable de comunicación con terminales DB9 y terminal DB9 para placa

Cuando conectamos un micro al PC normalmente sólo usamos los pines TX, RX y GND, sin embargo en este tipo de cables se llevan los 9 pines. Por ello puede resultar útil el utilizar otro tipo de cable.

También se puede fabricar un cable serie utilizando cable plano de bus, conectado un conector DB9 hembra para bus:

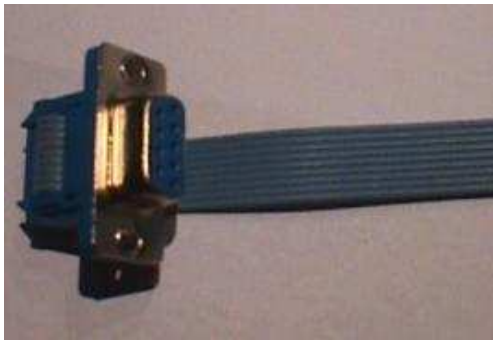


Figura 1.76 Bus de datos conectado a un terminal DB9

1.7 NORMA RS-485[8]

El modelo RS-485 permite características no previstas en el estándar RS-232, facilitando el trabajo industrial, al permitir velocidades de transmisión cercanas a 1 megabit por segundo, así como longitudes de la línea de hasta 1200 metros. Además permite el alargamiento de la red en otros 1200 metros al insertar un repetidor RS-485 en la línea.

También tiene otra característica muy importante en ambientes industriales, puede soportar hasta 32 nodos (equipos emisores/receptores) conectados por cada segmento de red. Estos distintivos lo hacen muy adecuado para el trabajo que fue diseñado, aplicaciones industriales.

Dentro del estándar RS-485 existen diferentes variantes, una de las cuales es la conocida como 4D-RS-485. En este caso se mantienen por separado los pares de cables de recepción y conexión. Las únicas señales que son necesarias para transmitir se muestran en la tabla A.

Nombre	Función
TXD (+)	TRANSMISIÓN DE DATOS (SALIDA +)
TXD (-)	TRANSMISIÓN DE DATOS (SALIDA -)
RXD (+)	RECEPCIÓN DE DATOS (ENTRADA +)
RXD (-)	RECEPCIÓN DE DATOS (ENTRADA -)
TIERRA	TIERRA

Tabla A. Señales utilizadas en la conexión 4D-RS-485

Como se puede visualizar en el grafico 1.77 tenemos varias conexiones en la misma línea de transmisión de con un máximo de 32 nodos.

[8] elp@i-micro.com

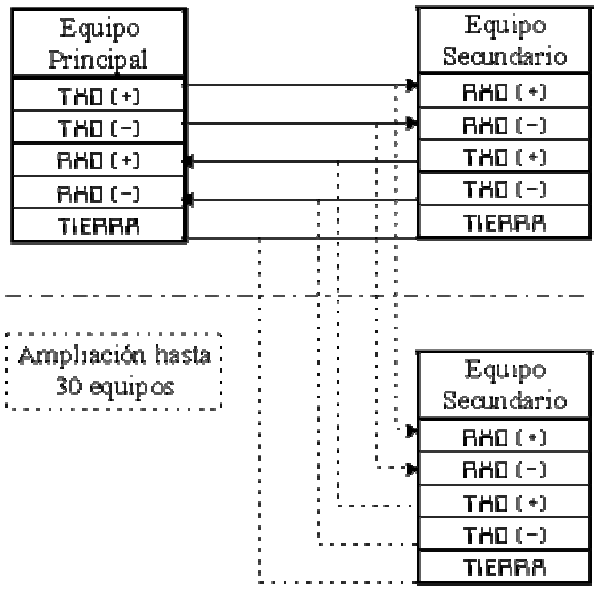


Figura 1.77 Conexión entre equipos 4D-RS-485.

El siguiente paso en simplificación es el estándar 2D-RS-485, el más comúnmente conocido por estándar RS-485. En este caso se elimina uno de los pares transmisión/recepción. Se utiliza una sola línea de transmisión balanceada bidireccional. Las características físicas de la línea se mantienen (longitud y velocidades de transmisión admisibles). La diferencia con el anterior es que los dispositivos deben conmutar entre modo receptor y modo transmisor, para evitar que varios dispositivos emitan simultáneamente.

Las señales necesarias en este caso son las mostradas en la tabla B.

Nombre	Función
TXD / RXD (+)	TRANSMISIÓN DE DATOS (SALIDA +)
TXD / RXD(-)	TRANSMISIÓN DE DATOS (SALIDA -)
TIERRA	TIERRA

Tabla B. Señales utilizadas en la conexión 2D-RS-485

El esquema de interconexión entre equipos debería establecerse como muestra la figura 1.78.

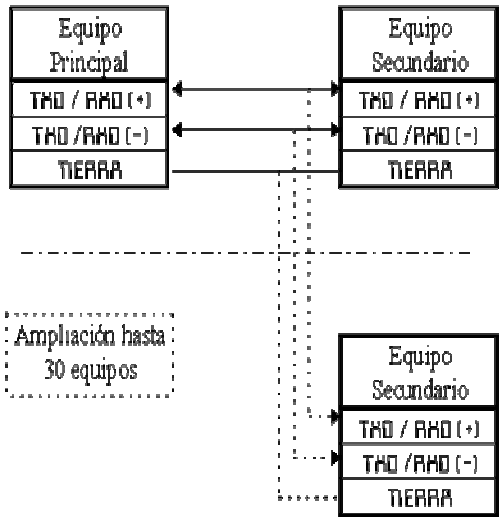


Fig.1.78 Conexión entre equipos 2D-RS-485.

En ambos modos RS-485 la tierra es opcional, debería utilizarse en conexiones donde puedan existir interferencias.

1.7.1 VENTAJAS DE RS-485

Esta interfase tiene muchas ventajas con respecto a RS-232, entre las cuales se mencionan:

1.7.1.1 Bajo costo

Los Circuitos Integrados para trasmitir y recibir son baratos y solo requieren una fuente de +5V para poder generar una diferencia mínima de 1.5V entre las salidas diferenciales. En contraste con RS-232 que en algunos casos requiere de fuentes dobles para alimentar algunos circuitos integrados.

1.7.1.2 Capacidad de interconexión

RS-485 es una interfase multi-enlace con la capacidad de poder tener múltiples transmisores y receptores. Con una alta impedancia receptora, los enlaces con RS-485 pueden llegar a tener a lo máximo hasta 256 nodos.

1.7.1.3 Longitud de Enlace

En un enlace RS-485 puede tener hasta 1200metros de longitud, comparado con RS-232 que tiene unos limites típicos de 15 a 30metros.

1.7.1.4 Rapidez

La razón de bits puede ser tan alta como 10 Mega bits por segundo (10Mbps).

1.7.2 BALANCEO Y DESBALANCEO DE LÍNEAS

La razón por la que RS-485 puede transmitir a largas distancias, es porque utiliza el balanceo de líneas. Cada señal tiene dedicados un par de cables, sobre uno de ellos se encontrará un voltaje y en el otro se estará su complemento, de esta forma, el receptor responde a la diferencia entre voltajes.

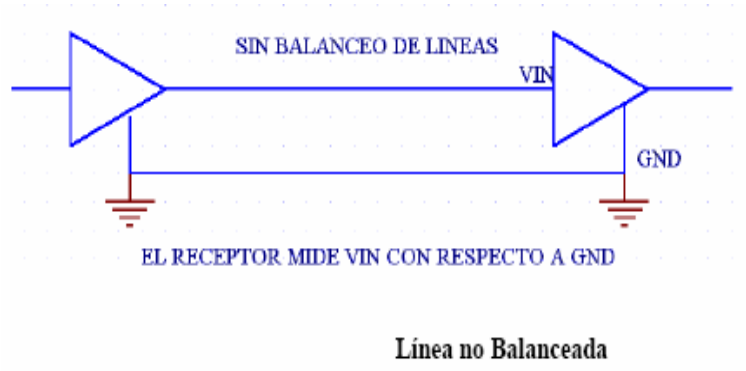


Figura 1.79 Imagen de línea no balanceada con Norma RS485

La ventaja de las líneas balanceadas es su inmunidad al ruido

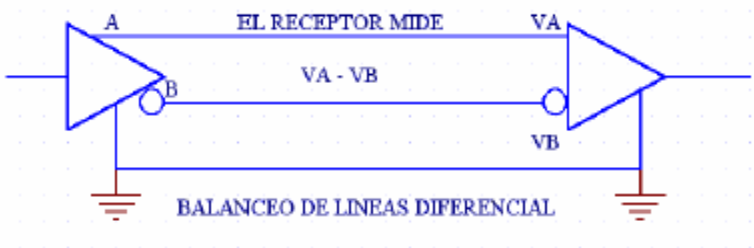


Figura 1.80 Imagen de línea balanceada con Norma RS485

En cuanto a las líneas balanceadas (Figura 1.79) la **TIA/EIA-485** designa a estas dos líneas como **A y B.** En el controlador de transmisión TX, una entrada alta TTL causa que la línea A sea más positiva (+) que la línea B, mientras que un bajo en lógica TTL causa que la línea B sea más positiva (+) que la línea A.. Por otra parte en el controlador de recepción RX, si la entrada A es más positiva (+) que la entrada B, la salida lógica TTL será “1” y si la entrada B es más positiva (+) que la entrada A, la salida lógica TTL será un “0”.

1.7.3 REQUERIMIENTOS DE VOLTAJE

Las interfases típicas RS-485 utilizan una fuente de +5V, pero los niveles lógicos de los transmisores y receptores no operan a niveles estándares de +5V o voltajes lógicos CMOS. Para una salida válida, la diferencia entre las salidas A y B debe ser al menos +1.5V. Si la interfase está perfectamente balanceada, las salidas estarán desfasadas igualmente a un medio de la fuente de Voltaje.

En el receptor RS-485, la diferencia de voltaje entre las entradas A y B necesita ser 0.2V. Si A es al menos 0.2V más positiva que B, el receptor ve un 1 lógico y si B es al menos 0.2V más positivo que A, el receptor ve un 0 lógico. Si la diferencia entre A y B es menor a 0.2V, el nivel lógico es indefinido. Si esto ocurre habría un error en la transmisión y recepción de la información.

La diferencia entre los requerimientos del Transmisor y el Receptor pueden tener un margen de ruido de 1.3V. La señal diferencial puede atenuarse o tener picos de largo como de 1.3V, y aun así el receptor vera el nivel lógico correcto. El margen de ruido es menor que el de un enlace RS-232, no hay que olvidar que RS-485 maneja señales diferenciales y que cancela la mayoría del ruido a través de su enlace.

El total de corriente utilizada por un enlace RS-485 puede variar debido a las impedancias de los componentes, incluyendo los Transmisores, Receptores, cables y la terminación de los componentes. Una baja impedancia a la salida del Transmisor y una baja impedancia en los cables, facilita los cambios de

nivel y asegura que el receptor vea la señal, no importa cuan larga sea la línea de transmisión. Una alta impedancia en el receptor decrementa la corriente en el enlace e incrementa la vida de la fuentes de voltaje.

La terminación de los componentes, cuando se utiliza tiene un gran efecto sobre la corriente en el enlace. Muchos enlaces con RS-485 tienen una resistencia de 120Ω a través de las líneas A y B en cada extremo de la línea. Por lo tanto cada enlace tiene dos terminales.

1.7.4 CONEXIONES DE COMUNICACION RS-485

1.7.4.1 Comunicación en modo Half Duplex

El término Half Duplex en un sistema de comunicación se refiere, a que solamente en un tiempo determinado, el sistema puede transmitir o recibir información, sin embargo no lo puede hacer al mismo tiempo. En muchos enlaces del tipo RS-485 se comparte el BUS.

Como se puede observar existe una línea de control, la cual habilita a los controladores en un solo sentido. Por lo tanto, se debe tener cuidado de no transmitir y recibir al mismo tiempo, ya que se podría crear una superposición de información. La siguiente figura muestra el esquema de una comunicación RS-485 en Modo Half Duplex.

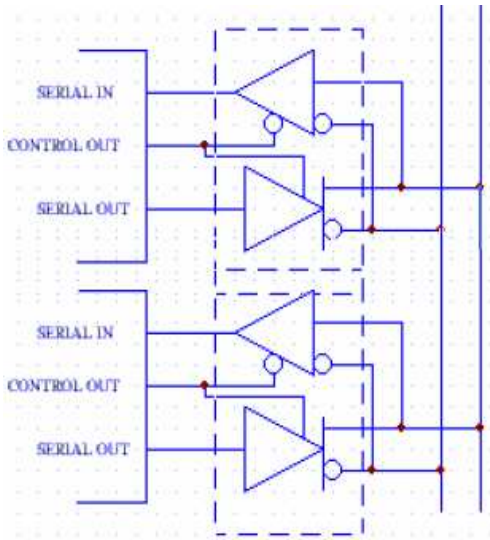


Figura 1.81 Esquema de una comunicación Half Duplex

1.7.4.2 Comunicación en modo Full Duplex.

Para este trabajo se utilizará la comunicación RS-485 en modo Full Duplex, ya que al contar con varios microcontroladores esclavos, se necesita que cada uno de ellos este reportando los datos obtenidos de cada proceso, sin embargo, como no se sabe cuando se necesitará dicha información, se requieren de dos canales, uno independiente del otro, para poder transmitir y recibir al mismo tiempo la información.

El término Full Duplex se refiere a que un sistema puede transmitir y recibir información simultáneamente. Bajo este concepto la interfase RS-485 está diseñada para sistemas multipunto, esto significa que los enlaces pueden llegar a tener más de un transmisor y receptor, ya que cada dirección o sea Transmisión y Recepción tienen su propia ruta. La siguiente figura muestra este tipo de conexión.

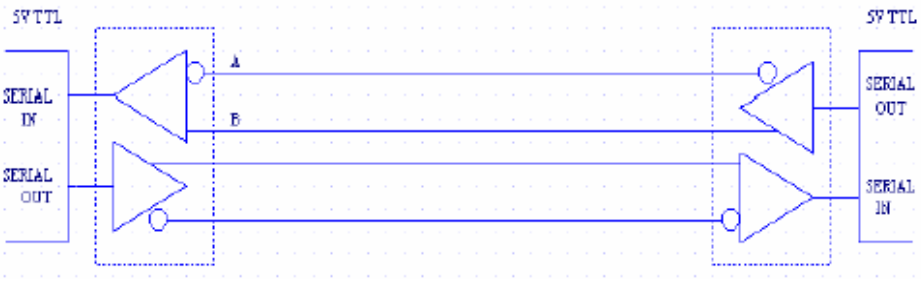


Figura 1.82 Conexión de modo Full Dúplex con RS-485

En la siguiente figura 1.83 se muestra como es posible utilizar la comunicación Full Duplex con múltiples nodos transmisores y receptores. En este arreglo del tipo maestro / esclavo, se pondrá como ejemplo que el nodo 1 es el maestro, por lo tanto tiene el control de la red y el asigna el permiso para transmitir.

Un par de cables están conectados del nodo trasmisor Maestro a todos los controladores receptores esclavos. En el otro sentido, un par de cables conectan a todos los esclavos al receptor del Maestro.

Todos los esclavos deben leer lo que el maestro envía, pero solo uno va a poder responder y lo hace a través de los cables opuestos.

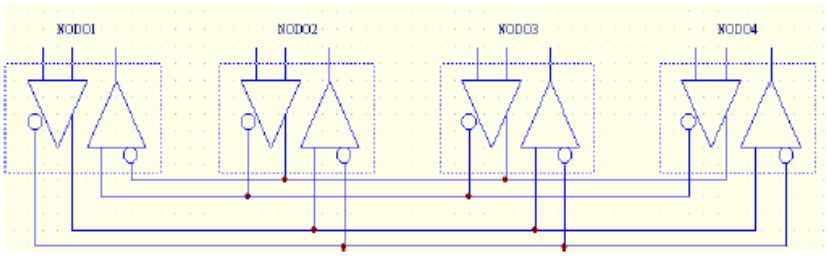


Figura 1.83 Topología Maestro / Esclavo en conexión RS-485

1.7.5 EL CIRCUITO SN75176

Para lograr la comunicación con el ordenador se elabora una interfase del tipo RS-485, para su elaboración, se utilizan dos circuitos integrados con la serie SN75176 de Texas Instruments, uno es para la recepción de datos y otro para la transmisión.

Estos dispositivos se encargan de hacer la conversión entre los niveles TTL del microcontrolador y las señales del tipo diferencial que se utilizan en el bus RS-485. Vale la pena decir que en el controlador de transmisión se agregó una línea de habilitación, esto se debe a que todas las salidas de los microcontroladores están conectadas a la línea de recepción del ordenador, así cada uno está siempre deshabilitado para enviar datos y solo se habilitará en el momento en que deba hacer una transmisión, evitando así conflictos o choques de información en la línea o bus de datos, a continuación la siguiente figura hace una breve descripción de este circuito integrado. Estructura interna y descripciones técnicas en el (ANEXO 4).

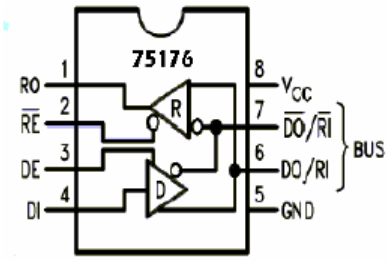


Figura 1.84 Distribución de pines y estructura interna de C.I. SN 75176.

- En las terminales Vcc y GND se encuentra la alimentación del circuito, que este caso es de +5V.

- La terminal R0 y DI recibe un nivel lógico TTL si y solo si la línea RE se habilita y como se puede observar es con un ‘0’ lógico
- Las terminales D0 y $\overline{D0}$ reciben también el nombre de A y B y son sobre estas líneas las que forman el Bus de Transmisión y Recepción.

Como se puede observar, cada chip consta de un transmisor y un receptor, si las terminales **RE (Pin 2) y DE (Pin 3)** se unen entre si con un solo Bit se puede controlar el flujo de la información.

1.8 CONVERSIÓN RS-232-C / RS-485

En el caso que de desee utilizar un ordenador personal como elemento principal de la red, es muy común encontrarse con el problema de la conversión entre los dos estándares. Si estamos en un ambiente de laboratorio puede ser suficiente con utilizar una placa de comunicación que se pueda configurar como un puerto estándar de PC. En el caso de encontrarnos en la industria, estos elementos suelen ser poco resistentes a interferencias o sobrecargas provocadas por la proximidad de elementos eléctricos.

La solución para estos casos es la utilización de un conversor optoaislado (aislado de corrientes externas mediante un sistema basado en diodos), que transforma la señal generada por el RS-232-C a RS-485, aislando totalmente las dos redes y con la posibilidad de soportar sobrecargas inducidas superiores a los 500 voltios.

Las señales necesarias para conectar el ordenador al conversor y el conversor al resto de elementos de la red suelen ser las especificadas en la figura 1.85.

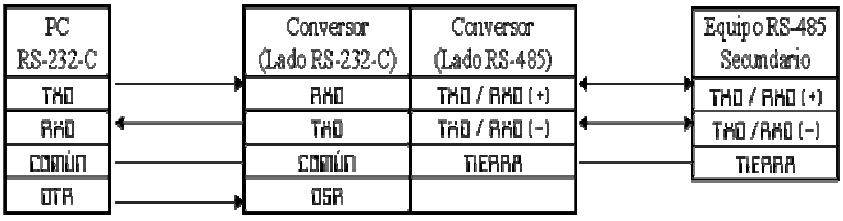


Figura 1.85 Conexión entre un PC y un dispositivo RS-485 utilizando un conversor.

La señal DTR/DSR entre el ordenador y el conversor es utilizada por el elemento principal (en este caso el PC) para indicar al conversor que cambie de modo receptor a emisor. Como se ha comentado anteriormente, en el modelo 2D-RS-485, los equipos deben estar siempre en modo receptor, excepto en el momento que deban transmitir, que pasan a modo de envío. La forma que tiene el equipo principal de indicar este cambio de recepción a envío es mediante esa señal de acoplamiento DTR/DSR. En otros conversores se utiliza otra señal de acoplamiento, como RTS/CTS, aunque la función es la misma.

1.9 ACCESO AL PUERTO SERIAL ATRAVÉS DE VBASIC.

Para poder acceder al puerto serial y así poder enviar datos, utilizamos una aplicación creada en Visual Basic, se hace uso del control **MS COMM**, el cual trae incorporadas todas las funciones para configurar el puerto. Es gracias a este control que el manejo del puerto serial se facilita enormemente. Las propiedades más importantes de este control son las siguientes:

- **ComPort:** Activa y regresa el número del puerto serial(Comm1, Comm2)
- **PortOpen:** Activa y regresa el acceso al puerto.
- **Input:** Regresa los caracteres del buffer receptor.
- **Output:** Escribe una cadena sobre el buffer Transmisor.
- **Settings:** Activa y regresa la razón de Baudios, paridad, número de bits, bits de paro.

Para poder tener acceso a cualquier propiedad del puerto serial se utiliza la siguiente sintaxis:

Nombre del Control. Propiedad = Valor

En este caso el objeto es MS Comm1, por lo tanto si quisiera abrir el puerto COM1, la instrucción sería:

MS Comm1.PortOpen = True

Sin embargo, para poder utilizar el puerto serial, primeramente, se debe colocar el control MS Comm1 en la forma y hacer clic con el botón derecho del mouse, para que puedan aparecer sus propiedades, tal y como lo muestra la siguiente figura 1.86.

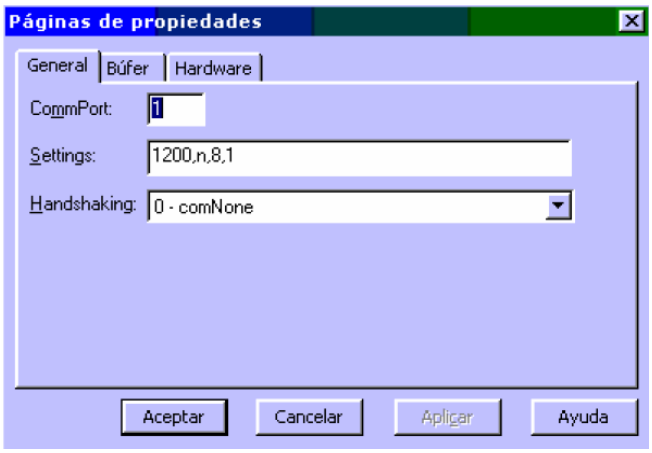


Figura 1.86 Ventana de configuración de propiedades de comunicación serial con VB.

Como la computadora sobre la cual se está trabajando solamente tiene un puerto serial y este es el comm1, en la propiedad **CommPort** debe tener el número 1. Para este ejemplo, la propiedad **Settings** se debe configurar con la siguiente cadena: **1200, n, 8, 1**, y como no se va a realizar ningún control sobre el flujo de la información la propiedad **handshaking** debe ser igual a 0.

El objeto MS Comm1 responde al siguiente evento **On Comm,** el cual genera una interrupción, indicando cuando hay comunicación o si algún error ha ocurrido en la transferencia de la información.

Para poder enviar una cadena de caracteres a través del puerto serial, lo único que se tiene que hacer es utilizar la propiedad output del objeto MS Comm:
Ejemplo:

MS Comm1.Output = “Esto es una prueba”

Como se observa, una vez configurado el puerto serial, con esta instrucción se envía a través del puerto la cadena de caracteres:

“Esto es una prueba”

1.9.1 PASOS PARA PODER ENVIAR DATOS A TRAVÉS DEL PUERTO SERIAL UTILIZANDO VBASIC

• Insertar el control **MS Comm** sobre la forma:
• Establecer las siguientes propiedades:

- **ComPort:**
- **Settings:**
- **Handshaking:**

• Abrir el puerto, si este ya está abierto por otra aplicación, entonces se debe cerrar esa aplicación, para después volverlo a abrir el puerto con una aplicación en Visual Basic, esto se hace utilizando la siguiente instrucción:

MSComm1.Portopen = true

• Definir el tamaño del buffer receptor, esto se hace con la propiedad

InputLen
MSComm1.InputLen = 1024

• Enviar los datos que se desean
• Cuando la aplicación se termine se debe cerrar el puerto.

CAPITULO 2

CONSTRUCCIÓN DEL MÓDULO DE CONTROL ELÉCTRICO.

2.1 CONSTRUCCIÓN Y ENSAMBLAJE DEL MÓDULO PARA EL CONTROL DE LA RED ELÉCTRICA

2.1.1 DIAGRAMA DE BLOQUES DEL CIRCUITO DE CONTROL

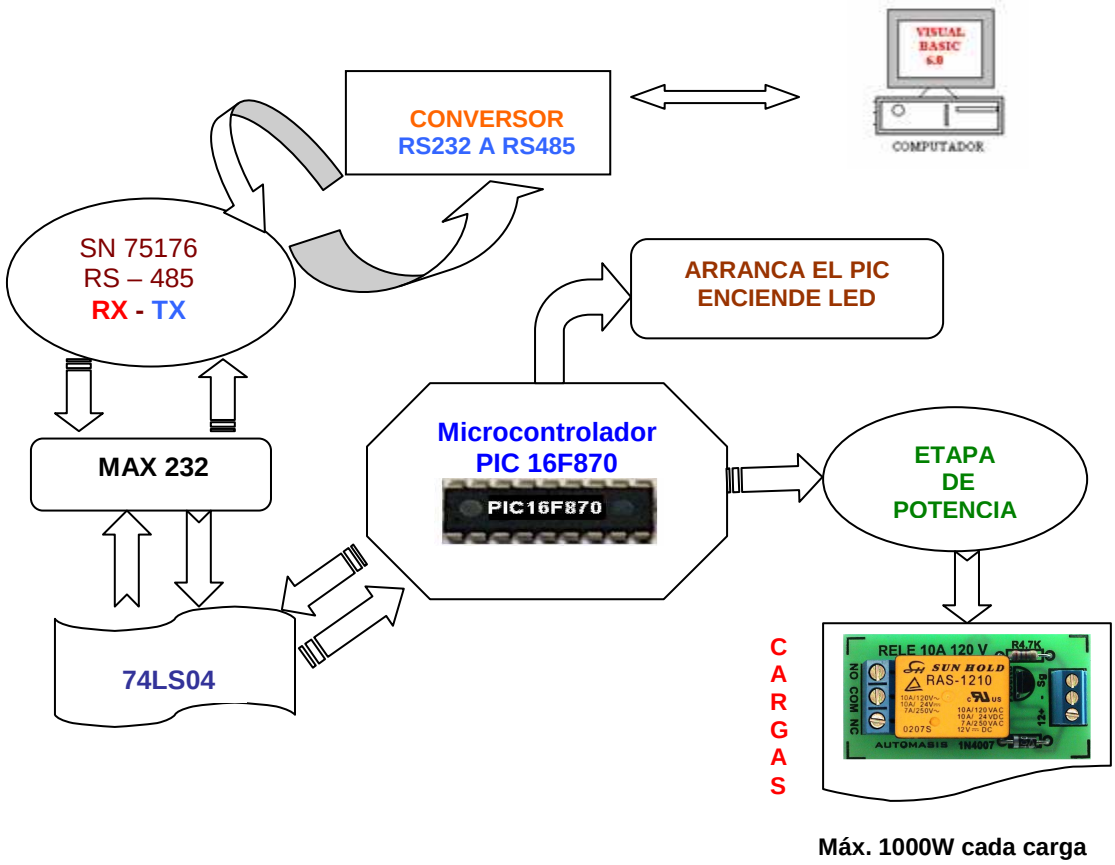


Figura 2.1. Diagrama de bloques del circuito de control

En la figura 2.1 se muestra el diagrama de bloques simplificado del sistema de control propuesto. Como puede verse consta de un circuito conversor RS-232 a RS-485, un MAX 232, un circuito integrado 74LS04, un microcontrolador PIC 16F870, un LED para observar que el PIC arranca y una etapa de potencia con sus respectivas cargas.

2.1.2 DIAGRAMA LÓGICO DE LA PARTE DEL CIRCUITO DE CONTROL

El diagrama lógico de la parte de control indicado en la figura 2.2, nos indica la manera de distribución de los elementos pasivos y activos del circuito de control, aquí podemos observar claramente el funcionamiento del módulo de control y que papel desempeña cada elemento.

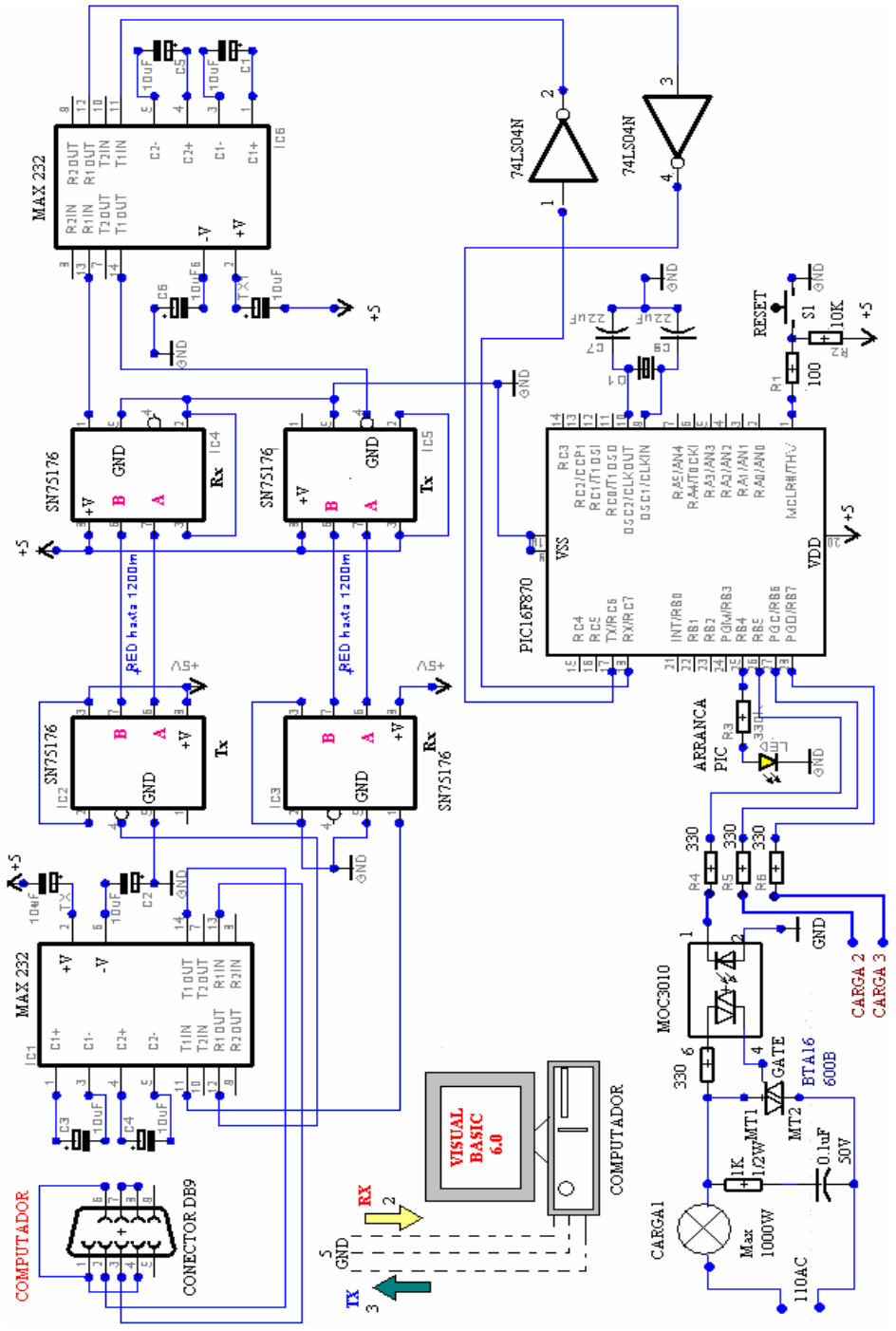


Figura 2.2 Diagrama lógico del circuito de control

2.1.3 FUNCIONAMIENTO DEL CIRCUITO DE CONTROL

2.1.3.1 Características principales del PIC 16F870

- Memoria de Programa de 2 K de 14 bits, EEPROM.
- Memoria de Datos RAM de 128 bytes.
- Memoria de Datos EEPROM de 64 bytes.
- Dispone de una pila de 8 niveles para permitir llamadas a subrutinas anidadas.
- Tiene 11 tipos diferentes de interrupciones.
- Un juego reducido y sencillo de 35 instrucciones.
- El encapsulado es de plástico DIP con 28 pines.
- Permite un rango de frecuencias de trabajo de hasta 20 MHz.
- Dispone de 3 timers (TMR0, TMR1, TMR2), y de perro guardián (WDT).
- Tiene 22 líneas de E/S digitales, divididas en 3 puertos (puerto A 6 líneas, puerto B 8 líneas, puerto C 8 líneas).
- Corriente máxima absorbida por línea: 25 mA.
- Corriente máxima suministrada por línea: 25mA.
- Voltaje de alimentación (VDD) entre 2 y 5.5 V.
- Módulo CCP (captura/comparación/PWM)
- USART (transmisor TX y receptor RX)
- Módulo A/D de 5 canales
- Frecuencia máxima de trabajo: 20 MHz
- Gama de temperaturas operacional de -40℃ a 85℃.

Estas y otras características lo hacen ideal en aplicaciones automotrices industriales, de electrónica de potencia, así como en equipos e instrumentos programables de todo tipo.

El primer paso importante es observar y conocer el diagrama de la distribución de pines del PIC 16F870, figura 2.3.

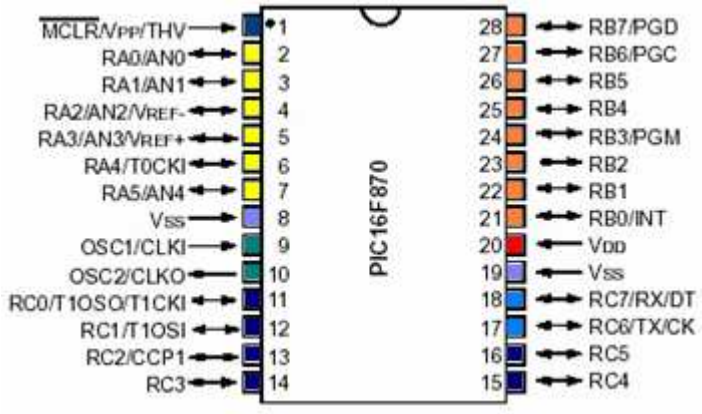


Figura 2.3 Distribución de pines del PIC 16F870

Este circuito integrado cuenta con 22 líneas de E/S digitales, divididas en 3 puertos:

- Puerto A 6 líneas,
- Puerto B 8 líneas, y
- Puerto C 8 líneas

, estos puertos pueden ser configurados como entradas o salidas según sea el caso y consta de 28 pines los cuales se encuentran asignados de la siguiente manera:

Pin 1: ……**MCLR/VPP/THV**
Pin 2: ……**RA0/AN0**
Pin 3: ……**RA1/AN1**
Pin 4: ……**RA2/AN2/RREF-**
Pin 5: ……**RA3/ AN3/RREF+**
Pin 6: ……**RA4/TOCK1**
Pin 7: ……**RA5/AN4**
Pin 8: ……**Vss**
Pin 9: ……**Osc1/CLKI**
Pin 10: ……**Osc2/CLKO**
Pin 11: ……**RC0/T1OSO/T1CK1**
Pin 12: ……**RC1/T1OSI**
Pin 13: ……**RC2/CCP1**
Pin 14: ……**RC3**

Pin 15: ……**RC4**
Pin 16: ……**RC5**
Pin 17: ……**RC6/TX/CK**
Pin 18: ……**RC7/RX/DT**
Pin 19: ……**Vss**
Pin 20: ……**VDD**
Pin 21: ……**RB0/INT0**
Pin22: ……**RB1**
Pin 23: ……**RB2**
Pin 24: ……**RB3/PGM**
Pin 25: ……**RB4**
Pin 26: ……**RB5**
Pin 27: ……**RB6/PGC**
Pin 28: ……**RB7/PGD**

El puerto A esta denotado sus pines en la figura 2.3 con color amarillo, el cual tiene sólo seis pines que pueden ser configurados como entrada o salida.

El pin 1, **MCLR/VPP/THV** sirve para el Reset del micro. También se usa para introducir el voltaje de programación. Para el Reset se debe conectar con una resistencia de 10kΩ a Vcc para que el PIC funcione, si lo quiere resetear entonces pondremos un pulsador normalmente abierto con una resistencia de 100 Ω a tierra. Por lo tanto lleva a cabo dos acciones importantes:

- Se carga un 0 en el Contador de Programa, de forma que después de un Reset siempre se ejecuta la instrucción que está en la posición 0 de la memoria de programa.
- Los registros de estado y control toman un estado conocido y determinado.

El pin 6, o sea, **RA4/TOCK1** a más de ser configurado como entrada o salida puede también sirve para ingresar una frecuencia externa para el temporizador TMR0. Cuando es salida se comporta como colector abierto, por lo tanto debemos poner una resistencia Pull-Up a Vcc de 1kΩ. Cuando es configurado como entrada, funciona como disparador Schmitt Trigger, por lo que puede reconocer señales con un poco de distorsión.

El pin 9, **OSC1/CLKIN** es la entrada del circuito oscilador externo y el pin 10, **OSC2/CLKOUT** es el auxiliar del circuito oscilador.

El puerto B en la figura 2.3, esta denotado sus pines con color anaranjado, y tiene ocho pines, que de igual manera pueden ser configurado como entradas o salidas.

El puerto C en la figura 2.3, esta denotado sus pines con color azul oscuro, y tiene ocho pines, y pueden ser configurado como entradas o salidas.

El pin 17, **RC6/TX/CK** es utilizado en nuestro proyecto, como la salida de datos RS232, además puede utilizarse como I/O señal de reloj asincrónico.

El pin 18, o sea, **RC7/RX/DT** es utilizado para la entrada de datos RS-232 y también puede funcionar como I/O dato serial asincrónico.

La máxima capacidad de corriente para los puertos o pórticos se muestra en la tabla 2.1.

	PUERTO A	PUERTO B	PUERTO C
Modo Sumidero	25mA cada pin	25mA cada pin	25mA cada pin
Modo fuente	25mA cada pin	25mA cada pin	25mA cada pin
Modo Sumidero	150mA por puerto	200mA por puerto	200mA por puerto
Modo fuente	150mA por puerto	200mA por puerto	200mA por puerto

Tabla 2.1 Máxima capacidad de corriente para los puertos

Por último tenemos los pines 8, 19 y 20 que son para la polarización es decir: V_{DD} = 5V y Vss = 0V. El necesario para que nuestro PIC pueda funcionar correctamente, que el oscilador externo este conectado. Sin esto el PIC no funciona. El PIC manda señales eléctricas que recibe el cristal. El cristal es para darle una señal de reloj, por tal razón es el corazón del PIC. A continuación indicamos el más utilizado en la figura 2.4, ya que esta compuesta por un cristal de 4MHz y dos condensadores cerámicos de 22pF.

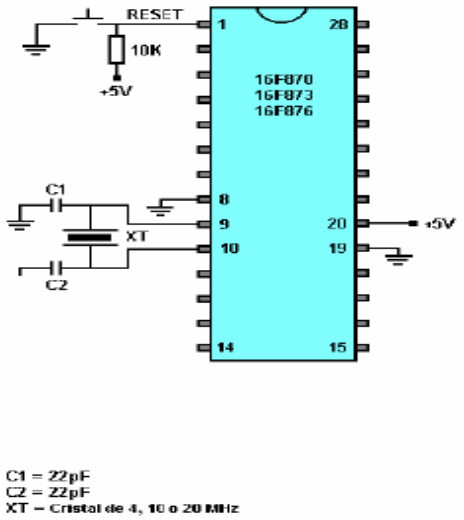


Figura 2.4 Conexión de alimentación, reset y oscilador externo del PIC

La distribución de pines del PIC 16F871 es idéntica a la del PIC 16F870, excepto que EL PIC 16F871, dispone mas líneas I/O adicionales, tanto para el puerto E y el puerto D. El puerto E, en el PIC 16F871 es A/D de tres canales y el puerto D, tiene ocho líneas, y están configurado como I/O digitales según sea el caso.

Para mas detalles del funcionamiento de los pines del PIC 16F870, ver (ANEXO 5).

Los dispositivos del PIC16F870/871 tienen un mueble mostrador de programa de 13 bits capaz de direccionamiento, un espacio de memoria de programa de 8K x 14. Sin embargo el PIC16F870/871 únicamente los primeros 2K x 14, desde 0000h hasta 07FFh, están implementados. El PIC16F870 tiene una memoria de programa tipo FLASH de 2 K direcciones, cada una de ellas con 14 bits, por lo que abarca un rango de direcciones de 0000h a la 03FFh (en total 2048 posiciones).

El vector de RESET está en la dirección 0000h y el vector de interrupción está en 0004h. La pila (STAK) es de 8 niveles, lo cual significa que puede soportar hasta 8 direcciones de retorno de subrutina, Observar en la figura 2.5.

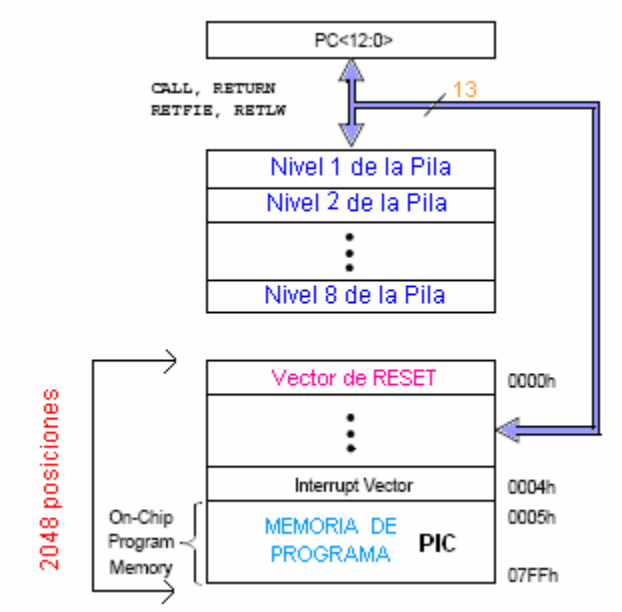


Figura 2.5 Mapa de la organización de la memoria de programa y pilas

El PIC 16F870/871 posee un espacio de memoria RAM de datos de 512x8. La memoria de datos, esta subdividida en cuatro bancos múltiples de 128 bytes cada uno, que contienen el Registros de Propósito General (GPR) y el Registros de Función Especial (SFR). Sin embargo, sólo están implementados 330 bytes, correspondiendo 224 bytes el área de los Registros de Propósito General y 36 bytes al área de los Registros de Función Especial. Los 70 bytes implementados son espejos de algunos SFR de uso frecuente, así como los últimos GPR del banco 0. Esto podemos observar en la figura 2.6.

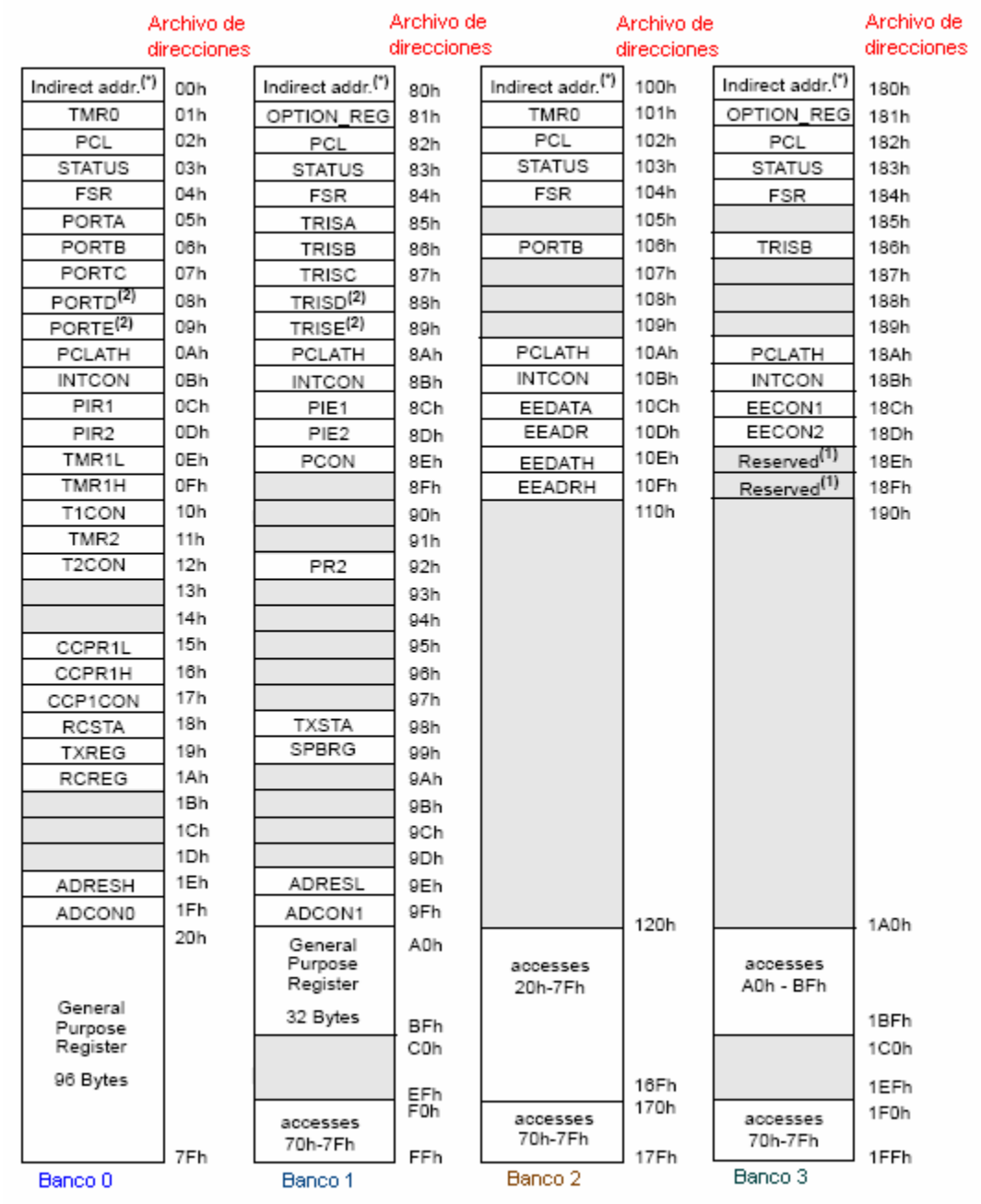


Figura 2.6 Mapa de los archivos de registro de la memoria de datos

En la figura 2.6, los bloques marcados con color plomo, no son implementados, se lee como “0”. Estos registros son reservados, manteniendo estos registros limpios. Además estos registros no son implementados en el PIC 16F870, observe (ANEXO 6).

Obsérvese en la figura anterior, que las localidades 20h a 7Fh, corresponden a los 96 bytes de los Registros de Propósito General (GPR), 80 bytes GPR en el banco 1 (A0h - EFh) y 48 bytes GPR en el banco 2 (120h – 14Fh), dando un total de 224 bytes disponible para el usuario.

Por ejemplo, en las posiciones 0Bh, 8Bh, 10Bh y 18Bh corresponden al registro **INTCON**, de modo que una operación hecha en cualquiera de ellas, se refleja automáticamente en los otros. Se dice, entonces, que las posiciones 8Bh, 10Bh y 18Bh están mapeadas en la posición 0Bh.

En la organización de la memoria de datos podemos seleccionar el banco de memoria con RP1 y RP0, ver en la tabla 2.2.

RP1	RP2	BANCO
0	0	*0*
0	1	*1*
1	0	*2*
1	1	*3*

Tabla 2.2 Organización de la memoria de datos

Cada banco se extiende hasta 7Fh (128 bytes). Las posiciones inferiores de cada banco están reservadas para el Registros de Función Especial (SFR). Por encima del Registros de Función Especial (SFR) está el Registros de Propósito General (GPR), implementado como RAM estática.

En cuanto se refiere a la programación del PIC 16F870, se lo realizó en el programador llamado **IC-Prog**, el mismo que nos muestra las opciones de configuración del PIC.

La programación del PIC está realizado en el programa **IC-Prog,** el cual es de bajo costo y muy fácil de utilizar, este programador esta introducido en el programa Microcode en cual, una vez generado el código hexadecimal lo envía a esté programa, el grafico siguiente vemos como es la pantalla principal del programador, observe figura 2.7.

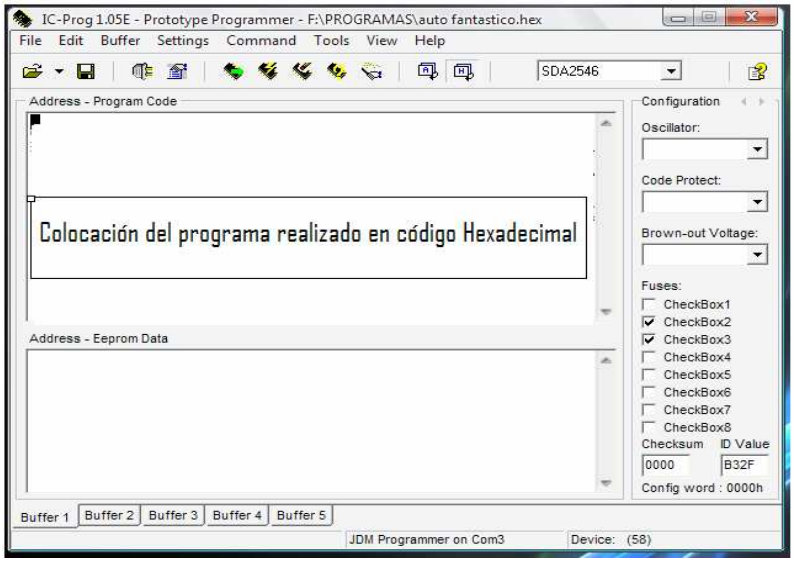


Fig. 2.7 Imagen de la pantalla principal del programador IC-Prog.

Para programar se utiliza un programador físico, el cual se comunica el PIC con el programa, este programador lo realizamos nosotros mismo siguiendo un diagrama investigado en un libro, es muy sencillo y fácil de realizarlo este programador debe ser conectado al puerto serial del computador, una vez que se encuentra conectado el programador verifica su estado y el estado el PIC, si todo esta bien el programador esta listo, pero si encuentra alguna falla el programa nos da una ventana de aviso de error. El programador realizado esta en la figura 2.8.

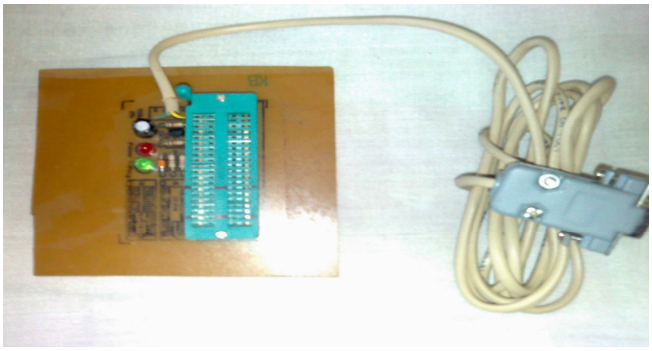


Figura 2.8 Programador de PIC con puerto serial realizado manualmente.

2.1.3.2 Circuito conversor RS-232 a RS-485

2.1.3.2.1 Diagrama lógico del circuito conversor RS-232 a RS-485

Este conversor es realizado para convertir datos de niveles de voltaje de maquina a voltajes TTL, para lo cual hemos consultado en internet pues es la fuente mundial de información en este instante y hemos encontrado un diagrama el cual lo vamos a implementar y realizar las pruebas de funcionamiento correspondientes.

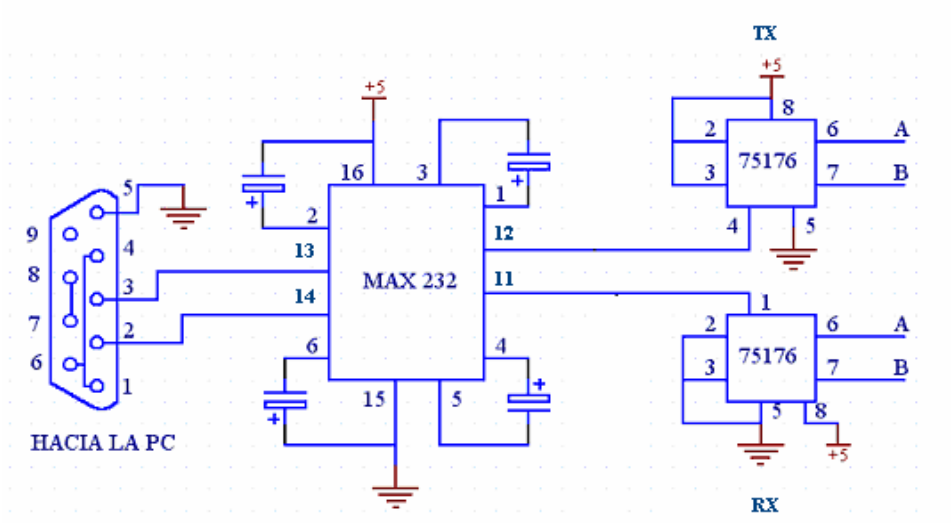


Figura 2.9 Diagrama lógico del circuito conversor RS-232 a RS-485

2.1.3.2.2 Diagrama circuital del circuito conversor RS-232 a RS-485

El conversor RS-232 a RS-485 es utilizado, para cuando se necesita transmitir a largas distancias o con más altas velocidades que RS-232, **RS-485** es la solución. Utilizando enlaces con RS-485 no hay limitación a conectar tan solo dos dispositivos. Dependiendo de la distancia, velocidad de transmisión y los circuitos integrados que utilicemos, se pueden conectar hasta 32 nodos con un simple par de cables.

Dado que la red está establecida con la norma RS-485, debe existir un circuito que convierta dichas señales al formato RS-232 para que así pueda conectarse en la red el dispositivo maestro, que en este caso es el computador, el cual envía o recibe la información. Está tarea implica convertir nuevamente las señales de tipo diferencial a niveles TTL mediante los circuitos integrados SN 75176 y a continuación un circuito integrado MAX 232, que invierte los niveles lógicos TTL a rangos de +15V y –15 V, los cuales son los niveles de tensión adecuados para el puerto serial. En el siguiente circuito observamos el diagrama circuital del conversor realizado por nosotros en el programa Eagle el cual es dictado en la materia de Taller de Circuitos Impresos en la Escuela Politécnica Nacional.

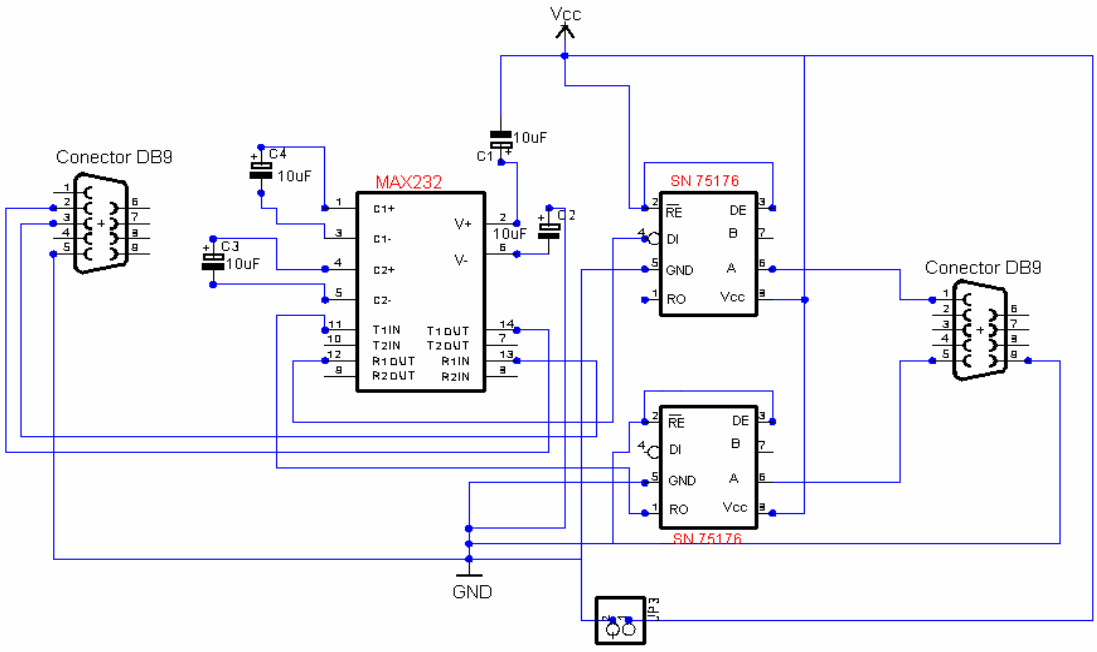


Figura 2.10 Circuito conversor RS-232 a RS-485.

Como observamos en el diagrama circuital en conversor utiliza un Circuito MAX232 y un SN75176 (MAX485) los cuales hace que la señal se convierta, el circuito MAX232 transformando los niveles de voltaje de maquina a niveles TTL y el MAX485 es utilizado para realizar una red balanceada y transmitir a mayor distancia y a mayor velocidad, además mejorando el circuito para que no le afecte el ruido en la transmisión.

2.1.3.3 Etapa de transmisión de datos

2.1.3.3.1 Funcionamiento del circuito de la transmisión de datos

Esta etapa de transmisión y recepción de datos la realizamos con la utilización del computador, el conversor de RS-232 a RS-485 antes mencionado y el circuito de control. Esta es la encargada de entregarnos una señal digital. Esta comunicación se realiza mediante comunicación serial utilizando el puerto **COM1,** con un conector DB9, mediante los pines 2 donde ingresas los datos (recepción) y pin 3 donde envían datos (transmisión), conectados al Circuito MAX232 en los pines 13 y 14 que corresponde al canal 1 del integrado.

Esta etapa de transmisión de datos comienza en el programa realizado en Visual Basic, pues el programa envía unos caracteres determinados a una velocidad predeterminada en el programa por el puerto serial COM1 del computador en el cual esta conectado nuestro DB9 .

Este transmite los datos por medio de pin 3 y conectado al pin 13 del Max en cual cambia los voltajes a niveles TTL, esta información tiene su salida por el pin 12 el cual va conectado a la entrada de circuito SN75176 para convertirse en una red RS485, los voltajes de información en el computador salen invertidos, esto se lo puede observar por medio de la herramienta Hyper Terminal, la cual colocamos en la entrada y salida de cada uno de los circuito utilizados, para poder observar que sucede con la información transmitida.

2.1.3.4 El circuito de potencia

2.1.3.4.1 Diagrama lógico de la parte de potencia

En la figura 2.11 tenemos la conexión de los elementos pasivos y activos del circuito de potencia y de la función que realiza cada uno de ellos en el control para la activación de las tres cargas.

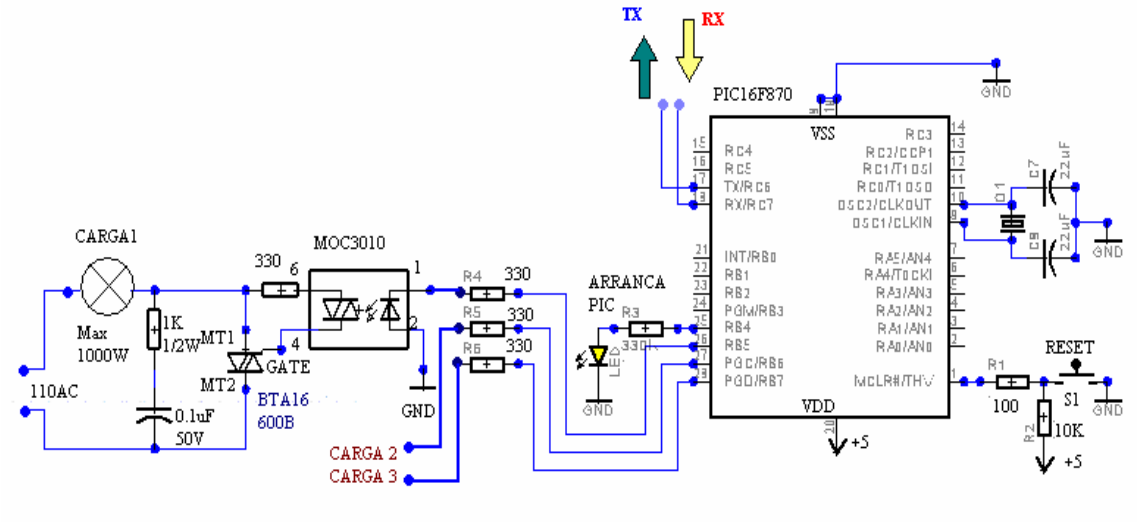


Figura 2.11 Diagrama lógico del circuito de la parte de potencia

2.1.3.4.2 Funcionamiento del circuito de potencia

En este sistema como podemos observar en el diagrama lógico de circuito de potencia se utiliza un opto acoplador MOC3010, un Triac BTA16 600B, dos resistencias de 330Ω a ½ watt, una resistencia de 1KΩ a ½ watt, un condensador de 0.1uF/50V y la respectiva carga 1. De igual manera se repite el mismo circuito para las cargas 2 y 3. Ya este circuito esta realizado para soportar una potencia de 1000W cada carga. Las resistencias son utilizadas para la limitación de corrientes.

Las características eléctricas tanto del Triac de potencia BTA 16 600B, como el opto acoplador MOC3010 podrá ser observado mas adelante, ver (ANEXO 7).

Los pines 26, 27 y 28 (RB5, RB6 y RB7) del microcontrolador PIC 16F870 son utilizados para el funcionamiento del circuito de potencia. Básicamente la etapa de potencia es la encargada de activar y desactivar las cargas.
En el momento que nosotros hagamos clic en los iconos respectivos del software que se realizó en Visual Basic 6.0 se activara la carga que nosotros señalemos.

Es decir el PIC se encuentra esperando por una confirmación de datos válidos mediante el envió de la letra "**S**", "**L**" o "**R**" según el programa que se realizó en Visual Basic 6.0, siendo así los puertos RB5, RB6 y RB7 se encuentra en un nivel alto (+5V), de esta manera el opto acoplador MOC3010 se activa.

La señal digital de los puerto es limitada en corriente y aplicada al cátodo del LED interno del opto acoplador. El ánodo de ese diodo esta conectado a tierra (0V). El brillo producido por el LED acciona al Diac del opto acoplador el cual da una señal de disparo para que a su vez accione al Triac de potencia el cuál habilita el paso de la corriente eléctrica activando a las cargas.

La red RC (resistencia 1KΩ y condensador 0.1uF) conectada en paralelo con el Triac de potencia limita la evolución de la tensión ante cargas inductivas.

2.1.3.5 Etapas de fuentes

2.1.3.5.1 Fuente de alimentación DC

Se compone de una etapa de filtrado a través del condensador electrolítico C1 (4700uF/50V), un regulador de voltaje LM317, dos resistencias (330Ω a ¼ W y 220Ω a ½ W), un diodo 1N4007 que sirve de protección para el LM317 de las corrientes parasitas que pueden ingresar al regular y causarlos serios daños, un diodo LED para observar que la fuente esta siendo regulada y un potenciómetro de 5kΩ para la regulación del voltaje desde (1.25V a 13.15V) para nuestro proyecto necesitamos 5V. Los condensadores C2 y C3 permiten el filtrado armónico, los valores de dichos condensadores son recomendados por el fabricante, ver figura 2.12.

Las características eléctricas del regulador LM317 podrán observarse en los (ANEXO 9).

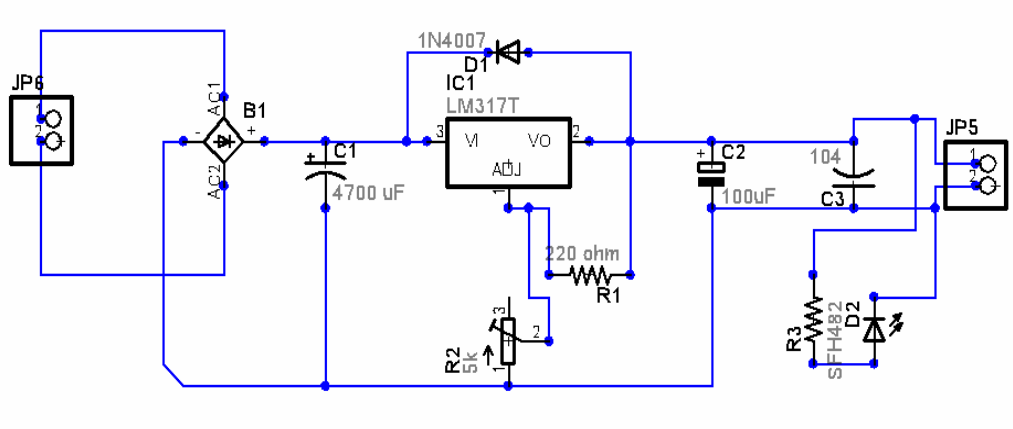


Figura 2.12 Diagrama lógico de la fuente de alimentación DC regulable

Además podemos ver la impresión de las pistas que se realizo en este proyecto para la fuente DC, ver figura 2.13.

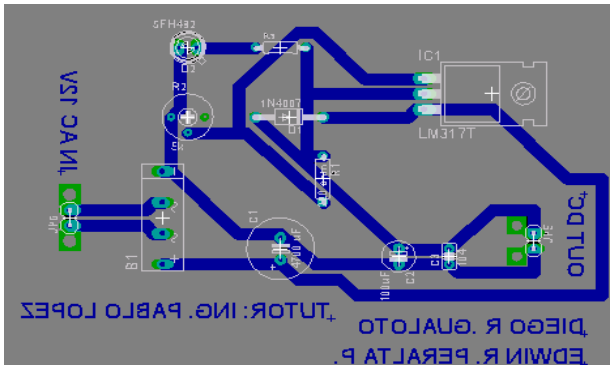


Figura 2.13 Diagrama de las pistas de la fuente DC

2.1.4 CONSTRUCCIÓN DEL EQUIPO

En la figura 2.14 se indica los dispositivos electrónicos que constituye la placa del conversor RS-232 a RS-485 y la forma de cómo se conecta entre si en la placa y con los demás periféricos externos como es el computador y la placa de control. La lista de materiales se proporcionara mas adelante.

Se instala primero los componentes pasivos de bajo perfil como puentes, resistencias, diodos rectificadores, diodo LED, borneras, un zócalo de 16 pines para el MAX232, zócalos de ocho pines para el SN75176, condensadores electrolíticos y los conectores DB9 para placa.

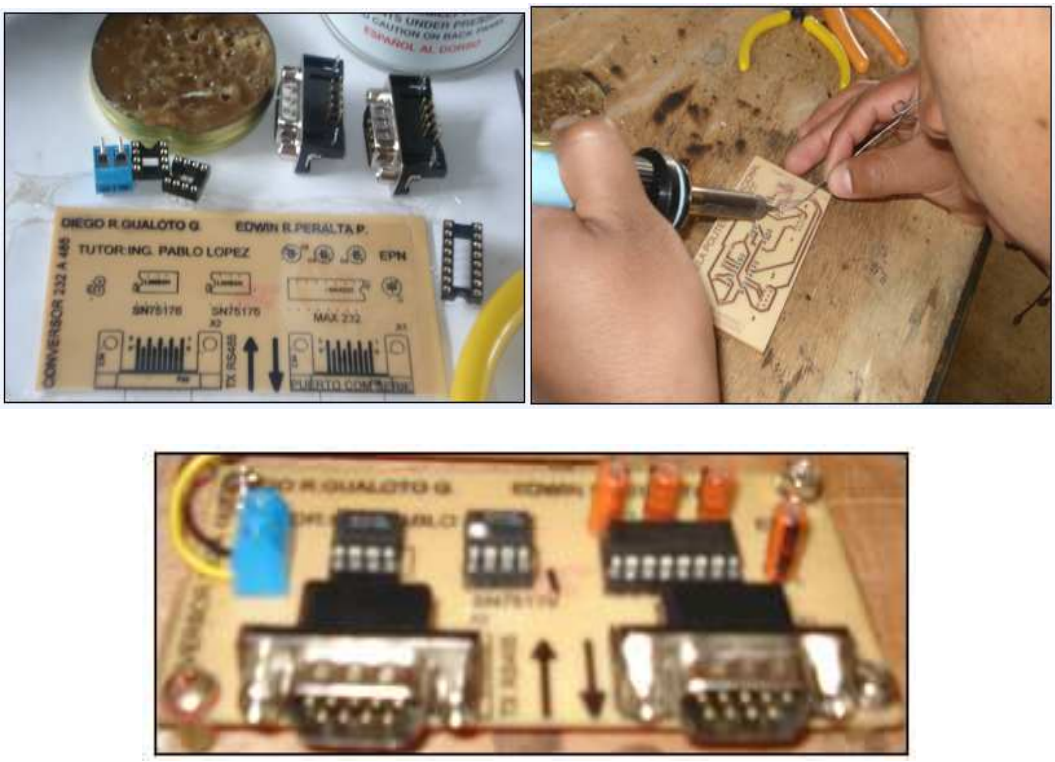


Figura 2.14 Componentes electrónicos del conversor RS-232 a RS-485

Antes de soldar los dispositivos electrónicos debemos revisar cuidadosamente la placa para localizar componentes extraviados, faltantes o mal orientados, pistas abiertas o en corto circuito, etc. Además podemos ver la placa de las pistas del conversor RS-232 a RS-485 realizado en el programa EAGLE, ver figura 2.15.

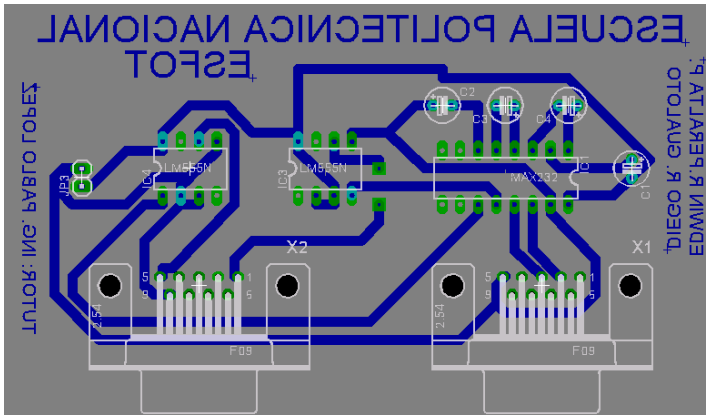


Figura 2.15 Diagrama de las pistas del conversor RS-232 a RS-485

Una vez detectado y corregidos todos los posibles errores de montaje. Luego procedemos a soldar con un cautín en perfectas condiciones, para luego limpiar con un cepillo pequeño y thinner la pasta sobrarte en las pistas. Una vez ya soldados los elementos y limpio las pistas procedemos a echar barniz para que el cobre de las pistas no se oxiden.

Aseguramos y conectamos los componentes externos asociados en una caja de montaje apropiada, ver figura 2.16.

Figura 2.16 Caja de montaje apropiada para el conversor RS-232 a RS-485

Luego en la figura 2.17, se indica los dispositivos electrónicos que constituye la placa de control y el circuito de potencia, y la forma de cómo se conecta entre si y con los componentes periféricos externos como es el computador y el conversor. La lista se detallara mas adelante.

Primero se debe instalar los componentes de bajo perfil como puente, resistencias, pulsador, borneras, zócalos (para el PIC 16F870, MAX232, SN75176, 74LS04, MOC3010), condensadores electrolíticos y cerámicos, un cristal 4MHz, los conectores DB9 para placa, diodos LED's, Triac's y borneras. Y una vez que procedamos a soldar debemos tener muy en cuenta en no calentar demasiado a los dispositivos electrónicos ya que podemos causarles daños serios y su funcionamiento no va ser el correcto. Debemos revisar cuidadosamente la placa para que ningún elemento falte en el momento de soldar la placa.

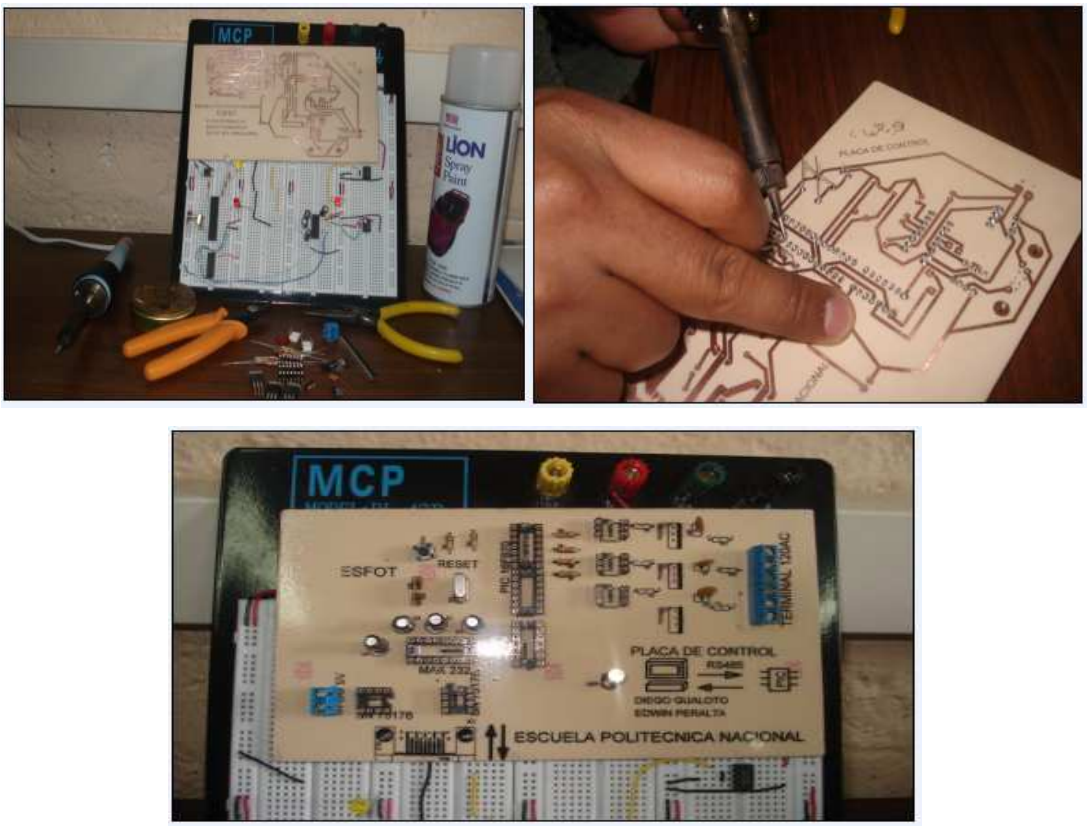


Figura 2.17 Componentes electrónicos del circuito de control y potencia

Además podemos observar la impresión para las pistas de la placa de control, figura 2.18. Como sabemos esto se realiza con el papel térmico para tener una correcta y efectivas pistas.

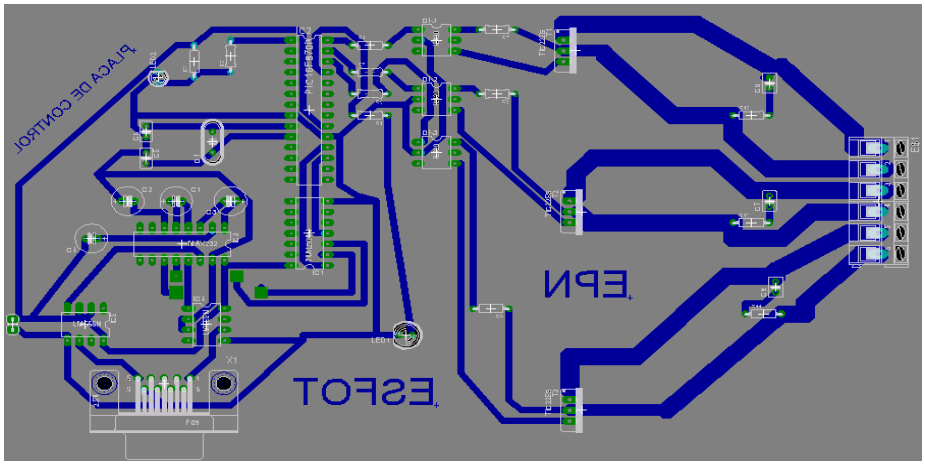


Figura 2.18 Diagrama de las pistas de la placa de control

Luego que la placa este completamente terminado de soldar y limpia procedemos a echar el barniz aislante para una mejor protección de las pistas, Debemos tener muy en cuenta y cuidado con los triac's ya que estos necesitan disipadores grandes, porque estos dispositivos disipan demasiado calor por lo cual en este proyecto a mas de colocar los disipadores se coloco un ventilador, aseguramos y conectamos los componentes externos asociados en una caja de montaje apropiada, ver figura 2.19.

Figura 2.19 Caja de montaje apropiada para el circuito de control y potencia

2.1.5 PRINCIPIO DE FUNCIONAMIENTO

El control computarizado propuesto se realiza mediante el computador, el conversor, la placa de control y las respectivas cargas. En la figura 2.20, se observa la conexión completa para el control de las cargas. Una vez conectado todo el equipo, vamos al computador, abrimos el programa que esta realizado en Visual Basic 6.0, donde nosotros damos un clic en cualquiera de las tres cargas con el mouse y hacemos la transmisión de datos mediante la interfaz 485, de esta manera enviamos una información al PIC y el cual es la encargada de realiza la respectiva activación de la carga. El PIC lee estos datos y activa al circuito de potencia respectivo. Adicionalmente cuando la carga esta activada el PIC envía información al computador indicando que la carga esta activada. Esta información se visualiza en el computador cuando la carga activada cambia de color. El programa realizado en Visual Basic es diferente al programa del PIC, pero siempre debe estar a la misma velocidad de transmisión.

Figura 2.03 Conexión completa del control computarizado

2.1.5.1 Descripción del montaje frontal de la placa de control

El montaje frontal de la placa de control presenta dos led´s indicadores que son que realiza lo siguiente:

- Led verde: activa a todo el circuito de control.
- Led amarillo: indica que el PIC arranco.

, proporcionando información necesaria para el control. Ver figura 2.21.

Figura 2.21 Parte frontal del circuito de control

De igual manera en la parte frontal de la caja se encuentra el conector DB9 que nos servirá para realizar la comunicación serial de datos con la interfaz 485.

2.2 DESARROLLO DEL SOFTWARE EN VISUAL BASIC PARA EL CONTROL COMPUTARIZADO.

Una vez realizada la construcción del equipo de control de la red eléctrica, vamos a observar en este subcapitulo, como se realizó el software para el funcionamiento de la comunicación serial de Visual Basic y PIC.

Este es la parte de la electrónica que más entusiasmo genera en este proyecto, poder controlar a través de un computador todos los periféricos de un departamento, casa o edificio (ascensor, luces, alarmas, cerraduras, áreas comunales, etc.), todo desde la pantalla de un computador. Con esto automatizamos el control de un edificio, esto hoy en día es el llamado edificio inteligente.

Entonces para introducirnos en el mundo de control computarizado hacemos en este proyecto un control de tres cargas (focos principales, tomacorriente y focos secundarios), los cuales nos responderán si están activados o no, para hacer el tablero de control se necesita saber programar en VISUAL BASIC (**VB**). Empezamos por diseñar los botones en un form VB, para esto ejecutamos el programa VB, en la pantalla principal escogemos **EXE estándar** y damos clic en abrir, ver figura 2.22.

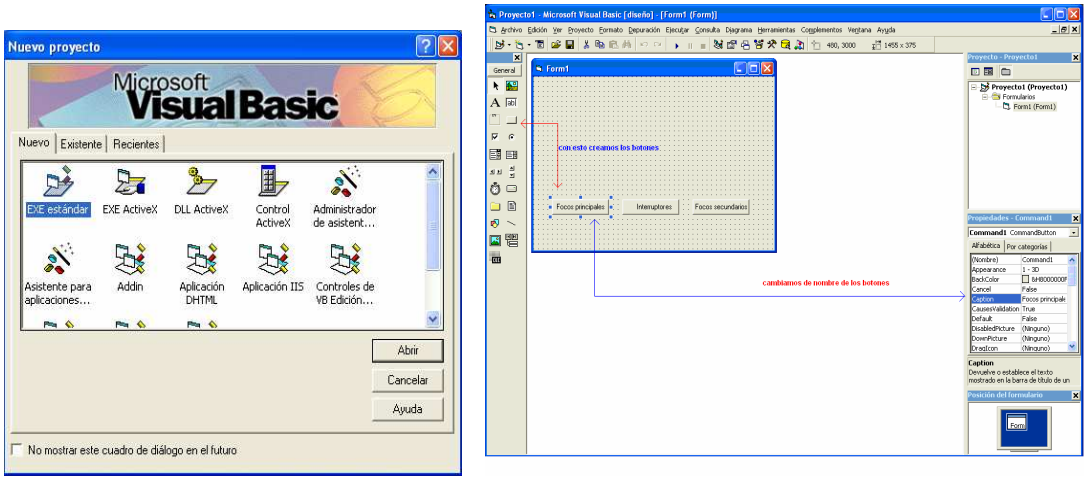


Figura 2.22 Pantalla para ejecutar VB y para diseñar los botones

Como se puede ver en la pantalla de la derecha hemos creado tres botones con la herramienta **commandButton** , si desea puede cambiar el texto de command1. Por ejemplo, por Focos principales, para esto debemos seleccionar el objeto en este caso Command1 y en propiedades del objeto que se encuentra al lado derecho especificamos en **Caption -** Command1, lo borramos y escribimos **Focos principales,** el otro botón escribimos **Interruptores** y el último **Focos secundarios**, quedará como ilustra la figura 2.22.

Una vez creado los tres botones con sus respectivos nombres, ahora vamos a dibujar tres objetos, esto se realizo utilizando la barra de herramientas de VB. Para poner sus respectivos colores y grosor de las líneas utilizamos en propiedades del objeto como antes ya se mencionó donde esta ubicado, es decir a lado derecho. Para crear texto presionamos en el lado izquierdo en **label** , lo colocamos en el lugar que deseamos y luego en **Caption** ponemos ON, OFF. Para cambiar el estilo de letra y el tamaño, primero seleccionamos el texto a modificar y en el lado derecho en propiedades **Front** elegimos los cambios que deseemos y listo, ver figura 2.23.

Figura 2.23 Pantalla con los objetos y el texto

Para dar colores a los objetos de los focos principales y secundarios, seleccionamos el círculo y en propiedades donde dice FillStyle **transparent** cambiamos a **Solid**, luego en FillColor escogemos la paleta y ponemos el color plomo para indicar que los focos están apagados.

Para habilitar la comunicación serial, damos clic con el botón derecho sobre el cuadro General y escogemos la opción componentes, luego saldrá una pantalla con una lista de componentes y escogemos **Microsoft Comm Control 6.0**, seleccionamos y damos clic en aceptar, ver figura 2.24

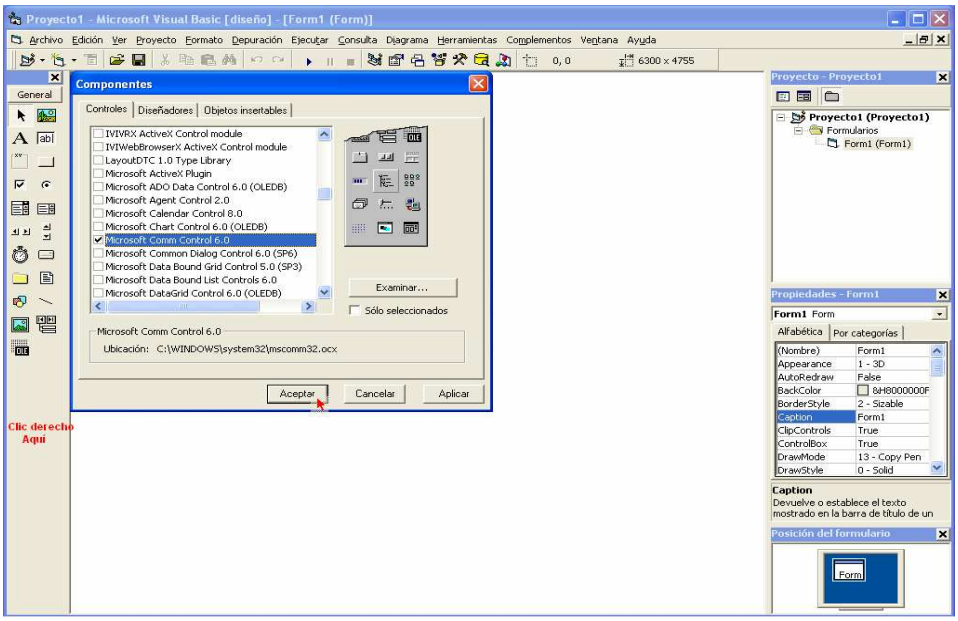


Figura 2.24 Pantalla para seleccionar la comunicación serial

Luego de hacer clic en Aceptar, notaremos que ahora aparece un icono nuevo un **teléfono**, colocamos este teléfono en la **form**, y en las propiedades **Commport** podemos modificar si es COM1 o COM2, también modificamos la velocidad de transmisión que por defecto viene con 9600, n, 8,1, y también colocamos un **Timer** , ver figura 2.25.

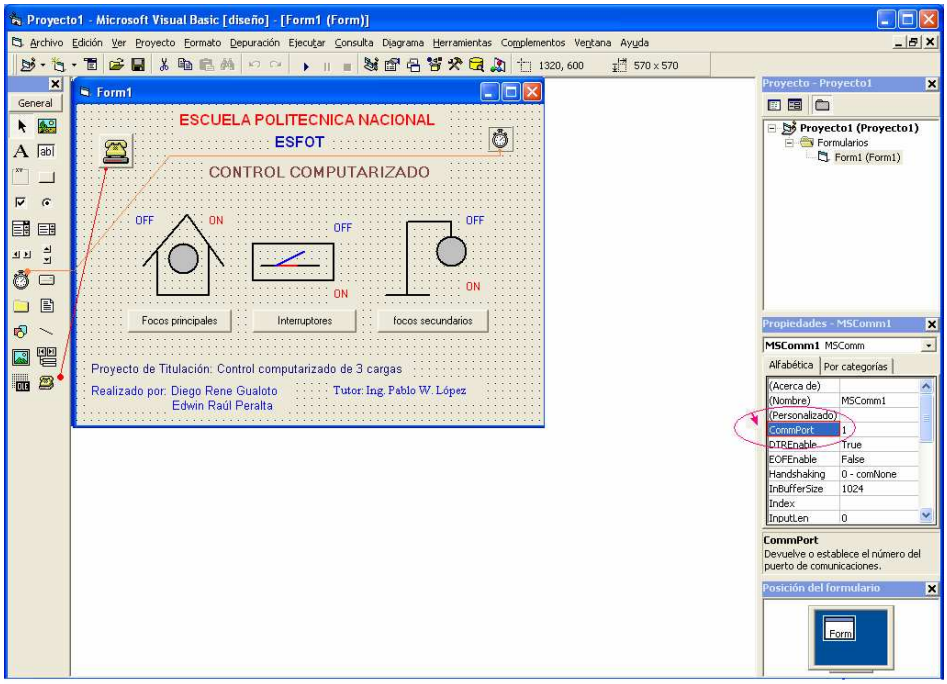


Figura 2.25 Pantalla para observar el puerto COM1

Bien, ahora ON y la línea del interruptor que señala a ON, no deben aparecer, sino hasta cuando el microcontrolador se los indique, por tal razón debemos ocultarlos, y esto lo realizamos seleccionando a cada uno y en propiedades donde dice Visible **True**, lo cambiamos por **False**, bien en ese instante no desaparecerán sino hasta cuando se ejecute el programa.

Es importante darles nombres a cada objeto sólo a los que necesariamente vamos a modificar, esto se realiza de la siguiente manera, primero seleccionamos, digamos que el botón **Focos principales**, al lado derecho el primer ítem de propiedades dice (Nombre) Command1 aquí lo ponemos el primer mismo nombre así FOCOSP, esto lo hacemos con el objeto de no confundir un nombre Caption Focos principales con el nombre del objeto FOCOSP, lo mismo hacemos con los siguientes objetos: la palabra ON y OFF, los tres botones Focos principales, interruptores y focos secundarios y la línea del interruptor, todos ellos su nuevo nombre a cambio de : Command1, Command2, Command3, ShapeX, LabelX, LineX, ver tabla 2.4.

Propiedad	N. antiguo	Propiedad	Nombre nuevo
(Nombre)	Command1	(Nombre)	FOCOSP
(Nombre)	Command2	(Nombre)	INTERRUP
(Nombre)	Command3	(Nombre)	FOCOSZ
(Nombre)	ShapeX	(Nombre)	Shape1
(Nombre)	ShapeX	(Nombre)	Shape2
(Nombre)	LabelX	(Nombre)	ON1
(Nombre)	LabelX	(Nombre)	OFF1
(Nombre)	LabelX	(Nombre)	ON2
(Nombre)	LabelX	(Nombre)	OFF2
(Nombre)	LabelX	(Nombre)	ON3
(Nombre)	LabelX	(Nombre)	OFF3
(Nombre)	LineX	(Nombre)	LINEAOFF
(Nombre)	LineX	(Nombre)	LINEAON

Tabla 2.4 Tabla que muestra los cambios de nombre a realizar

A continuación una imagen de cómo se cambia el nombre de un objeto, observen que se seleccionó el Círculo y en el lado derecho decía (Nombre) Shape1, se lo cambió por FOCOSP, de esta manera cambiamos los 13 objetos y los nombres que aparece en la tabla 2.4, ver figura 2.26.

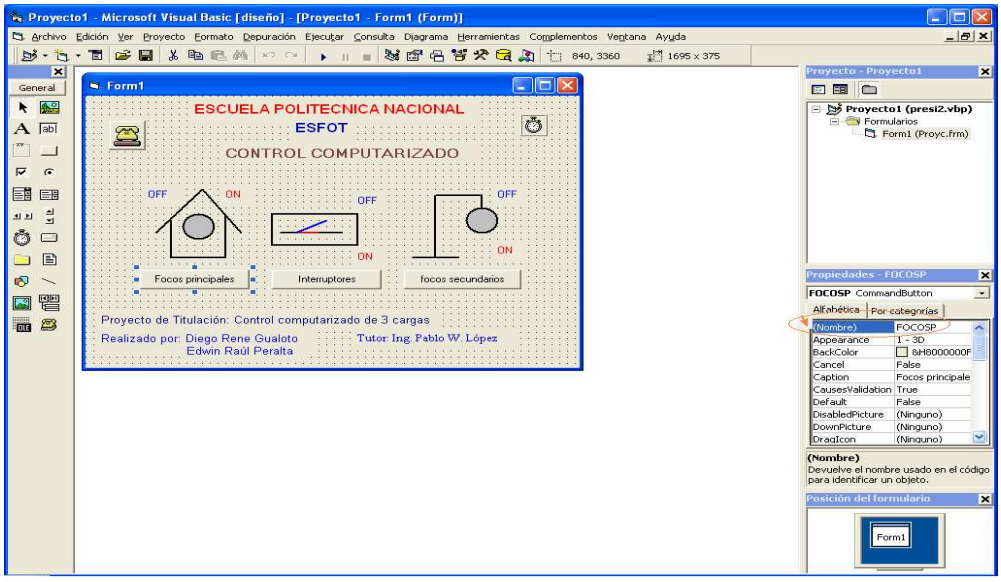


Figura 2.26 Pantalla de cómo cambiar el nombre

Es hora de programar las funciones de los botones, para esto primero damos doble clic en cualquier parte de la form, esto hará que la otra pantalla (Código), en la que sale un texto así, ver figura 2.27.

Figura 2.27 Pantalla de código

Aquí escribimos habilitar el puerto y corre el Timer con intervalos de 1ms, y empezamos a programar cada uno de los botones.

```
Private Sub Form_Load()
MSComm1.PortOpen = True    'habilitar el puerto 1
Timer1.Interval = 1        'correr el timer a 1 milis
End Sub
```

Esto nos indica que cuando se ejecute este programa corra su contenido, es decir abra el puerto **COM** y empiece a correr el Timer con intervalos de 1ms. El programa completo se lo puede apreciar en el anexo, ver (ANEXO 9).

Una vez ya cambiado todos los nombres, realizado el programa lo hacemos correr presionando  y cuando ya ha probado con la comunicación del PIC y sabe que está bien puede crear un archivo.exe (ejecutable) en donde dice, figura 2.28, que se ilustra.

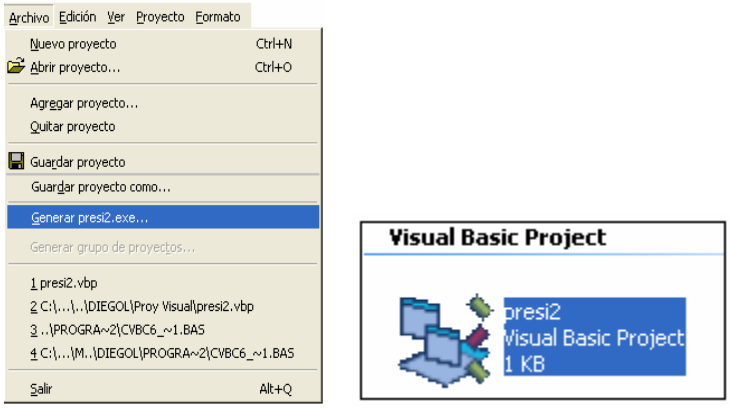


Figura 2.28 Crear un archivo ejecutable (presi2) y el icono

2.3 DESARROLLO DEL PROGRAMA PARA EL C.I. PIC 16F870

Una vez realizado el programa en Visual Basic 6.0, es el momento de realizar el programa que va ir grabado en el PIC. Para esto hemos utilizado los programas Microcode Studio y IC-Prog 1.05E.

Antes de comenzar debemos tener instalado el Microcode Studio y IC-Prog 1.05E en nuestro computador, el Microcode es un programa editor de texto como el Bloc de notas de Windows, pero con la diferencia que este está hecho para facilitar la programación de los microcontroladores PIC, estos programas se lo pueden descargar gratuitamente en Internet. Para empezar ejecutamos el programa Microcode Studio, haciendo doble clic el icono mostrado anteriormente. Una vez abierta la pantalla principal comenzamos a escribir el programa, ver figura 2.29.

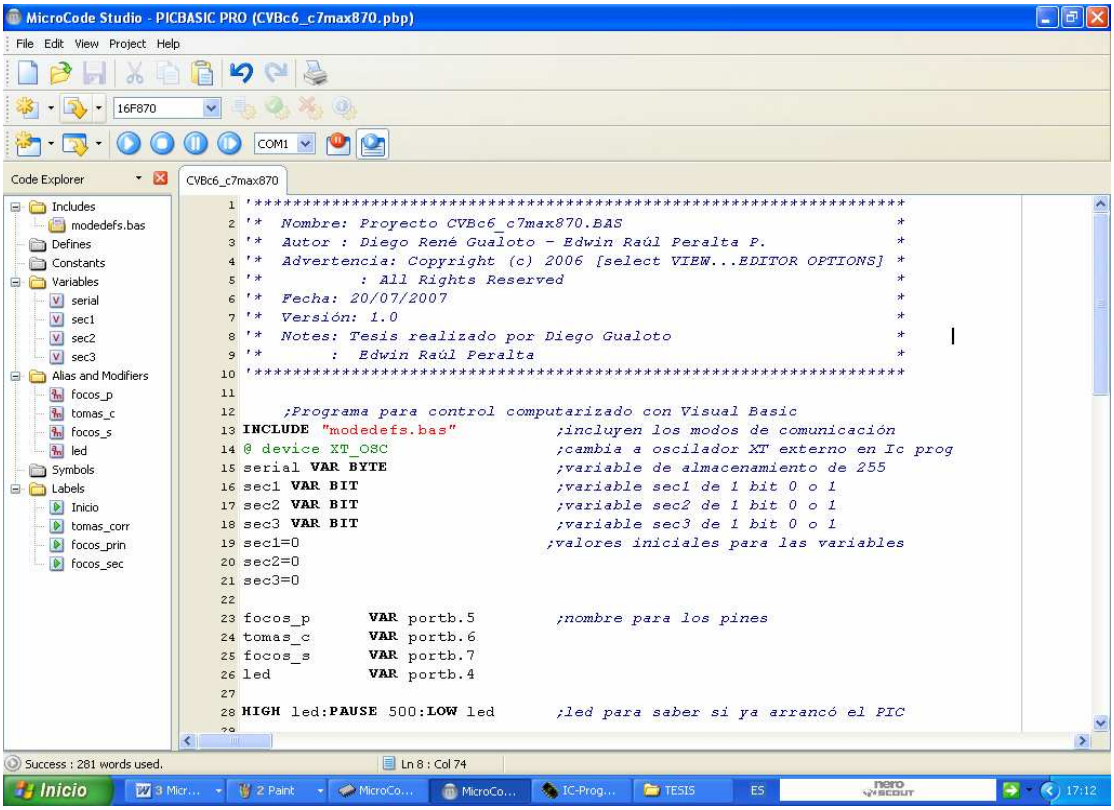


Figura 2.29 Pantalla principal del Microcode Studio con el programa x .bas

El programa realizado de detallara mas adelante en el anexo. Ver (ANEXO 10).

Ahora, los procedimientos para programar son muy sencillos primero debemos escoger el modelo PIC a utilizar, es decir PIC 16F870 que se lo puede visualizar en la pantalla principal en la parte superior izquierda de la barra de herramienta. Además seleccionar el COM adecuado, para este caso es el COM1. Y por último escoger la misma velocidad de transmisión que se utilizó en el programa de Visual Basic, es decir 9600, n, 8, 1.

Luego de escribir todo el programa completo guárdelo bajo un nombre o sea un archivo ejecutable (CVBc6_c7max870.BAS). Luego terminado la configuración hacemos la compilación y programación respectiva. Para esto presionamos la tecla F10 o el icono  . Existe otra manera de realizarlo, en la barra de menú hacer clic en Project y luego seleccionamos **Compile and Program**, si el programa esta bien escrito y sin fallas se compilará y mostrará en la parte inferior el espacio que requiere en el PIC, para este proyecto es necesario un espacio de 281 palabras usadas, ver figura 2.29. Luego de esto enseguida se creara automáticamente 3 archivos: .MAC, .ASM y .HEX (hexadecimal), este último es el mas importante para el PIC y es el que se debe grabar en el microcontrolador. Una vez compilado el programa, automáticamente se abre otra pantalla con el programa IC-Prog 1.05E, esto nos sirve para cargar el programa que esta con archivo hexadecimal (**CVBc6_c7max870.Hex**) y luego procedemos a grabar. Para esto primero debemos saber en que carpeta esta grabado el programa, entonces en Ic-Prog 1.05E, hacemos clic en Archivo en las barra de menú, y seleccionamos Abrir archivo o Ctrl+O y hacemos clic en Abrir, como se ilustra en la figura 2.30.

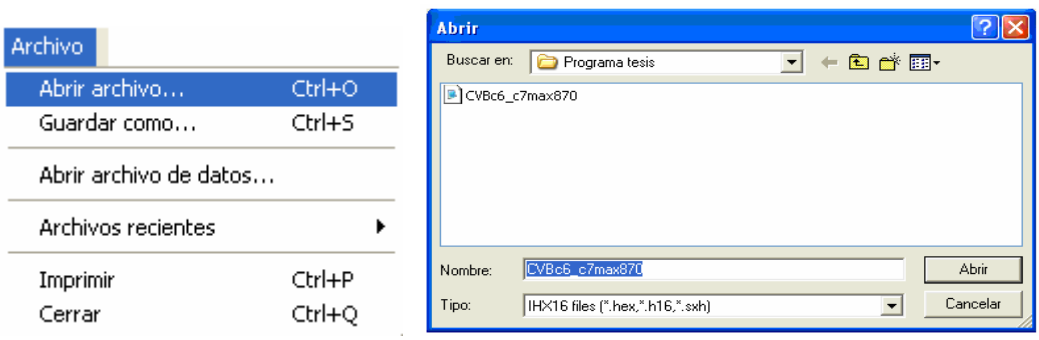


Figura 2.30 Pasos para abrir el programa hexadecimal

Una vez abierto el archivo observamos en la pantalla del IC-Prog 1.05E, como esta cargado el archivo que se va a grabar en el PIC, debemos también seleccionar el modelo del PIC, noten que el código del programa está sólo 3FFFh esto quiere decir que está vació, no hay ningún programa a grabarse. Para lo cual debemos tener conectado el grabador de micros para introducir el PIC, para conectar este grabador debemos tener en cuenta en que puerto COM esta configurado, es decir conectar en el COM correcto, en nuestro caso está configurado para el COM1, ya que el computador consta de dos puertos COM, ver figura 2.31.

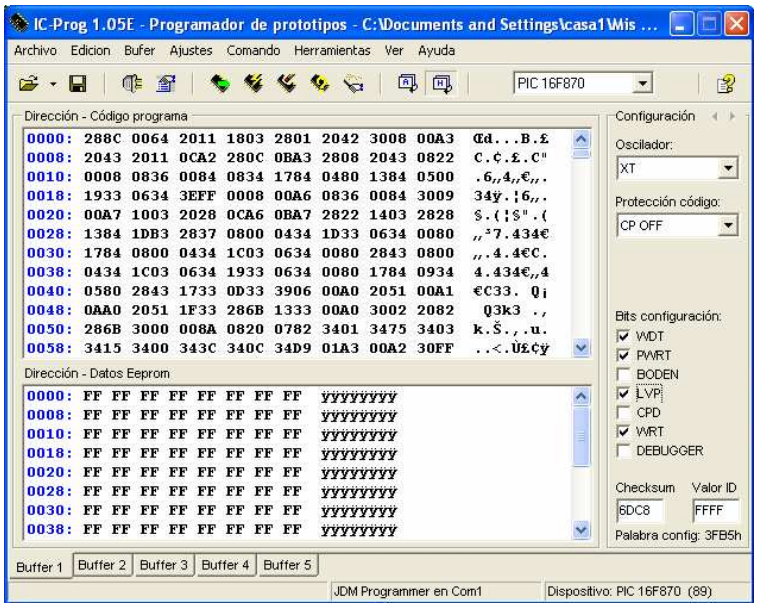


Figura 2.31 Presentación de la pantalla cargado el archivo hexadecimal

Después de abrir el archivo .HEX (No antes), proceda a cambiar la configuración del oscilador a XT y la protección de código apagada, esto se lo puede localizar en la parte izquierda de la figura 2.31. Si ya esta listo se instalado el PIC en el grabador de micros, presione la tecla  o F5 y espera a que salga el siguiente mensaje: **Verificación correcta!**

Esto nos indicará que el PIC fue grabado exitosamente, por último procedemos a sacar del grabador el PIC para ser utilizado. El archivo cargado en hexadecimal para grabar el PIC lo podemos ver (ANEXO 11).

2.4 PRUEBA DE LA RED DE CONTROL ELÉCTRICA CON EL MICROCONTROLADOR Y EL COMPUTADOR.

2.4.1 PRUEBA DE LA RED DE CONTROL CON EL MICROCONTROLADOR Y EL COMPUTADOR.

Como se podrá dar cuenta esta es la parte más importante del proyecto, ya que se va a realizar las verdaderas pruebas con cargas reales. Entonces para comenzar necesariamente se necesita ejecutar el programa realizado en Visual Basic, Hyper Terminal y realizar las conexiones correctas entre los periféricos internos y externos como se indicó anteriormente en los otros subcapitulos.

Para la ejecución del Hyper Terminal hacemos clic en **Inicio,** seleccionar **Todos los programas, Accesorios, Comunicaciones** y para finalizar clic en **Hyper Terminal.** Una vez abierto el programa, primero nos pide escribir un nombre (Diegol) y elegir un icono para la conexión (seleccionamos), ver figura 2.32.

Figura 2.32 Pantalla de la descripción de la conexión

Luego hacemos clic Aceptar y nos presenta otra pantalla donde nos pide escribir detalles del número de teléfono que se desea marcar, para nuestro proyecto donde dice **País o región** seleccionar con el mouse *Ecuador (593)*,

Código de área escribimos *02*, Número de teléfono dejamos vació y en **Conectar usando** seleccionamos *COM1*, ver figura 2.33.

Figura 2.33 Pantalla para escribir detalles de números a conectar

Una vez escritos todos los datos pedidos como se demostró en la figura 2.33 hacemos clic en Aceptar y nuevamente aparece otra pantalla está es la principal ya que aquí se va a configurar las propiedades de COM1, ver figura 2.34.

Figura 2.34 Pantalla para las propiedades de COM1

Para comenzar la configuración del puerto, en **Bits por segundos** seleccionar la velocidad que se utilizó en el programación del PIC y del Visual Basic, ya que es la misma velocidad (*9600*), **Bits de datos** (*8*), **Paridad** (*Ninguno*), **Bits de parada** (*1*) y **Control de flujo** (*Ninguno*), como se ilustra en la figura 2.34.

Y por último hacemos clic en Aplicar y Aceptar y se abre la pantalla principal donde se va observar los datos que envía y recibe tanto el computador como el PIC, ver figura 2.35.

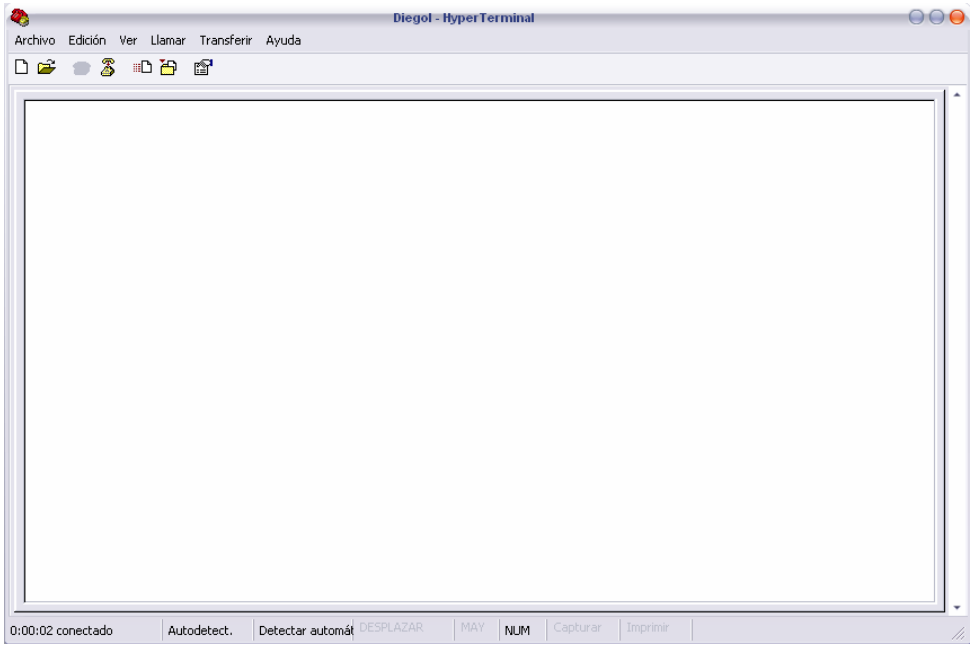


Figura 2.35 Pantalla principal del Hyper Terminal

Para observar los datos tanto enviados como recibidos debemos nosotros utilizar el programa Visual Basic (VB), una vez conectados todo correctamente hacemos clic en el primer botón de VB, es decir en Focos principales e inmediatamente observamos que en el tablero de control de VB el icono cambia de color **plomo** a color **verde** esto nos da la certeza que la carga esta encendida y si hacemos clic en el segundo botón (Interruptores) aquí la línea **azul** inclinado se cambia a la línea recta de color **rojo** y el último botón (focos secundarios) el icono cambia de color **plomo** a color **celeste** , ver figura 2.36.

También debemos notar que en los tres botones del tablero de control una vez activados las cargar la palabra ON aparece automáticamente esto se realiza en la programación de VB.

Figura 2.36 Tablero de control con las cargas activadas y desactivadas

Por lo tanto una vez observado en el tablero de control los cambios antes descritos, vamos a fijarnos en la pantalla del Hyper Terminal que las letras sean las correctas, es decir las que se programo en Visual Basic. Por ejemplo aquí utilizamos un cable multifilar conectado el DB9, esto nos servirá para ir viendo los datos en cualquier entrada o salida que nosotros necesitemos ver el dato, aquí necesitamos la ayuda de dos computadores; una para el manejo de VB y otro para ver en la pantalla del Hyper Terminal si las letras son correctas para la primera carga se debe visualizar la letra “S”, segunda carga la letra “R” y la tercera carga la letra “L”, ver figura 2.37.

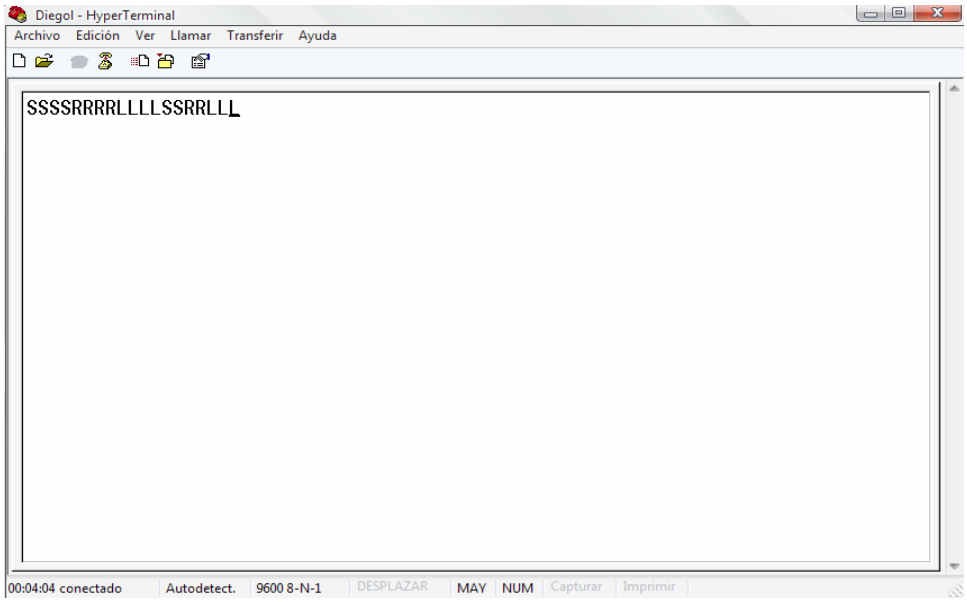


Figura 2.37 Pantalla visualizando las datos correctos en Hyper Terminal

2.5 ANÁLISIS TÉCNICO-ECONÓMICO

2.5.1 ANÁLISIS TÉCNICO DEL PROYECTO

Los microcontroladores PIC, han permitido potencializar sus aplicaciones gracias a la gran facilidad de su programación y las diversas ventajas que ofrecen con respecto a los microprocesadores. Prueba de ello es que se puede realizar sistemas muy completos con el menor uso de recursos.

A continuación detallamos algunas características del proyecto realizado:

Fácil manejo.- El control computarizado presenta al usuario una fácil manipulación desde el computador en el control de aparatos de ventilación hasta el control de iluminación (áreas comunales). Con un simple clic en el icono correspondiente, el usuario puede activar o desactivar las cargar antes mencionados en este proyecto.

Amplio mercado de repuestos.- Todos los dispositivos utilizados en este proyecto están disponibles en el mercado nacional, lo cual permite reemplazar cualquiera de ellos sin ninguna dificultad.

2.5.2 ANÁLISIS ECONÓMICO DEL PROYECTO

Los controles para automatizar un edificio o una casa existentes en el mercado que se ofrecen en la actualidad son de costos relativamente elevados, por lo que este proyecto se convierte en una opción para reducir los costos de los mismos. A continuación se detalla en la tabla 2.5 el costo de todos los dispositivos electrónicos usados para la realización de este proyecto.

	ELEMENTOS	CANTIDAD	C. UNIDAD	C. TOTAL
1	Baquelita 10x20 (vidrio)	1	1,25	1,25
2	Baquelita 10x20	2	0,85	1,70
3	Acido sulfúrico	4	0,45	1,80
4	Papel térmico	2	1,90	3,80

5	Esponja de acero	1	0,60	0,60
6	PIC 16F870	1	7,90	7,90
7	CI MAX 232	2	2,50	5
8	CI SN75176	4	1,50	6
9	CI 74LS04	1	0,50	0,50
10	MOC3010	3	1	3
11	LM317	2	0,80	1,60
12	Triac BTA16 600B	3	0,95	2,85
13	Puente de diodos	2	0,60	1,20
14	Transformador 110/12V	2	3,50	7
15	Disipador de calor	3	0,70	2,10
16	Conector DB9 placa	3	1,05	3,15
17	Conector DB9 hembra	4	0,80	3,20
18	Tapa para el DB9	4	0,16	0,64
19	Bornera de dos pines	9	0,45	4,05
20	Cristal 4MHz	1	0,85	0,85
21	Diodo 1N4007	2	0,10	0,20
22	Conector para fuente	1	2,50	2,50
23	Cable conector	1	1,50	1,50
24	Conector macho fuente	1	0,70	0,70
25	Zócalo para el PIC	3	0,20	0,60
26	Zócalo para el MAX	2	0,32	0,64
27	Zócalo para el MOC	3	0,85	0,85
28	Zócalo para el 75176	4	0,15	0,60
29	Cable multifilar 2 pares	100m	0,37	37
30	Potenciómetro 5k	2	0,35	0,70
31	Led verde 5mm	2	0,10	0,20
32	Led amarillo 5mm	1	0,10	0,10
33	Condensador 10uF/25V	12	0,15	1,80
34	Condensador 4700uF	2	0,85	1,70
35	Condensador 22pF	2	0,10	0,20
36	Condensador 0,01uF	5	0,10	0,50
37	Condensador 100uF	2	0,15	0,30

38	Pulsador N.A. placa	1	0,15	0,15
39	Resistencia 100Ω ¼ W	2	0,05	0,10
40	Resistencia 10kΩ ¼ W	2	0,05	0,10
41	Resistencia 100Ω ½ W	1	0,08	0,08
42	Resistencia 1kΩ ½ W	3	0,08	0,24
43	Resistencia 330Ω ½ W	3	0,08	0,24
44	Resistencia 220Ω ½ W	2	0,08	0,16
45	Resistencia 330Ω ¼ W	6	0,05	0,30
46	Ventilador 110VAC	1	7,60	7,60
47	Caja para conversor	1	17	17
48	Caja para control	1	25	25
49	Cable para conexión	5m	0,10	0,50
50	Focos 110/200W	5	0,85	4,25
51	Tomacorriente	1	10	10
52	Reflector de 1000W	1	12	12
53	Barniz	1	2,50	2,50
54	Otros	-	-	15
			TOTAL $ →	**200,10**

Tabla 2.5 Costos de los elementos electrónicos utilizados en el proyecto

CAPITULO 3

CONCLUSIONES Y RECOMENDACIONES

3.1 CONCLUSIONES

- Los resultados obtenidos en las pruebas del control computarizado son bastantes satisfactorios. En los circuitos no se presentaron problemas en su implementación. Con la realización de este proyecto se lograron los objetivos requeridos como estudiante. Con esto reafirmamos los conocimientos obtenidos en la electrónica tanto analógica como digital, específicamente en el área de los circuitos controlados por un microcontrolador PIC, así como el manejo y la comunicación serial mediante el interfaz 485. Se trabajó en el área de comunicaciones y electrónica de control y potencia.

- El proyecto realizado logra de manera eficiente, llevar a la práctica un sistema óptimo con una tecnología adecuada acorde a nuestros conocimientos y capacidades proporcionando un control automatizado para activar sistemas de iluminación, sensores, alarmas, cerraduras, etc, además de controlar los movimientos de un brazo robótico, controlar la producción en una fabrica de una determinada área, con esto beneficiamos el ahorro de tiempo al activar una o varias carga (sistemas de iluminación, sensores, alarmas, cerraduras, etc.).

- Se realizó muchas pruebas en el funcionamiento del equipo utilizando Hyper Terminal, el resultados que se obtuvo siempre fue satisfactorio, por tal motivo se puede considerar que este proyecto es apropiado para soportar el trabajo diario en cualquier lugar ya sea desde una casa hasta el control de un edificio.

- Este proyecto se convierte en una de las mejores opciones principales para el control ya que es de fácil manipulación, con esto cualquier persona que sepa manejar una computadora podrá hacer el uso de este proyecto.

- Dependiendo del área donde vaya a ser utilizado nuestro control computarizado se lo puede adaptar a las condiciones requeridas por el personal o por dicha área, como por ejemplo, en vez de manejar focos de pasillos podemos manejar el sistema de ventilación no existe ningún problema.

- Los objetivos planteados en este proyecto se han cumplido de una manera muy satisfactoria, ya que las pruebas realizadas para verificar el correcto funcionamiento del control de las cargas nos arrojó resultados positivos y principalmente la seguridad de confiar en el correcto desempeño del equipo construido.

3.2 RECOMENDACIONES

- Dado que el equipo construido consta de dos partes, es decir del conversor RS232 a RS485 y el circuito de control debemos tener siempre conectado una tierra común entre los dos equipos para el funcionamiento correcto, además para esto siempre debe estar encendido el computador.

- En las pruebas que se realizó al equipo se pudo comprobar que la corriente que absorbe el MAX 232 era muy elevado, por lo cual la entrada de ese pin (pin 11) se quemaba, por tal razón se recomienda utiliza una resistencia de bajo valor para disminuir la corriente, en nuestro proyecto utilizamos una resistencia de 10Ω.

- Se recomienda utilizar en el circuito de potencia unos buenos disipadores de calor y un ventilador, ya que cada triac soporta una potencia de 1000W por ende se produce demasiado calor. Esto es necesario para poder cuidar y alargar la vida útil de los triac utilizados.

- Se recomienda utilizar un voltaje de alimentación exactamente de +5V para la alimentación de los dos circuitos, tanto para el conversor RS232 a RS485 y el circuito de control, ya que una pequeña variación de voltajes (en el orden de los milisegundos), pueda enviar un dato erróneo y el funcionamiento no será el esperado. Y no es recomendable utilizar un adaptador de pared, ya que el voltaje de salida no siempre es el mismo del que indica su fabricante, por ultimo puede utilizar un circuito con un diodo zener de 5.1V o procure utilizar un regulador de voltaje como el 7805.

- Para la programación del PIC se recomienda tener instalado correctamente los programas antes mencionados a utilizar, es decir Microcode Studio y IC-Prog 1.05E, como ya se informó esto puede ser descargado gratuitamente mediante el Internet.

- Oscilador interno RC que posee no tiene muy buena precisión.

- Este proyecto esta diseñado para ser implementado en cualquier industria, dando así un gran beneficio para el control y siendo el comienzo de una serie de facilidades ya que no solo se puede apagar o encender cargas de distinta índole, sino también se lo podría realizar variaciones como las siguientes: cambiar de intensidad en el sistema de luminarias para poder ajustar la adecuada iluminación que la persona requiere en su casa, trabajo u otro sitio a utilizarse este control.

BIBLIOGRAFIA

- ✓ www.mecanique.co.uk
- ✓ www.IC-prog.com
- ✓ www.melabs.com
- ✓ www.microchip.com
- ✓ www.electronicaestudio.com
- ✓ http://www.canalvisualbasic.net/manual/tema1.asp
- ✓ http://www.elguille.info/vb/cursos_vb/basico/indice.htm
- ✓ http://www.aprendavb.blogspot.com/
- ✓ elp@i-micro.com
- ✓ http://picbasic.com/products/pbpis.htm
- ✓ http://es.wikipedia.org/wiki/Lenguaje_de_programación%C3%83n
- ✓ REYES, Carlos; *Microcontroladores PIC Programación en Basic 16F62X, 16F8XX, 16F87X*, Editorial Rispergraf C.A, Segunda edición; Ecuador 2006, Pág. 142-151.
- ✓ CAMPBELL, J. 1988. El libro del RS232. Anaya
- ✓ José M Angulo Usategui, Susana Romero Yesa e Ignacio Angulo Martínez "Microcontroladores PIC – Diseño práctico de aplicaciones segunda parte- PIC16F87x"**,** *ed. Graw Hill* 1ª Edición
- ✓ ESPINOZA, Hernán; *Sistemas de seguridad residencial mediante alarma y vía telefónica,* Escuela de Formación Tecnológica, 2005.
- ✓ *La ruta practica a VISUAL BASIC*, Editorial Macro ERIL, 1ra. Edición; Perú 2003, Pág. 10-24.